StarBurn SDK CD/DVD/Blu-Ray/HD-DVD Toolkit

Application Programming Interface

Table of Contents

1 Introduction

The architecture of this toolkit makes it possible to access a wide variety of CD/DVD/Blu-Ray/HD-DVD devices using a simple interface. Support for peripheral devices in Windows 95, Windows 98, Windows Me (Millennium Edition), Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008 R2 and Windows 7, Windows 2012 R2, Windows 8, Windows Server 2016, Windows 10 is normally achieved through device specific drivers layered on top of the operating systems' native SCSI support. Because of the great diversity of CD/DVD/Blu-Ray/HD-DVD devices, no driver can support all of them. Instead, different ways are needed for supporting each major class of installed CD/DVD/Blu-Ray/HD-DVD device. This toolkit isolates such a different handling inside and export the functions that allow the user to handle all of CD/DVD/Blu-Ray/HD-DVD devices in "generic" way when the user uses the same routines for example to burn the CD/DVD/Blu-Ray/HD-DVD on the devices that really requires very different actions to be performed for real process of burning. The goal of the toolkit is to hide all the differences in hardware from the user and provide easy and ready to go mechanism for working with CD/DVD/Blu-Ray/HD-DVD devices.

Before you begin your development efforts, be sure that you have a solid understanding of the SCSI and MMC specifications. You should understand what SCSI sense data is and how to treat the different conditions that can happen during CD/DVD/Blu-Ray/HD-DVD burning or grabbing process. Much of your success in developing a CD/DVD/Blu-Ray/HD-DVD burning, grabbing or mastering module is dependent on your understanding of these specifications.

1.1 Copyright

1.2 Trademarks

1.3 Changes

The material in this document is for information only and is subject to change without notice. While reasonable efforts have been made in the preparation of this document to assure its accuracy, StarBurn Software assumes no liability resulting from errors or omissions in this document, or from the use of the information contained herein. StarBurn Software reserves the

right to make changes in the product design without reservation and without notification to its users.

1.4 Technical support and service

If you have questions about installing or using this software, check this and other documents first - you will find answers to most of your questions here or there. If you need further assistance, please contact us.

2 Symbol Reference

2.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	StarBurn_CdvdBurnerGrabber_AuthorizeDVD (see page 17)	This function authorizes DVD that has CSS protection system.
	StarBurn_CdvdBurnerGrabber_AuthorizeDVDEx (see page 19)	This function authorizes DVD that has CSS protection system and keeps bus key for disc key.
	StarBurn_CdvdBurnerGrabber_Blank (see page 21)	This function erases the rewritable media inserted into CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_BluRayFormat (see page 23)	This function formats Blu-Ray rewritable media (BD-RE) inserted into the Blu-Ray burner device.
	StarBurn_CdvdBurnerGrabber_BluRayFormatQuery (see page 25)	This function queries available Blu-Ray formats for the currently inserted Blu-Ray rewritable media (BD-RE).
	StarBurn_CdvdBurnerGrabber_Cancel (see page 27)	This function cancels current read or write process on the CD/DVD/Blu-Ray/HD-DVD burner.
	StarBurn_CdvdBurnerGrabber_CloseSession (see page 29)	This function closes session on on CD/DVD/Blu-Ray/HD-DVD burner device after something was written using StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184) or StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172).
	StarBurn_CdvdBurnerGrabber_Create (see page 31)	This function creates CD/DVD/Blu-Ray/HD-DVD burner device object. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD releated actions.
	StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)	This function creates CD/DVD/Blu-Ray/HD-DVD burner device object using SPTI (SCSI Pass Through Interface) transport. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD related actions.
	StarBurn_CdvdBurnerGrabber_CreateExEx (see page 35)	This function creates CD/DVD/Blu-Ray/HD-DVD burner device object using SPTD (SCSI Pass Through Direct) transport. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD related actions.
	StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile (see page 37)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be one sound file or full set of sound files if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once PQ mode.
	StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileUnicode (see page 39)	This is function StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileUnicode.
	StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree (see page 40)	This function records ISO9660 or Joliet file system image located in file tree object with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once PQ mode.
	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile (see page 42)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be one sound file or full set of sound files if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once raw P-W mode.
	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileAudioUnicode (see page 45)	This API accepts track names in Unicode format and does AUDIO only DAO burn.
	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileUnicode (see page 46)	This is function StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileUnicode.

	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree (see page 46)	This function records ISO9660 or Joliet file system image located in file tree object with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once raw P-W mode.
	StarBurn_CdvdBurnerGrabber_Eject (see page 49)	This function ejects the media on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_ExecuteGeneric (see page 50)	This function executes custom CDB (Command Descriptor Block) on CD/DVD/Blu-Ray/HD-DVD burner device object. This could be used to deal with the propriatry hardware or execute some special code StarBurn does not support out-of-box.
	StarBurn_CdvdBurnerGrabber_GetAdvancedSupportedMediaFormats (see page 53)	This function returns CD/DVD/Blu-Ray/HD-DVD burner device object extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetBUP (see page 55)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device supports BUP (Buffer Underrun Protection) or not and what is BUP current status (if supported). This information can be used later to set BUP status before starting write operations.
	StarBurn_CdvdBurnerGrabber_GetDeviceInformation (see page 57)	This function returns CD/DVD/Blu-Ray/HD-DVD burner device object information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetDeviceInformationUnicode (see page 59)	This is function StarBurn_CdvdBurnerGrabber_GetDeviceInformationUnicode.
	StarBurn_CdvdBurnerGrabber_GetDiscFileSystem (see page 59)	This function detects recorded optical disc file system.
	StarBurn_CdvdBurnerGrabber_GetDiscFreeSpace (see page 60)	This function gets disc free space of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetDiscInformation (see page 62)	This function gets disc information of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetDiscUsedSpace (see page 64)	This function gets disc used space of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem (see page 66)	This function gets DVD protection system of inserted DVD media into DVD device.
	StarBurn_CdvdBurnerGrabber_GetDVDRegionMask (see page 68)	This function gets DVD region mask from DVD media inserted to CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.
	StarBurn_CdvdBurnerGrabber_GetInsertedDiscType (see page 70)	This function returns inserted to CD/DVD/Blu-Ray/HD-DVD burner device disc type. This data can be used to report it to the user and to select the stream type to record.
	StarBurn_CdvdBurnerGrabber_GetLastAddress (see page 71)	This function gets last recorded address from CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object. Such an address could be used for already recorded session import with the UDF mastering.
	StarBurn_CdvdBurnerGrabber_GetLastTrack (see page 73)	This function gets last recorded track from CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object. Such a track number could be used for already recorded session import.
	StarBurn_CdvdBurnerGrabber_GetMechanicalOptions (see page 75)	This function gets mechanical options (is device capable of programmatically load/eject and lock/unlock etc) from CD/DVD/Blu-Ray/HD-DVD burner device object.
	StarBurn_CdvdBurnerGrabber_GetMediaTrayStatus (see page 77)	This function gets media (inserted or not) and tray (opened or closed) statuses from CD/DVD/Blu-Ray/HD-DVD burner device object.
	StarBurn_CdvdBurnerGrabber_GetNumberOfSystemDescriptors (see page 79)	This function gets number of system descriptors in system structures stream. This value is used for updating head of the image when creating signle track session import code.
	StarBurn_CdvdBurnerGrabber_GetRPC (see page 81)	This function gets RPC (region play code) and some additional DVD region management information from "DVD part" of the CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.
	StarBurn_CdvdBurnerGrabber_GetSpeeds (see page 83)	This function gets current read and write and top supported read and write speeds on CD/DVD/Blu-Ray/HD-DVD burner device. This speeds later will be used to set current speeds during all read and write operations.

	StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats (see page 85)	This function returns CD/DVD/Blu-Ray/HD-DVD burner device object extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsEx (see page 88)	This function returns DVD+R(W) burner device object extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsExEx (see page 90)	This function returns DVD+R(W) burner device object extended extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetTOCInformation (see page 92)	This function gets TOC information of the currently inserted disc in the CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94)	This function gets track information of the asked track number on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetTrackInformationEx (see page 96)	This function gets track extended information of the asked track number on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GrabCD (see page 98)	This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format.
	StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)	This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files.
	StarBurn_CdvdBurnerGrabber_GrabDVDEx (see page 102)	This function grabs disc image (extended) from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files. StarBurn_CdvdBurnerGrabber_GrabDVDEx function for all I/O files operations use user's callback functions.
	StarBurn_CdvdBurnerGrabber_GrabDVDFast (see page 105)	This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files. Works faster than StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)
	StarBurn_CdvdBurnerGrabber_GrabRange (see page 108)	This function grabs range of logical blocks from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GrabTrack (see page 110)	This function grabs track from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GrabTrackEx (see page 112)	This function grabs track (extended) from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device. StarBurn_CdvdBurnerGrabber_GrabTrackEx function for all I/O files operations use user's callback functions.
	StarBurn_CdvdBurnerGrabber_GrabTrackUnicode (see page 115)	This is function StarBurn_CdvdBurnerGrabber_GrabTrackUnicode.
	StarBurn_CdvdBurnerGrabber_IsDiscBlank (see page 115)	This function detect blank state of currently inserted disc in CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_Load (see page 117)	This function loads the media on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_LoadEx (see page 119)	This function loads the media on CD/DVD/Blu-Ray/HD-DVD burner device. Unlike StarBurn_CdvdBurnerGrabber_Load (see page 117)(...) this API call allows to control should it return immediately or wait until the drive would become into ready state.
	StarBurn_CdvdBurnerGrabber_Lock (see page 121)	This function locks the media inside the CD/DVD/Blu-Ray/HD-DVD device.
	StarBurn_CdvdBurnerGrabber_MSFToLBA (see page 123)	This function converts MSF (minute:second:frame) address into the LBA (logical block address).
	StarBurn_CdvdBurnerGrabber_ProbeSupportedReadModes (see page 123)	This function probes supported read modes on the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_ProbeSupportedWriteModes (see page 126)	This function probes supported read modes on the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_ReadATIP (see page 128)	This function reads ATIP information from the media currently inserted into CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.
	StarBurn_CdvdBurnerGrabber_ReadCooked (see page 130)	This function reads one or more logical block(s) from the currently inserted CD/DVD/Blu-Ray/HD-DVD in cooked format.
	StarBurn_CdvdBurnerGrabber_ReadLB (see page 132)	This function reads one logical block from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_ReadRaw (see page 134)	This function reads one or more logical block(s) from the currently inserted CD in RAW format.

	StarBurn_CdvdBurnerGrabber_Release (see page 137)	This function releases the media inside the CD/DVD/Blu-Ray/HD-DVD device after the media was locked before.
	StarBurn_CdvdBurnerGrabber_SendOPC (see page 139)	This function sends OPC (Optimum Power Calibration) for current write speed on CD/DVD/Blu-Ray/HD-DVD burner device and current CD/DVD/Blu-Ray/HD-DVD media. The set laser level will be used later during all write operations.
	StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141)	This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) or ISO image with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Session-At-Once mode.
	StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW (see page 143)	This function records raw image with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in raw Session-At-Once mode.
	StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWUnicode (see page 146)	This is function StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW Unicode.
	StarBurn_CdvdBurnerGrabber_SessionAtOnceUnicode (see page 146)	This is function StarBurn_CdvdBurnerGrabber_SessionAtOnceUnicode.
	StarBurn_CdvdBurnerGrabber_SetBUP (see page 147)	This function sets BUP (Buffer Underrun Protection) on CD/DVD/Blu-Ray/HD-DVD burner device that supports BUP. BUP must be enabled for successful completion of write operations.
	StarBurn_CdvdBurnerGrabber_SetCDTextItem (see page 149)	Sets CD-Text item for the specified CD/DVD/Blu-Ray/HD-DVD device.
	StarBurn_CdvdBurnerGrabber_SetReadSpeed (see page 151)	This function sets read speed for CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object.
	StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153)	This function sets current read and write speeds on CD/DVD/Blu-Ray/HD-DVD burner device This speeds later will be used during all read and write operations.
	StarBurn_CdvdBurnerGrabber_StopPlayScan (see page 155)	This function resets media state inside CD/DVD/Blu-Ray/HD-DVD burner device object. This actions allow to update disc content w/o need to eject the disc. Could be used for multiple burnings w/o disc eject or manual disc data verification.
	StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157)	This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDEx (see page 159)	This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDExEx (see page 162)	This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDExExUnicode (see page 164)	This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDExExUnicode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDExUnicode (see page 165)	This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDExUnicode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDUnicode (see page 165)	This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDUnicode.
	StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations.
	StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see page 168)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations. The only difference between this and simple StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166)() is that this code passed parameter to wait for unit to become ready. And "simple" code uses default timeout of 20 seconds.
	StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx (see page 170)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations. This "double extended" variant of the call also can wait for particular amount of seconds and do "fast exit" if there's no disc in drive.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be sound file if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicode (see page 174)	This API call deals with audio tracks with Unicode names only.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicodeEx (see page 175)	This API call deals with audio tracks with Unicode names only. Also it allows to set track-to-track pause length.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileEx (see page 176)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be sound file if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileUnicode (see page 178)	This is function StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileUnicode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemory (see page 179)	This function records ISO9660 or Joliet file system image or audio stream located in a memory with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemoryEx (see page 181)	This function records ISO9660 or Joliet file system image or audio stream located in a memory with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode. If audio content is recorded this API call can change default Track-At-Once track-to-track pause length from 150 logical blocks (2 seconds) to any passed one.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184)	This function records ISO9660 or Joliet file system image located in a pipe with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx (see page 186)	This function records ISO9660 or Joliet file system image or audio stream located in a pipe with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189)	This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTreeEx (see page 191)	This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192)	This function parses the UDF file system on disc and calls a callback function for each found file. The information about LBAs that file resides on is passed to callback function.
	StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193)	This function parses the UDF file system on disc and calls a callback function for each found file and directory. The information about LBAs that file resides on is passed to callback function.
	StarBurn_CdvdBurnerGrabber_VerifyFile (see page 194)	This function verifies recorded file system image on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular image recording (or maybe other image). This information can be used to be sure recorded disc is readable and really contains recorded information.
	StarBurn_CdvdBurnerGrabber_VerifyFileEx (see page 196)	This function verifies recorded file system image on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular image recording (or maybe other image). This information can be used to be sure recorded disc is readable and really contains recorded information.
	StarBurn_CdvdBurnerGrabber_VerifyFileExUnicode (see page 198)	This is function StarBurn_CdvdBurnerGrabber_VerifyFileExUnicode.
	StarBurn_CdvdBurnerGrabber_VerifyFileUnicode (see page 198)	This is function StarBurn_CdvdBurnerGrabber_VerifyFileUnicode.
	StarBurn_CdvdBurnerGrabber_VerifyTree (see page 199)	This function verifies recorded file system tree on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular tree recording. This information can be used to be sure recorded disc is readable and really contains recorded information.
	StarBurn_CdvdBurnerGrabber_VerifyTreeEx (see page 201)	This function verifies recorded file system tree on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular tree recording. This information can be used to be sure recorded disc is readable and really contains recorded information.
	StarBurn_CdvdBurnerGrabber_VideoCD (see page 203)	This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

	StarBurn_CdvdBurnerGrabber_VideoCDEx (see page 205)	This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_VideoCDExEx (see page 208)	This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_VideoCDExExUnicode (see page 210)	This is function StarBurn_CdvdBurnerGrabber_VideoCDExExUnicode.
	StarBurn_CdvdBurnerGrabber_VideoCDExUnicode (see page 211)	This is function StarBurn_CdvdBurnerGrabber_VideoCDExUnicode.
	StarBurn_CdvdBurnerGrabber_VideoCDUnicode (see page 211)	This is function StarBurn_CdvdBurnerGrabber_VideoCDUnicode.
	StarBurn_CorrectISO9660Name_Default (see page 212)	This function generates next ISO9660 file name to make it unique.
	StarBurn_CorrectJolietName_Default (see page 212)	This function generates next Joliet file name to make it unique.
	StarBurn_CreatePipe (see page 213)	This function creates pipe for all of the StarBurn pipe operations and returns both "read" and "write" handles for it.
	StarBurn_Destroy (see page 214)	This function frees allocated memory in the way of destroying the object that toolkit created internally. This is universal call, it frees all the objects and does not care about object type.
	StarBurn_DestroyPipe (see page 215)	This function closes both "read" and "write" pipe handles.
	StarBurn_DownShut (see page 216)	This function uninitialize burning toolkit. It's expected to be called after very last function call before exiting from StarBurn client application. It's a good idea to gather the trash before exiting. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used.
	StarBurn_DVDVideo_Create (see page 216)	This function creates DVD-Video file system object from passed pointer to VIDEO_TS directory with DVD corresponding files.
	StarBurn_DVDVideo_CreateUnicode (see page 218)	This is function StarBurn_DVDVideo_CreateUnicode.
	StarBurn_DVDVideo_Destroy (see page 218)	This function destroys DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.
	StarBurn_DVDVideo_GetSizeInUCHARs (see page 219)	This function gets file system content size in UCHARs from DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.
	StarBurn_DVDVideo_GetTreePointer (see page 219)	This function returns pointer to ISO9660 file tree object embedded to DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.
	StarBurn_DVDVideo_PatchHeader (see page 220)	This function patches IFO/BUF file header RBA for BUP.
	StarBurn_DVDVideo_PatchTable (see page 221)	This function patches IFO/BUF file table RBAs.
	StarBurn_DVDVideo_Read (see page 221)	This function reads requested number of UCHARs from DVD-Video file system object (created with StarBurn_DVDVideo_Create (see page 216) API call) from it's current read position.
	StarBurn_DVDVideo_SeekToBegin (see page 222)	This function moves DVD-Video file system object (created with StarBurn_DVDVideo_Create (see page 216) API call) current read position to the very beginning.
	StarBurn_ElTorito_BootCatalogAddSection (see page 223)	This function creates ElTorito boot catalog (ANSI names).
	StarBurn_ElTorito_BootCatalogAddSectionUnicode (see page 223)	This function creates ElTorito boot catalog (UNICODE names).
	StarBurn_ElTorito_CreateBootCatalog (see page 224)	This function creates ElTorito boot catalog (ANSI names).
	StarBurn_ElTorito_CreateBootCatalogUnicode (see page 225)	This function creates ElTorito boot catalog (UNICODE names).
	StarBurn_FileClose (see page 225)	This fucntion closes file handle embedded to file object.
	StarBurn_FileCreate (see page 226)	This function creates new file from the passed file path and name.
	StarBurn_FileInitialize (see page 226)	This function initializes file handle embedded to file object with bad file handle value.
	StarBurn_FileOpen (see page 226)	This function opens existing file from the passed file path and name.
	StarBurn_FileOpenEx (see page 227)	This function opens existing file from the passed file path and name.
	StarBurn_FileRead (see page 227)	This function reads asked amount of unsigned chars from file at current seek position.
	StarBurn_FileReWind (see page 228)	This function rewinds file position in unsigned chars to the very beginning for the file.

	StarBurn_FileSeek (see page 228)	This function sets file position in unsigned chars to required one.
	StarBurn_FileSeekRead (see page 229)	This function sets file position in unsigned chars to required one and reads requested amounts unsigned chars.
	StarBurn_FileSizeInUCHARsGet (see page 229)	This function gets file size in unsigned chars.
	StarBurn_FileUnicodeCreate (see page 230)	This function creates new file from the passed UNICODE file path and name.
	StarBurn_FileUnicodeOpen (see page 230)	This function opens existing file from the passed UNICODE file path and name.
	StarBurn_FileUnicodeOpenEx (see page 230)	This function opens existing file from the passed UNICODE file path and name.
	StarBurn_FileWrite (see page 231)	This function writes asked amount of unsigned chars to file at current seek position
	StarBurn_FindDevice (see page 231)	This function finds device of the specified device type.
	StarBurn_GetAudioFileStreamSizeInUCHARs (see page 233)	This function returns audio stream size in UCHARs for passed supported as source audio file.
	StarBurn_GetAudioFileStreamSizeInUCHARs_Fast (see page 234)	This function returns audio stream size in UCHARs for passed supported as source audio file.
	StarBurn_GetAudioFileStreamSizeInUCHARs_UnicodeFast (see page 235)	This is function StarBurn_GetAudioFileStreamSizeInUCHARs_UnicodeFast.
	StarBurn_GetAudioFileStreamSizeInUCHARsUnicode (see page 236)	This is function StarBurn_GetAudioFileStreamSizeInUCHARsUnicode.
	StarBurn_GetBufferUnderrunTimeOutInMs (see page 236)	This function returns buffer underrun timeout in milliseconds. It's amount of time StarBurn core would wait before resubmitting each command to the drive if burning was faster then reading and StarBurn had exhausted software cache it uses for buffering. Modern hardware has BUP (Buffer Underrun Protection) and would perfectly survive in such a condition and keep burning disc still usable.
	StarBurn_GetDeviceLetter (see page 237)	This function returns device letter (device root path) for a device name, like the one used in StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)(...) API call.
	StarBurn_GetDeviceLetterUnicode (see page 237)	This is function StarBurn_GetDeviceLetterUnicode.
	StarBurn_GetDeviceNameByDeviceAddress (see page 238)	This function gets device name by device address.
	StarBurn_GetDeviceTimeOutByDeviceAddress (see page 239)	This function gets device timeout in seconds by device SCSI address.
	StarBurn_GetDVDPadding (see page 239)	This function returns is DVD padding mode (when at least 1GB of the data would be recorded to DVD media - required for DVD-Video compilations to work properly on standalone DVD players) enabled (TRUE) or disabled (FALSE).
	StarBurn_GetDVDPLUSRDLCompatibleMode (see page 240)	This function returns is DVD+R DL (Dual Layer) so-called "compatible mode" (when layer would be switched exactly at 1/2 of the data payload recorded on the disc - required for DVD-Video compilations to work properly) enabled (TRUE) or disabled (FALSE).
	StarBurn_GetEjectAfterFail (see page 241)	This function returns is so-called "eject after fail" (if during burning process error would happen - should StarBurn eject disc or not) enabled (TRUE) or disabled (FALSE).
	StarBurn_GetFastReadTOC (see page 241)	This function returns is so-called "fast read TOC" (TOC information is not 100% accurate - to get EXACT track length StarBurn trys to read beginning and end of the track to find sub-channel index switch indicating EXACT track beginning or EXACT track end) mode enabled (TRUE) or disabled (FALSE).
	StarBurn_GetId (see page 242)	This function gets ASPI layer ID strings.
	StarBurn_GetIs64KBIO (see page 242)	This function returns is so-called "64KB I/O" (when StarBurn issues 64KB I/O packets instead of the 32KB ones) currently enabled (TRUE) or disabled (FALSE).
	StarBurn_GetIsCollisionDetectionDisabled (see page 243)	This function returns is collision detection disabled (TRUE) or enabled (FALSE). If collision detection is enabled - every new file added to ISO9660 or Joliet file tree would be checked to have unique name. If such a file already exist - special callback (collision detection one) would be called so user would be able to either replace or rename new or old file. However if collision detection is disabled - no comparison and no actions would be performed. File would be just added to the destination file system image AS IS, having it's original name.

	StarBurn_GetIsSafeGrabbingEnabled (see page 243)	This function returns is so-called "safe grabbing" (when StarBurn would mix read commands with checking for device to become ready - required to workaround broken ATAPI-to-USB bridges) enabled (TRUE) or disabled (FALSE).
	StarBurn_GetVersion (see page 244)	This function returns a 32-bit unsigned long value that contains packed numbers that represent the year, the month and the date of the toolkit build. So 10th of March, year of 1978 will look like 0x19780310.
	StarBurn_GetVolumeIDs (see page 245)	This function gets volume ID from ISO9660/Joliet volume name.
	StarBurn_ImageFile_UDFFileSystemLookup (see page 245)	This function parses the UDF file system inside image file and calls a callback function for each found file. The information about LBAs that file resides on is passed to callback function.
	StarBurn_ImageFile_UDFFileSystemLookupEx (see page 246)	This function parses the UDF file system inside image file and calls a callback function for each found file and directory. The information about LBAs that file resides on is passed to callback function.
	StarBurn_IsAudioFileSupported (see page 246)	This function returns is currently passed file supported as source audio file.
	StarBurn_IsAudioFileSupportedUnicode (see page 247)	This is function StarBurn_IsAudioFileSupportedUnicode.
	StarBurn_IsLocalDateTimeUsed (see page 248)	This is function StarBurn_IsLocalDateTimeUsed.
	StarBurn_ISO2_Check1999Name (see page 248)	This function checks if given string valid ISO9660:1999 file (or directory) name
	StarBurn_ISO2_CheckJolietName (see page 248)	This function checks if given string valid ISO9660 Joliet file (or directory) name
	StarBurn_ISO2_CheckLevel1Name (see page 249)	This function checks if given string valid ISO9660 Level 1 file (or directory) name
	StarBurn_ISO2_CheckLevel2Name (see page 249)	This function checks if given string valid ISO9660 Level 2 file (or directory) name
	StarBurn_ISO2_CheckRelaxedJolietName (see page 250)	This function checks if given string valid ISO9660 Joliet file (or directory) name with less restrictions: names may contain more than one dot.
	StarBurn_ISO2_CreateNewName (see page 250)	This function creates new ANSI name by adding ~XX (where XX is a number (Seed)) at the end of file name.
	StarBurn_ISO2_DirectoryBrowse (see page 250)	This function browses ISO9660 directory content.
	StarBurn_ISO2_DirectoryCreate (see page 251)	This function creates ISO9660 directory.
	StarBurn_ISO2_DirectoryDestroy (see page 251)	This function destroys created ISO9660 directory object.
	StarBurn_ISO2_DirectoryProcess (see page 252)	This function processes single ISO9660 directory.
	StarBurn_ISO2_DirectoryQuery (see page 252)	This function do query for ISO9660 file properties.
	StarBurn_ISO2_DirectoryRootCreate (see page 253)	This function creates root ISO9660 directory.
	StarBurn_ISO2_DirectoryUnicodeCreate (see page 253)	This function creates UNICODE ISO9660 directory.
	StarBurn_ISO2_DirectoryUnicodeProcess (see page 254)	This function processes single UNICODE ISO9660 directory.
	StarBurn_ISO2_FileCreate (see page 255)	This function creates ISO9660 file.
	StarBurn_ISO2_FileDestroy (see page 255)	This function destroys created ISO9660 file object.
	StarBurn_ISO2_FileDirectoryDateTimeGet (see page 256)	This function gets date and time from created ISO9660 file or directory object.
	StarBurn_ISO2_FileDirectoryDateTimeSet (see page 256)	This function sets date and time on created ISO9660 file or directory object.
	StarBurn_ISO2_FileDirectoryNameSet (see page 256)	This function set new name for ISO9660 file or directory object.
	StarBurn_ISO2_FileDirectoryUnicodeNameSet (see page 257)	This function set new UNICODE name for ISO9660 file or directory object.
	StarBurn_ISO2_FileMemoryCreate (see page 257)	This function creates ISO9660 memory file.
	StarBurn_ISO2_FileMemoryUnicodeCreate (see page 258)	This function creates Unicode ISO9660 memory file.
	StarBurn_ISO2_FileQuery (see page 258)	This function do query for ISO9660 file properties.
	StarBurn_ISO2_FileSetAttributes (see page 259)	This function sets attributes for ISO9660 file.
	StarBurn_ISO2_FileUnicodeCreate (see page 259)	This function creates UNICODE ISO9660 file.

	StarBurn_ISO2_ImpVolumeCreate (see page 260)	This function processes contents of single UNICODE ISO9660 directory. __stdcall StarBurn_ISO2_DirectoryContentsUnicodeProcess(const unsigned short *UnicodePathName, unsigned long *GUID, void *Root, PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK (see page 587) Callback, const void *Context, STARBURN_ISO2_FILE_DATE_TIME *DateTime); This function creates ISO9660 import volume.
	StarBurn_ISO2_ImpVolumeDestroy (see page 261)	This function destroys created ISO9660 import volume object.
	StarBurn_ISO2_ImpVolumeImport (see page 261)	This function imports ISO9660 import volume content.
	StarBurn_ISO2_ISO9660FileTreeCreate (see page 262)	This function create ISO image from already created ISO9660 volume.
	StarBurn_ISO2_VolumeCreate (see page 262)	This function creates ISO9660 volume.
	StarBurn_ISO2_VolumeCreateBoot (see page 263)	This function creates ISO9660 bootable volume (ANSI names).
	StarBurn_ISO2_VolumeCreateBootElTorito (see page 264)	This function creates ISO9660 bootable volume (ANSI names).
	StarBurn_ISO2_VolumeCreateEx (see page 265)	This function creates ISO9660 volume (with both Unicode and ANSI names).
	StarBurn_ISO2_VolumeCreateExBoot (see page 266)	This function creates ISO9660 bootable volume (with both Unicode and ANSI names).
	StarBurn_ISO2_VolumeCreateUnicode (see page 267)	This function creates ISO9660 volume.
	StarBurn_ISO2_VolumeCreateUnicodeBoot (see page 268)	This function creates ISO9660 bootable volume (Unicode names).
	StarBurn_ISO2_VolumeCreateUnicodeBootElTorito (see page 269)	This function creates ISO9660 bootable volume (Unicode names).
	StarBurn_ISO2_VolumeDescriptorsBufferGet (see page 270)	This function get ISO9660 volume descriptors buffer.
	StarBurn_ISO2_VolumeDestroy (see page 270)	This function destroys created ISO9660 volume object.
	StarBurn_ISO2_VolumeSeekRead (see page 271)	This function seeks to passed offset and reads requested amount of data.
	StarBurn_ISO2_VolumeSizeInUCHARsGet (see page 271)	This function get ISO9660 volume size in unsigned chars.
	StarBurn_ISO9660JolietFileTree_Add (see page 272)	This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.
	StarBurn_ISO9660JolietFileTree_AddEx (see page 274)	This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.
	StarBurn_ISO9660JolietFileTree_AddMemory (see page 277)	This function adds the node created from the memory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.
	StarBurn_ISO9660JolietFileTree_AddW (see page 279)	This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations. Unlike other tree management calls this one deals with Unicode names (wide char variant of basic call).
	StarBurn_ISO9660JolietFileTree_AddZeroFile (see page 282)	This function adds the new "filled with zeros" file to already created ISO9660 or Joliet file tree. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.

	StarBurn_ISO9660JolietFileTree_BuildImage (see page 283)	This function builds ISO9660 or Joliet file system image from ISO9660 or Joliet file tree (assign all the logical block offsets and allocates and initializes all file system internal structures). Later this tree can be used to store the "virtual" ISO9660 or Joliet image to the file on the disk or to burn this ISO9660 or Joliet image directly to the CD/DVD/Blu-Ray/HD-DVD media.
	StarBurn_ISO9660JolietFileTree_BuildImageEx (see page 285)	This function builds ISO9660 or Joliet file system image from ISO9660 or Joliet file tree (assign all the logical block offsets and allocates and initializes all file system internal structures). Later this tree can be used to store the "virtual" ISO9660 or Joliet image to the file on the disk or to burn this ISO9660 or Joliet image directly to the CD/DVD/Blu-Ray/HD-DVD media. It's identical to StarBurn_ISO9660JolietFileTree_BuildImage (see page 283) except has some extra parameter to put into file system descriptor.
	StarBurn_ISO9660JolietFileTree_Cancel (see page 287)	This function cancel ISO9660 or Joliet file tree creation process.
	StarBurn_ISO9660JolietFileTree_Create (see page 288)	This function allocates in the memory and initializes ISO9660 or Joliet file tree. Later this file tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be recorder directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate buffering operations on the hard disk.
	StarBurn_ISO9660JolietFileTree_GetAttributes (see page 290)	This function returns ISO9660 or Joliet file tree node attributes.
	StarBurn_ISO9660JolietFileTree_GetAutoSorting (see page 292)	This function returns ISO9660 or Joliet file tree node sorting mode.
	StarBurn_ISO9660JolietFileTree_GetFileSizeInUCHARs (see page 292)	This function returns ISO9660 or Joliet file tree file size in UCHARs.
	StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)	This function returns ISO9660 or Joliet file tree node first kid pointer.
	StarBurn_ISO9660JolietFileTree_GetFullPath (see page 296)	This function returns full path to ISO9660 or Joliet file tree node.
	StarBurn_ISO9660JolietFileTree_GetKidType (see page 297)	This function returns type of ISO9660 or Joliet file tree node.
	StarBurn_ISO9660JolietFileTree_GetLevel (see page 297)	This function returns ISO9660 or Joliet file tree level. The number of levels in the created tree.
	StarBurn_ISO9660JolietFileTree_GetNames (see page 299)	This function returns ISO9660 or Joliet file tree node names (long, short and absolute).
	StarBurn_ISO9660JolietFileTree_GetNamesEx (see page 300)	This function returns ISO9660 or Joliet file tree node names (long, short and absolute). Also it returns file system dependent names in ANSI and Unicode.
	StarBurn_ISO9660JolietFileTree_GetNamesW (see page 302)	StarBurn_ISO9660JolietFileTree_GetNamesW function is not documented
	StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)	This function returns ISO9660 or Joliet file tree root node next kid pointer.
	StarBurn_ISO9660JolietFileTree_GetNodeISO9660DateTime (see page 305)	This function returns ISO9660 or Joliet file tree node ISO9660 time stamp (date and time).
	StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)	This function returns ISO9660 or Joliet file tree node power (file size) in UCHARs.
	StarBurn_ISO9660JolietFileTree_GetNodeStartingLBA (see page 307)	This function returns ISO9660 or Joliet file tree node starting LBA after the image was built.
	StarBurn_ISO9660JolietFileTree_GetNodeSystemTime (see page 308)	This function returns ISO9660 or Joliet file tree node SYSTEMTIME.
	StarBurn_ISO9660JolietFileTree_GetParent (see page 309)	This function returns ISO9660 or Joliet file tree node parent pointer.
	StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)	This function returns ISO9660 or Joliet file tree root node pointer.
	StarBurn_ISO9660JolietFileTree_GetSizeInUCHARs (see page 312)	This function returns ISO9660 or Joliet built file system image size in UCHARs.
	StarBurn_ISO9660JolietFileTree_ImportFile (see page 314)	This function imports file for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_ImportTrack (see page 316)	This function imports track for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_Read (see page 318)	This function reads ISO9660 or Joliet file system image from current offset into the user's data buffer. ISO9660 or Joliet file tree must have assigned all the logical block offsets and all file system internal structures allocated and initialized. The data that is read can be either stored in the file on the hard disk or directly written to CD/DVD/Blu-Ray/HD-DVD media.

	StarBurn_ISO9660JolietFileTree_Remove (see page 320)	This function deletes ISO9660 or Joliet file tree node and all it's kids.
	StarBurn_ISO9660JolietFileTree_SeekToBegin (see page 322)	This function seeks the ISO9660 or Joliet file system image from current offset to the very beginning so read operation will start from the 0 offset of the file system image. ISO9660 or Joliet file tree must have assigned all the logical block offsets and all file system internal structures allocated and initialized.
	StarBurn_ISO9660JolietFileTree_SetAttributes (see page 324)	This function sets ISO9660 or Joliet file tree node attributes.
	StarBurn_ISO9660JolietFileTree_SetAutoSorting (see page 325)	This function sets ISO9660 or Joliet file tree node auto sorting mode.
	StarBurn_ISO9660JolietFileTree_SetBootImage (see page 326)	This function sets boot image for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_SetBootImageEx (see page 328)	This extended function sets boot image for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_SetNames (see page 330)	This function sets ISO9660 or Joliet file tree node names (long, short and absolute).
	StarBurn_SetBufferUnderrunTimeOutInMs (see page 332)	This function sets buffer underrun timeout in milliseconds. It's amount of time StarBurn core would wait before resubmitting each command to the drive if burning was faster then reading and StarBurn had exhausted software cache it uses for buffering. Modern hardware has BUP (Buffer Underrun Protection) and would perfectly survive in such a condition and keep burning disc still usable.
	StarBurn_SetDeviceTimeOutByDeviceAddress (see page 333)	This function sets device timeout in seconds by device SCSI address.
	StarBurn_SetDVDPadding (see page 333)	This function sets DVD padding mode (when at least 1GB of the data would be recorded to DVD media - required for DVD-Video compilations to work properly on standalone DVD players) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetDVDPLUSRDLCompatibleMode (see page 334)	This function sets DVD+R DL (Dual Layer) so-called "compatible mode" (when layer would be switched exactly at 1/2 of the data payload recorded on the disc - required for DVD-Video compilations to work properly) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetEjectAfterFail (see page 335)	This function sets so-called "eject after fail" (if during burning process error would happen - should StarBurn eject disc or not) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetFastReadTOC (see page 335)	This function sets so-called "fast read TOC" (TOC information is not 100% accurate - to get EXACT track length StarBurn try to read beginning and end of the track to find sub-channel index switch indicating EXACT track beginning or EXACT track end) mode to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetIs64KBIO (see page 336)	This function sets current so-called "64KB I/O" (when StarBurn issues 64KB I/O packets instead of the 32KB ones) to enabled (TRUE) or disabled (FALSE) state.
	StarBurn_SetIsCollisionDetectionDisabled (see page 336)	This function sets collision detection to disabled (TRUE) or enabled (FALSE). If collision detection is enabled - every new file added to ISO9660 or Joliet file tree would be checked to have unique name. If such a file already exist - special callback (collision detection one) would be called so user would be able to either replace or rename new or old file. However if collision detection is disabled - no comparison and no actions would be performed. File would be just added to the destination file system image AS IS, having it's original name.
	StarBurn_SetIsSafeGrabbingEnabled (see page 337)	This function sets so-called "safe grabbing" (when StarBurn would mix read commands with checking for device to become ready - required to workaround broken ATAPI-to-USB bridges) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetUsingLocalDateTime (see page 337)	This is function StarBurn_SetUsingLocalDateTime.
	StarBurn_SetUsingUTCDateTime (see page 338)	This is function StarBurn_SetUsingUTCDateTime.
	StarBurn_sprintf_s (see page 338)	This function wrap runtime sprintf_s function. In Visual Studio 2003 function sprintf_s not present.
	StarBurn_SPTD_GetVersion (see page 338)	This function checks if the SPTD (SCSI Pass Through Direct) driver is present and gets SPTD driver version.
	StarBurn_StarPort_DeviceAddLocal (see page 339)	This function adds local device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.

	StarBurn_StarPort_DeviceAddLocalEx (see page 340)	This function adds local device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Device can be added persistent.
	StarBurn_StarPort_DeviceAddRemote (see page 341)	This function adds remote device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Device can be mount persistent.
	StarBurn_StarPort_DeviceAddRemoteEx (see page 341)	This function adds remote device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.
	StarBurn_StarPort_DeviceListQueryLocal (see page 342)	This function gets device list from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Unlike StarBurn_StarPort_DeviceListQueryRemote (see page 343) API call which retrieves information about REMOTE device list this call gets information about LOCAL device list.
	StarBurn_StarPort_DeviceListQueryRemote (see page 343)	This function gets device list from remote StarWind iSCSI target. StarWind exports RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) targets. Unlike StarBurn_StarPort_DeviceListQueryLocal (see page 342) API call which retreives information about LOCAL device list this call gets information about REMOTE device list.
	StarBurn_StarPort_DeviceRemove (see page 344)	This function removes device from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.
	StarBurn_StarPort_DeviceRemoveEx (see page 344)	This function removes device from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.
	StarBurn_StarPort_GetDeviceInformation (see page 345)	This function retrieves device type, vendor ID, product ID and revision for StarPort device by its target ID.
	StarBurn_StarPort_GetDeviceLetter (see page 346)	This function retrieves device letter (symbolic link) for StarPort device by its target ID.
	StarBurn_StarPort_GetDeviceSCSIAddress (see page 347)	This function retrieves device SCSI address for StarPort device by its target ID.
	StarBurn_StarPort_GetVersion (see page 347)	This function checks if the StarPortLite driver is installed and that its API version is equal to the one StarBurn was built with.
	StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348)	This function seeks all of the internal object pointers to the very beginning of it so following read operations would start from the very beginning instead of the currently set position.
	StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349)	This function creates compressed file reader object from passed compressed audio file name.
	StarBurn_StarWave_CompressedFileReaderObjectCreateUnicode (see page 350)	This is function StarBurn_StarWave_CompressedFileReaderObjectCreateUnicode.
	StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350)	This function destroys compressed file reader object.
	StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351)	This function reads uncompressed payload data chunk from underlying compressed file reader object.
	StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352)	This function returns uncompressed payload size in UCHARs for underlying compressed file reader object.
	StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet_Fast (see page 353)	This function returns uncompressed payload size in UCHARs for underlying compressed file reader object.
	StarBurn_StarWave_CompressedFileReaderObjectUnicodeCreate (see page 354)	This is function StarBurn_StarWave_CompressedFileReaderObjectUnicodeCreate.
	StarBurn_StarWave_CompressedFileRecompress (see page 354)	This function recompresse already compressed file to new one with another compression.
	StarBurn_StarWave_CompressedFileRecompressUnicode (see page 355)	This is function StarBurn_StarWave_CompressedFileRecompressUnicode.
	StarBurn_StarWave_CompressedFileSupportedIs (see page 355)	This function checks is file name actually points to supported compressed audio file.
	StarBurn_StarWave_CompressedFileUncompress (see page 356)	This function uncompresses compressed file to either WAV or RAW PCM.

	StarBurn_StarWave_CompressedFileUncompressUnicode (see page 357)	This is function StarBurn_StarWave_CompressedFileUncompressUnicode.
	StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358)	This function creates compressed file writer object.
	StarBurn_StarWave_CompressedFileWriterObjectCreateUnicode (see page 359)	This is function StarBurn_StarWave_CompressedFileWriterObjectCreateUnicode.
	StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359)	This function destroys compressed file writer object.
	StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360)	This function writes uncompressed data chunk to compressed file writer object.
	StarBurn_StarWave_UncompressedFileCompress (see page 361)	This function compresses uncompressed file to new one with asked compression.
	StarBurn_StarWave_UncompressedFileCompressUnicode (see page 362)	This is function StarBurn_StarWave_UncompressedFileCompressUnicode.
	StarBurn_StarWave_UncompressedFileSupportedIs (see page 362)	This function checks is file name actually points to supported uncompressed audio file.
	StarBurn_StarWave_UncompressedFileSupportedUnicodeIs (see page 363)	This function checks is file name actually points to supported uncompressed audio file.
	StarBurn_StarWave_UncompressedFileSupportedUnicodeIsEx (see page 364)	This function checks is file name actually points to supported uncompressed audio file.
	StarBurn_StarWave_VersionGet (see page 364)	This function returns StarWave library version.
	StarBurn_StarWave2_Convert (see page 365)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_ConvertEx (see page 366)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_ConvertExUnicode (see page 367)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_ConvertUnicode (see page 367)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_EncodeMP3OGGFromWAV (see page 368)	This function compress audio file from WAV (44100Hz, 16, stereo) format to MP3 and OGG format.
	StarBurn_StarWave2_EncodeMP3OGGFromWAVUnicode (see page 369)	This is function StarBurn_StarWave2_EncodeMP3OGGFromWAVUnicode.
	StarBurn_StarWave2_IsFileSupported (see page 369)	This function determines audio file by StarWave2 library supported is.
	StarBurn_StarWave2_IsFileSupportedUnicode (see page 370)	This function determines audio file by StarWave2 library supported is.
	StarBurn_strcpy_s (see page 370)	This function wrap runtime strcpy_s function. In Visual Studio 2003 function strcpy_s not present.
	StarBurn_swprintf_s (see page 371)	This function wrap runtime swprintf_s function. In Visual Studio 2003 oriinal swprintf_s not present.
	StarBurn_UDF_Add (see page 371)	Creates the UDF Tree Node
	StarBurn_UDF_AddUnicode (see page 372)	This is function StarBurn_UDF_AddUnicode.
	StarBurn_UDF_CleanUp (see page 372)	This function removes created UDF tree from the pointed UDF tree node sequentially. Processes file formatted UDF nodes and either closes file handles or frees memory (non-cached vs. cached content).
	StarBurn_UDF_Create (see page 374)	StarBurn_UDF_Create() function is obsolete. Use StarBurn_UDF_CreatEx() instead of it
	StarBurn_UDF_CreateEx (see page 374)	This function creates UDF tree from the allocated data buffers.
	StarBurn_UDF_CreateExUnicode (see page 377)	This is function StarBurn_UDF_CreateExUnicode.
	StarBurn_UDF_CreateUnicode (see page 377)	This is function StarBurn_UDF_CreateUnicode.
	StarBurn_UDF_Destroy (see page 377)	This function devastates created UDF tree from the pointed root recursively.
	StarBurn_UDF_DestroyNodeAndKids (see page 379)	Destroys UDF Tree Node and all its kids
	StarBurn_UDF_FormatTreeItemAsDirectory (see page 379)	This function formats UDF tree item as directory.
	StarBurn_UDF_FormatTreeItemAsDirectoryUnicode (see page 382)	This is function StarBurn_UDF_FormatTreeItemAsDirectoryUnicode.
	StarBurn_UDF_FormatTreeItemAsFile (see page 382)	This function formats UDF tree item as file.
	StarBurn_UDF_FormatTreeItemAsFileUnicode (see page 384)	This is function StarBurn_UDF_FormatTreeItemAsFileUnicode.

	StarBurn_UDF_GetFirstChild (see page 385)	This function returns the pointer to first child node
	StarBurn_UDF_GetNextSibling (see page 385)	This function returns the pointer to next sibling node
	StarBurn_UDF_GetNodeObject (see page 385)	Returns the object to current Tree Node
	StarBurn_UDF_GetParent (see page 386)	This function returns the pointer to next parent node
	StarBurn_UDF_GetPreviousSibling (see page 386)	This function returns the pointer to previous sibling node
	StarBurn_UDF2_DirectoryBrowse (see page 387)	This function browses UDF directory content.
	StarBurn_UDF2_DirectoryContentsProcess (see page 387)	This function processes contents of single UDF directory.
	StarBurn_UDF2_DirectoryContentsProcessEx (see page 388)	This function processes contents of single UDF directory.
	StarBurn_UDF2_DirectoryContentsUnicodeProcess (see page 388)	This function processes contents of single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryContentsUnicodeProcessEx (see page 389)	This function processes contents of single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryCreate (see page 390)	This function creates UDF directory.
	StarBurn_UDF2_DirectoryDestroy (see page 390)	This function destroys created UDF directory object.
	StarBurn_UDF2_DirectoryProcess (see page 390)	This function processes single UDF directory.
	StarBurn_UDF2_DirectoryProcessEx (see page 391)	This function processes single UDF directory.
	StarBurn_UDF2_DirectoryProcessExEx (see page 392)	This function processes single UDF directory.
	StarBurn_UDF2_DirectoryQuery (see page 392)	This function does query for UDF directory properties.
	StarBurn_UDF2_DirectoryRootCreate (see page 393)	This function creates root UDF directory.
	StarBurn_UDF2_DirectoryUnicodeCreate (see page 393)	This function creates UNICODE UDF directory.
	StarBurn_UDF2_DirectoryUnicodeProcess (see page 394)	This function processes single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryUnicodeProcessEx (see page 394)	This function processes single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryUnicodeProcessExEx (see page 395)	This function processes single UNICODE UDF directory.
	StarBurn_UDF2_FileBootCreate (see page 395)	This function creates UDF boot file.
	StarBurn_UDF2_FileBootUnicodeCreate (see page 396)	This function creates UNICODE UDF boot file.
	StarBurn_UDF2_FileCreate (see page 396)	This function creates UDF file.
	StarBurn_UDF2_FileDestroy (see page 397)	This function destroys created UDF file object.
	StarBurn_UDF2_FileDirectoryDateTimeGet (see page 397)	This function gets date and time from created UDF file or directory object.
	StarBurn_UDF2_FileDirectoryDateTimeGetEx (see page 398)	This function gets date and time from created UDF file or directory object. It may get creation, last access, or last write date and time.
	StarBurn_UDF2_FileDirectoryDateTimeSet (see page 398)	This function sets date and time on created UDF file or directory object.
	StarBurn_UDF2_FileDirectoryDateTimeSetEx (see page 399)	This function sets date and time on created UDF file or directory object. It may set creation, last access, or last write date and time.
	StarBurn_UDF2_FileDirectoryNameSet (see page 399)	This function set new name for UDF file or directory object.
	StarBurn_UDF2_FileDirectoryUnicodeNameSet (see page 400)	This function set new UNICODE name for UDF file or directory object.
	StarBurn_UDF2_FileMemoryCreate (see page 400)	This function creates UDF memory file.
	StarBurn_UDF2_FileMemoryUnicodeCreate (see page 401)	This function creates UNICODE UDF memory file.
	StarBurn_UDF2_FileQuery (see page 401)	This function do query for UDF file properties.
	StarBurn_UDF2_FileSetAttributes (see page 402)	This function sets attributes for UDF file.
	StarBurn_UDF2_FileSystemGetSizes (see page 402)	This function calculates UDF file system sizes in Bytes and LBS on the fly (without build volume).
	StarBurn_UDF2_FileUnicodeCreate (see page 403)	This function creates UNICODE UDF file.
	StarBurn_UDF2_ImpVolumeCreate (see page 403)	This function creates UDF import volume.
	StarBurn_UDF2_ImpVolumeDestroy (see page 404)	This function destroys created UDF import volume object.
	StarBurn_UDF2_ImpVolumeImport (see page 404)	This function imports UDF import volume content.
	StarBurn_UDF2_ISO9660FileTreeCreate (see page 404)	This function create ISO image from already created UDF volume.
	StarBurn_UDF2_ISO9660FileTreeCreateEx (see page 405)	This function create ISO image from already created UDF volume.
	StarBurn_UDF2_VolumeAnchorGet (see page 405)	This function gets first anchor volume descriptor pointer we'll have to replace on rewritable media.
	StarBurn_UDF2_VolumeCreate (see page 406)	This function creates UDF 1.02 volume.
	StarBurn_UDF2_VolumeCreateBoot (see page 406)	This function creates UDF Bootable volume.
	StarBurn_UDF2_VolumeCreateBootElTorito (see page 408)	This function creates UDF Bootable volume.
	StarBurn_UDF2_VolumeCreateEx (see page 409)	This function creates UDF volume.
	StarBurn_UDF2_VolumeCreateUnicodeBootElTorito (see page 410)	This function creates UDF Bootable volume.
	StarBurn_UDF2_VolumeDestroy (see page 411)	This function destroys created UDF volume object.

	StarBurn_UDF2_VolumeInitialSizeInLBsGet (see page 411)	This function get UDF volume size in unsigned chars.
	StarBurn_UDF2_VolumeSeekRead (see page 412)	This function seeks to passed offset and reads requested amount of data.
	StarBurn_UDF2_VolumeSizeInLBsGet (see page 412)	This function get UDF volume size in unsigned chars.
	StarBurn_UDF2_VolumeSizeInUCHARsGet (see page 413)	This function get UDF volume size in unsigned chars.
	StarBurn_UDF2_VolumeStore (see page 413)	This function stores UDF volume to file.
	StarBurn_UDF2_VolumeUnicodeCreate (see page 413)	This function creates UDF 1.02 UNICODE volume.
	StarBurn_UDF2_VolumeUnicodeCreateBoot (see page 414)	This function creates UDF Unicode Bootable volume.
	StarBurn_UDF2_VolumeUnicodeCreateEx (see page 415)	This function creates UDF UNICODE volume.
	StarBurn_UDFBridge_CreateEx (see page 416)	This is function StarBurn_UDFBridge_CreateEx.
	StarBurn_UDFBridge_CreateExUnicode (see page 417)	This is function StarBurn_UDFBridge_CreateExUnicode.
	StarBurn_UpStart (see page 417)	This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used. License file StarBurn.key is assumed to be located near main application executable. Since StarBurn SDK V12 it's possible to use this call again. It will just enable using embedded evaluational StarBurn SDK key (file shipped with StarBurn SDK... more (see page 417)
	StarBurn_UpStartEx (see page 418)	This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used. The difference between this call and call to StarBurn_UpStart (see page 417)(...) is that this particular call assumes StarBurn key is being embedded to the application binary and StarBurn_UpStart (see page 417)(...) always looks for StarBurn.key near application executable. Use this API... more (see page 418)
	StarBurn_UpStartExEx (see page 419)	This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Unlike StarBurn_UpStartEx (see page 418)(...) API call this one allows to use custom debug output control. See StarBurn_UpStartEx (see page 418)(...) API call for more information about this topic.

2.1.1 StarBurn_CdvdBurnerGrabber_AuthorizeDVD Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_AuthorizeDVD(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.

IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function authorizes DVD that has CSS protection system.

Remarks

Please see the GrabDisc sample to see how to grab DVD disc into MDS format and how StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem (see page 66) can be used to determine protection system of currently inserted DVD disc and how StarBurn_CdvdBurnerGrabber_AuthorizeDVD() can be used to authorize CSS protected DVD.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem (see page 66)

Example

This example allocates CdvdBurnerGrabber object, gets protection system, authorizes DVD disc and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__DVDProtectionSystem = DVD_PROTECTION_NONE;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
```

```
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get inserted disc type, supposed to be DVD...

// Try to get DVD protection system
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__DVDProtectionSystem
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to authorize the DVD disc (it's expected that DVD disc has CSS protection)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_AuthorizeDVD(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.2 StarBurn_CdvdBurnerGrabber_AuthorizeDVDEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_AuthorizeDVDEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__BusKeyForDiscKey
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__BusKeyForDiscKey	Pointer to bus key for disc key.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function authorizes DVD that has CSS protection system and keeps bus key for disc key.

Remarks

Please see the GrabDisc sample to see how to grab DVD disc into MDS format and how StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem (see page 66) can be used to determine protection system of currently inserted DVD disc and how StarBurn_CdvdBurnerGrabber_AuthorizeDVDEx() can be used to authorize CSS protected DVD.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem (see page 66)

Example

This example allocates CdvdBurnerGrabber object, gets protection system, authorizes DVD disc and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__DVDProtectionSystem = DVD_PROTECTION_NONE;
UCHAR l__UCHAR__BusKeyForDiscKey[ 5 ] = { 0, 0, 0, 0, 0 };

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
```

```
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get inserted disc type, supposed to be DVD...

// Try to get DVD protection system
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__DVDProtectionSystem
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to authorize the DVD disc (it's expected that DVD disc has CSS protection) and keep
bus key for disc key
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_AuthorizeDVDEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__BusKeyForDiscKey
    );

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.3 StarBurn_CdvdBurnerGrabber_Blank Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_Blank(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
```

```
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN ERASE_TYPE p__ERASE_TYPE
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN ERASE_TYPE p__ERASE_TYPE	Erase type (ERASE_TYPE_BLANK_DISC_FULL and ERASE_TYPE_BLANK_DISC_FAST supported for now).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function erases the rewritable media inserted into CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the Blank sample that will demonstrate how to erase the rewrittable disc.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), ERASE_TYPE (see page 486)

Example

This example allocates CdvdBurnerGrabber object, erases the media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
```

```
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to erase the media
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Blank(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ERASE_TYPE_BLANK_DISC_FAST
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.4 StarBurn_CdvdBurnerGrabber_BluRayFormat Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_BluRayFormat(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN ULONG p__ULONG__FormatProfileNumber
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN ULONG p__ULONG__FormatProfileNumber	Format profile number to use.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function formats Blu-Ray rewritable media (BD-RE) inserted into the Blu-Ray burner device.

Remarks

Please see the BluRayFormat sample that will demonstrate how to format rewrittable Blu-Ray disc (BD-RE).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_BluRayFormatQuery (see page 25), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, formats the BD-RE media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber = NULL;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError = ERROR_SUCCESS;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
ULONG l__ULONG__FormatProfileNumber = 0;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
```

```
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Check for the BD-RE media inserted

// Get possible format profiles and pick up format profile number

// Try to format the Blu-Ray rewrittable media (BD-RE)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_BluRayFormat(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__ULONG__FormatProfileNumber
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.5 StarBurn_CdvdBurnerGrabber_BluRayFormatQuery Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_BluRayFormatQuery(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__FormatProfileBuffer,
    OUT PULONG p__PULONG__FormatProfileBufferSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PVOID p__PVOID__CdvdBurnerGrabber`	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.

OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__FormatProfileBuffer	Pointer to the profile format buffer.
p__PULONG__FormatProfileBufferSizeInUCHARS	Pointer to the format profile buffer size in UCHAR(s).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes). If the passed buffer size is too small StarBurn will set p__PULONG__FormatProfileBufferSizeInUCHARs to the required buffer size in UCHAR(s). After completion p__PULONG__FormatProfileBufferSizeInUCHARs will be set to the number of UCHAR(s) copied.

Description

This function queries available Blu-Ray formats for the currently inserted Blu-Ray rewritable media (BD-RE).

Remarks

Please see the BluRayFormat sample that will demonstrate how to format rewrittable Blu-Ray media (BD-RE) and query all available format profiles.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_BluRayFormat (see page 23), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, queries available Blu-Ray formats and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber = NULL;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError = ERROR_SUCCESS;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__FormatProfileBuffer[ PAGE_SIZE_IN_UCHARS ];
ULONG l__ULONG__FormatProfileBufferSizeInUCHARs = sizeof( l__UCHAR__FormatProfileBuffer );

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
```

```
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Check for the BD-RE media inserted

// Query available Blu-Ray formats
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_BluRayFormatQuery(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__FormatProfileBuffer,
    &l__ULONG__FormatProfileBufferSizeInUCHARs
    );

// Try to format the Blu-Ray rewrittable media

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.6 StarBurn_CdvdBurnerGrabber_Cancel Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CdvdBurnerGrabber_Cancel(
    IN PVOID p__PVOID__CdvdBurnerGrabber
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PVOID p__PVOID__CdvdBurnerGrabber`	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)().

Returns

Nothing. This function cannot fail.

Description

This function cancels current read or write process on the CD/DVD/Blu-Ray/HD-DVD burner.

Remarks

There is no sample provided for this code so please check the code below.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_VideoCD (see page 203), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157), StarBurn_CdvdBurnerGrabber_GrabTrack (see page 110), StarBurn_CdvdBurnerGrabber_GrabCD (see page 98), StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)

Example

This example allocates CdvdBurnerGrabber object, cancels current I/O process and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Initiate some I/O process here

// Cancel initiated I/O process
StarBurn_CdvdBurnerGrabber_Cancel();

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );
```

```
// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.7 StarBurn_CdvdBurnerGrabber_CloseSession Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_CloseSession(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function closes session on on CD/DVD/Blu-Ray/HD-DVD burner device after something was written using StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184) or StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172).

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burnon the CD/DVD/Blu-Ray/HD-DVD media and with the help of the StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172) or StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184) and how session can be closed with the help of StarBurn_CdvdBurnerGrabber_CloseSession.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139),

StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image to the CD/DVD/Blu-Ray/HD-DVD media, closes session and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the ISO9660 or Joliet file system image in Track-At-Once mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "C:\ISO9660.ISO",
    FALSE,
    FALSE,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
```

```
}

// Try to close session
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_CloseSession(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.8 StarBurn_CdvdBurnerGrabber_Create Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_Create(
    OUT PVOID * p__PPVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCALLBACK p__PCALLBACK,
    IN PVOID p__PVOID__Context,
    IN UCHAR p__UCHAR__PortID,
    IN UCHAR p__UCHAR__BusID,
    IN UCHAR p__UCHAR__TargetID,
    IN UCHAR p__UCHAR__LUN,
    IN LONG p__LONG__CacheSizeInMBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID * p__PPVOID__CdvdBurnerGrabber	Pointer to pointer to the object that toolkit will set to the CdvdBurnerGrabber object it will allocate.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCALLBACK p__PCALLBACK	Callback that will be called to indicate progress of various actions.
IN PVOID p__PVOID__Context	Context value that will be passed to callback function.
IN UCHAR p__UCHAR__PortID	SCSI port ID device is located at.
IN UCHAR p__UCHAR__BusID	SCSI bus ID device is located at.
IN UCHAR p__UCHAR__TargetID	SCSI bus target device is located at.

IN UCHAR p__UCHAR__LUN	SCSI LUN device is located at.
IN LONG p__LONG__CacheSizeInMBs	Cache size to use during I/O operations, 0 is a special value that tells the toolkit to allocate default cache size.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function creates CD/DVD/Blu-Ray/HD-DVD burner device object. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD releated actions.

Remarks

Please take a look at FindDevice sample to find out how StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) could be used to create CD/DVD/Blu-Ray/HD-DVD burner object. Please note that StarBurn_CdvdBurnerGrabber_Create would create ASPI device thus would work not only under Windows NT/2000/XP/2000 but also under Windows 95/98/Me. ASPI is natively supported under Windows 95/98/Me and we provide own ASPI-to-SPTI wrapper for StarBurn). If you don't need Windows 95/98/Me support it's a good idea to use new extended variant of the call StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) as it creates device handle using SPTI transport.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
```

```
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.9 StarBurn_CdvdBurnerGrabber_CreateEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_CreateEx(
    OUT PVOID * p__PPVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCALLBACK p__PCALLBACK,
    IN PVOID p__PVOID__Context,
    IN PCHAR p__PCHAR__DeviceName,
    IN LONG p__LONG__CacheSizeInMBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID * p__PPVOID__CdvdBurnerGrabber	Pointer to pointer to the object that toolkit will set to the CdvdBurnerGrabber object it will allocate.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCALLBACK p__PCALLBACK	Callback that will be called to indicate progress of various actions.
IN PVOID p__PVOID__Context	Context value that will be passed to callback function.
IN PCHAR p__PCHAR__DeviceName	Pointer to device symbolic link name we'll use. Make sure it DOES NOT include any slashes. So the name should be either "CdRom0" or say "F:" but NOT "\.CdRom0" or "\.F:".
IN LONG p__LONG__CacheSizeInMBs	Cache size to use during I/O operations, 0 is a special value that tells the toolkit to allocate default cache size.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function creates CD/DVD/Blu-Ray/HD-DVD burner device object using SPTI (SCSI Pass Through Interface) transport. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD related actions.

Remarks

Please take a look at FindDeviceEx sample to find out how StarBurn_CdvdBurnerGrabber_CreateEx could be used to create CD/DVD/Blu-Ray/HD-DVD burner object. Please note that StarBurn_CdvdBurnerGrabber_CreateEx would create SPTI device thus would work only under Windows NT/2000/XP/2003. If you need Windows 95/98/Me compatibility please use StarBurn_CdvdBurnerGrabber_Create (see page 31) legacy call as it's for ASPI transport (ASPI is natively supported under Windows 95/98/Me and we provide own ASPI-to-SPTI wrapper for StarBurn).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_CreateExEx (see page 35), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber using name "D:" with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_CreateEx(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    "D:",
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
```

```
}
```

2.1.10 StarBurn_CdvdBurnerGrabber_CreateExEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_CreateExEx(
    OUT PVOID * p__PPVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCALLBACK p__PCALLBACK,
    IN PVOID p__PVOID__Context,
    IN PWCHAR p__PWCHAR__DeviceName,
    IN BOOLEAN p__BOOLEAN__IsExclusiveAccess,
    IN LONG p__LONG__CacheSizeInMBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID * p__PPVOID__CdvdBurnerGrabber	Pointer to pointer to the object that toolkit will set to the CdvdBurnerGrabber object it will allocate.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCALLBACK p__PCALLBACK	Callback that will be called to indicate progress of various actions.
IN PVOID p__PVOID__Context	Context value that will be passed to callback function.
IN PWCHAR p__PWCHAR__DeviceName	Pointer to device symbolic link name we'll use.
IN BOOLEAN p__BOOLEAN__IsExclusiveAccess	Is device open with exclusive access
IN LONG p__LONG__CacheSizeInMBs	Cache size to use during I/O operations, 0 is a special value that tells the toolkit to allocate default cache size.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function creates CD/DVD/Blu-Ray/HD-DVD burner device object using SPTD (SCSI Pass Through Direct) transport. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD related actions.

Remarks

Please take a look at FindDeviceExEx sample to find out how StarBurn_CdvdBurnerGrabber_CreateExEx could be used to create CD/DVD/Blu-Ray/HD-DVD burner object. Please note that StarBurn_CdvdBurnerGrabber_CreateExEx would create SPTD device thus would work only under Windows NT/2000/XP/2003. If you need Windows 95/98/Me compatibility please use StarBurn_CdvdBurnerGrabber_Create (see page 31) legacy call as it's for ASPI transport (ASPI is natively supported under Windows 95/98/Me and we provide own ASPI-to-SPTI wrapper for StarBurn). ATTENTION! SPTD applications must be digitally signed. That's why only dynamic linking with the StarBurn.DLL is possible if SPTD must be used (unless you'll

directly license SPTD layer from Duplex Secure and will get signer for your EXE from them).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber using name "CdRom0" with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_CreateExEx(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    L"CdRom0",
    TRUE,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.11 StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN BOOLEAN p__BOOLEAN__IsSubChannelRepairRequired,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT	Pointer to preinitialized DAO disc layout.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE if next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN BOOLEAN p__BOOLEAN__IsSubChannelRepairRequired	BOOLEAN set to TRUE if you want library to repair broken sub-channel in the image and FALSE otherwise (if you want to burn all the data AS IS -- maybe required to keep working "special" software titles).
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be one sound file or full set of sound files if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once PQ mode.

Remarks

Please see the DiscAtOnceFromFile sample that will demonstrate how ISO9660 or Joliet file system image (also it can be one sound file or full set of sound files if an audio CD is mastered) can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile (see page 42)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
DAO_DISC_LAYOUT l__DAO_DISC_LAYOUT;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare DAO disc layout
RtlZeroMemory(
    &l__DAO_DISC_LAYOUT,
    sizeof( l__DAO_DISC_LAYOUT )
    );
```

```
// Set file name
strcpy(
    l__DAO_DISC_LAYOUT.m__DAO_DISC_LAYOUT_ENTRY.m__CHAR__TrackName,
    "C:\ISO9660.ISO"
    );

// Set number of tracks
l__DAO_DISC_LAYOUT.m__LONG__NumberOfTracks = 1;

// Try to record the ISO9660 or Joliet file system image in Disc-At-Once PQ mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__DAO_DISC_LAYOUT,
    FALSE, // No XA, just ordinary CDROM
    FALSE, // No test write, real burn
    FALSE, // No next session allowed
    FALSE, // No sub-channel repair
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.12
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN BOOLEAN p__BOOLEAN__IsSubChannelRepairRequired,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileUnicode.

2.1.13 StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PVOID p__PVOID__ISO9660FileTree,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE if next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree object.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in file tree object with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once PQ mode.

Remarks

Please see the DiscAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image located in a file tree object can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile (see page 42)

Example

This example allocates CdvdBurnerGrabber object, creates ISO9660 or Joliet file system image located in a file tree object and burns it to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
DAO_DISC_LAYOUT l__DAO_DISC_LAYOUT;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Create and initialize ISO9660 or Joliet file tree object here pointed by
```

```
l__PVOID__ISO9660JolietFileTree

// Try to record the ISO9660 or Joliet file system image in Disc-At-Once PQ mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__PVOID__ISO9660JolietFileTree,
    FALSE, // No XA, just ordinary CDROM
    FALSE, // No test write, real burn
    FALSE, // No next session allowed
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.14
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN BOOLEAN p__BOOLEAN__IsSubChannelRepairRequired,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT	Pointer to preinitialized DAO disc layout.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media. p__BOOLEAN__IsSubChannelRepairRequired -- BOOLEAN set to TRUE if you want library to repair broken sub-channel in the image and FALSE otherwise (if you want to burn all the data AS IS -- maybe required to keep working "special" software titles).
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be one sound file or full set of sound files if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once raw P-W mode.

Remarks

Please see the DiscAtOnceFromFile sample that will demonstrate how ISO9660 or Joliet file system image (also it can be one sound file or full set of sound files if an audio CD is mastered) can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile (see page 37)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
DAO_DISC_LAYOUT l__DAO_DISC_LAYOUT;
```

```
// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare DAO disc layout
RtlZeroMemory(
    &l__DAO_DISC_LAYOUT,
    sizeof( l__DAO_DISC_LAYOUT )
    );

// Set file name
strcpy(
    l__DAO_DISC_LAYOUT.m__DAO_DISC_LAYOUT_ENTRY.m__CHAR__TrackName,
    "C:\ISO9660.ISO"
    );

// Set number of tracks
l__DAO_DISC_LAYOUT.m__LONG__NumberOfTracks = 1;

// Try to record the ISO9660 or Joliet file system image in Disc-At-Once raw P-W mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__DAO_DISC_LAYOUT,
    FALSE, // No XA, just ordinary CDROM
    FALSE, // No test write, real burn
    FALSE, // No next session allowed
    FALSE, // No sub-channel repair
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

```
// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.15
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileAudioUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileAudioUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN BOOLEAN p__BOOLEAN__IsSubChannelRepairRequired,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to CStarBurn_CdvdBurnerGrabber object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB failure information.
IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT	Pointer to DAO disc layout.
IN BOOLEAN p__BOOLEAN__IsXA	Is this CDROM XA or just ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	Is this test write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	Is next session allowed.
IN BOOLEAN p__BOOLEAN__IsSubChannelRepairRequired	Is subchannel repair required.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Exception number

Description

This API accepts track names in Unicode format and does AUDIO only DAO burn.

2.1.16
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PDAO_DISC_LAYOUT p__PDAO_DISC_LAYOUT,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN BOOLEAN p__BOOLEAN__IsSubChannelRepairRequired,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileUnicode.

2.1.17
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PVOID p__PVOID__ISO9660FileTree,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE if next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree object.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in file tree object with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once raw P-W mode.

Remarks

Please see the DiscAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image located in a file tree object can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile (see page 42)

Example

This example allocates CdvdBurnerGrabber object, creates ISO9660 or Joliet file system image located in a file tree object and burns it to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
DAO_DISC_LAYOUT l__DAO_DISC_LAYOUT;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
```

```
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Create and initialize ISO9660 or Joliet file tree object here pointed by
l__PVOID__ISO9660JolietFileTree

// Try to record the ISO9660 or Joliet file system image in Disc-At-Once raw P-W mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__PVOID__ISO9660JolietFileTree,
    FALSE, // No XA, just ordinary CDROM
    FALSE, // No test write, real burn
    FALSE, // No next session allowed
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.18 StarBurn_CdvdBurnerGrabber_Eject Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_Eject(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function ejects the media on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_Eject can be used to eject the CD/DVD/Blu-Ray/HD-DVD media to initiate file system unmount/mount sequence (to refresh contents of just burn CD/DVD/Blu-Ray/HD-DVD media).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, ejects the media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
```

```
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to eject the media
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Eject(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.19 StarBurn_CdvdBurnerGrabber_ExecuteGeneric Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_ExecuteGeneric(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
```

```
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PUCHAR p__PUCHAR__CDB,
    IN ULONG p__ULONG__CDBSizeInUCHARs,
    IN OUT PUCHAR p__PUCHAR__DataBuffer,
    IN ULONG p__ULONG__DataBufferSizeInUCHARs,
    IN BOOLEAN p__BOOLEAN__IsDirectionToScsiTarget
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PUCHAR p__PUCHAR__CDB	Pointer to CDB we want to execute.
IN ULONG p__ULONG__CDBSizeInUCHARs	CDB size in UCHARs.
IN OUT PUCHAR p__PUCHAR__DataBuffer	Pointer to I/O data buffer.
IN ULONG p__ULONG__DataBufferSizeInUCHARs	I/O data buffer size in UCHARs.
IN BOOLEAN p__BOOLEAN__IsDirectionToScsiTarget	Is data moved from memory-to-target (TRUE) or from target-to-memory (FALSE).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function executes custom CDB (Command Descriptor Block) on CD/DVD/Blu-Ray/HD-DVD burner device object. This could be used to deal with the propriatry hardware or execute some special code StarBurn does not support out-of-box.

Remarks

Please see ExecuteGeneric sample on how to send custom SCSI commands to CD/DVD/Blu-Ray/HD-DVD burner device.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_ImportTrack

Example

This example allocates CdvdBurnerGrabber object, sends custom command and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__CDB[ 12 ];
UCHAR l__UCHAR__Data[ 0x2000 ];

// Prepare exception text buffer
```

```
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Format CDB and data buffer here...

// Try to send custom CDB to CD/DVD/Blu-Ray/HD-DVD burner device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_ExecuteGeneric(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__CDB,
    sizeof( l__UCHAR__CDB ),
    &l__UCHAR__Data,
    sizeof( l__UCHAR__Data ),
    TRUE
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.20 StarBurn_CdvdBurnerGrabber_GetAdvancedSupportedMediaFormats Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_GetAdvancedSupportedMediaFormats(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    IN PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS
p__PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)() or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)() API calls.
IN PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS p__PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS	Pointer to pre-allocated and pre-initialized structure containing set of capabilities flags.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function returns CD/DVD/Blu-Ray/HD-DVD burner device object extended information. This data can be used to report the capabilities of the device to the user.

Remarks

Please see the TrackAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what use can be taken from CdvdBurnerGrabber device extended information.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDeviceInformation (see page 57), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsEx (see page 88), StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats (see page 85), StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsExEx (see page 90)

Example

This example allocates CdvdBurnerGrabber object, retrieves device extended information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
```

```
ULONG l__ULONG__SystemError;
ULONG l__ULONG__CacheBufferSizeInUCHARs;
CHAR l__CHAR__ExceptionText[ 1024 ];
STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS l__STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Get CdvdBurnerGrabber device information here...

// Prepare advanced supported media formats
RtlZeroMemory(
    &l__STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS,
    sizeof( l__STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS )
    );

// Set structure size in UCHARs
l__STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS.m__ULONG__SizeInUCHARs = sizeof(
l__STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS );

// Get CdvdBurnerGrabber device extended information
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetAdvancedSupportedMediaFormats(
    l__PVOID__CdvdBurnerGrabber,
    &l__STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS
    );

// Do something with CdvdBurnerGrabber device object and it's extended information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.21 StarBurn_CdvdBurnerGrabber_GetBUP Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetBUP(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PBOOLEAN p__PBOOLEAN__IsBUPEnabled,
    OUT PBOOLEAN p__PBOOLEAN__IsBUPSupported
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PBOOLEAN p__PBOOLEAN__IsBUPEnabled	Pointer to the BOOLEAN variable that will be set to TRUE if BUP is currently enabled on the CD/DVD/Blu-Ray/HD-DVD device, set to FALSE if BUP is disabled.
OUT PBOOLEAN p__PBOOLEAN__IsBUPSupported	Pointer to the BOOLEAN variable that will be set to TRUE if BUP is generally supported by the CD/DVD/Blu-Ray/HD-DVD device, set to FALSE if BUP is not supported.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function tests is CD/DVD/Blu-Ray/HD-DVD burner device supports BUP (Buffer Underrun Protection) or not and what is BUP current status (if supported). This information can be used later to set BUP status before starting write operations.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what StarBurn_CdvdBurnerGrabber_GetBUP can be used for.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, gets BUP status and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
```

```
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
BOOLEAN l__BOOLEAN__IsBUPEnabled;
BOOLEAN l__BOOLEAN__IsBUPSupported;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get BUP on the device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetBUP(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__BOOLEAN__IsBUPEnabled,
    &l__BOOLEAN__IsBUPSupported
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.22 StarBurn_CdvdBurnerGrabber_GetDeviceInformation Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CdvdBurnerGrabber_GetDeviceInformation(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__VendorID,
    OUT PCHAR p__PCHAR__ProductID,
    OUT PCHAR p__PCHAR__ProductRevisionLevel,
    OUT PULONG p__PULONG__BufferSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)().
OUT PCHAR p__PCHAR__VendorID	Pointer to array of CHARs that will be used to store vendor ID string.
OUT PCHAR p__PCHAR__ProductID	Pointer to array of CHARs that will be used to store product ID string.
OUT PCHAR p__PCHAR__ProductRevisionLevel	Pointer to array of CHARs that will be used to store product revision level string.
OUT PULONG p__PULONG__BufferSizeInUCHARs	Pointer to ULONG that will contain the device internal cache buffer size in UCHARs.

Returns

None. This function cannot fail.

Description

This function returns CD/DVD/Blu-Ray/HD-DVD burner device object information. This data can be used to report the capabilities of the device to the user.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what use can be taken from CdvdBurnerGrabber device information.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, retrieves device information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
ULONG l__ULONG__CacheBufferSizeInUCHARs;
CHAR l__CHAR__ExceptionText[ 1024 ];
CHAR l__CHAR__VendorID[ 1024 ];
CHAR l__CHAR__ProductID[ 1024 ];
CHAR l__CHAR__ProductRevisionLevel[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
```

```
// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare vendor ID and product ID and product revision level buffers

RtlZeroMemory(
    &l__CHAR__VendorID,
    sizeof( l__CHAR__VendorID )
    );

RtlZeroMemory(
    &l__CHAR__ProductID,
    sizeof( l__CHAR__ProductID )
    );

RtlZeroMemory(
    &l__CHAR__ProductRevisionLevel,
    sizeof( l__CHAR__ProductRevisionLevel )
    );

// Get CdvdBurnerGrabber device information
StarBurn_CdvdBurnerGrabber_GetDeviceInformation(
    l__PVOID__CdvdBurnerGrabber,
    ( PCHAR )( &l__CHAR__VendorID ),
    ( PCHAR )( &l__CHAR__ProductID ),
    ( PCHAR )( &l__CHAR__ProductRevisionLevel ),
    &l__ULONG__CacheBufferSizeInUCHARs
    );

// Do something with CdvdBurnerGrabber device object and it's information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.23 StarBurn_CdvdBurnerGrabber_GetDeviceInformationUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CdvdBurnerGrabber_GetDeviceInformationUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PWCHAR p__PCHAR__VendorID,
    OUT PWCHAR p__PCHAR__ProductID,
    OUT PWCHAR p__PCHAR__ProductRevisionLevel,
    OUT PULONG p__PULONG__BufferSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_GetDeviceInformationUnicode.

2.1.24 StarBurn_CdvdBurnerGrabber_GetDiscFileSystem Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetDiscFileSystem(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PDISC_FILESYSTEM p__PDISC_FILESYSTEM
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PDISC_FILESYSTEM p__PDISC_FILESYSTEM	Pointer to structure to receive disc file system flags.

Returns

Execution status.

Description

This function detects recorded optical disc file system.

Example

Please see GUI DataBurner sample as example how to use StarBurn_GetDiscFileSystem API call.

2.1.25 StarBurn_CdvdBurnerGrabber_GetDiscFreeSpace Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetDiscFreeSpace(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    IN STARBURN_CD_MODE p__CD_MODE,
    OUT PLONG p__PLONG__FreeSpaceInLBs,
    OUT PLARGE_INTEGER p__PLARGE_INTEGER__FreeSpaceInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
IN STARBURN_CD_MODE p__CD_MODE	CD mode. Ignored if currently inserted disc type is DVD.
OUT PLONG p__PLONG__FreeSpaceInLBs	Pointer to the LONG will be filled with number of free logical blocks on the disc.
p__PLARGE_INTEGER__FreeSpaceInUCHARS	Pointer to the LARGE_INTEGER will be filled with free space in UCHARs on the disc.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets disc free space of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

If currently inserted disc type is DVD then parameter p__CD_MODE is ignored.For DVD disc type default LB size is 2048 UCHARs. If p__CD_MODE set to CD_MODE_UNKNOWN or CD_MODE_RESERVED function return p__PLARGE_INTEGER__FreeSpaceInUCHARS is zero.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDiscInformation (see page 62), StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94), StarBurn_CdvdBurnerGrabber_GetTOCInformation (see page 92), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), STARBURN_DISC_INFORMATION (see page 555), STARBURN_TRACK_INFORMATION (see page 563), STARBURN_TOC_INFORMATION

Example

This example allocates CdvdBurnerGrabber object, gets free disc space and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
```

```
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
CD_MODE l__CD_MODE(CD_MODE_DATA);
LONG l__LONG__FreeSpaceInLBs(0);
LARGE_INTEGER l__LARGE_INTEGER__FreeSpaceInUCHARs;

l__LARGE_INTEGER__FreeSpaceInUCHARs.QuadPart = 0;

// Prepare exception text buffer
RtlZeroMemory(
&l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText )
);

// Prepare CDB failure information
RtlZeroMemory(
&l__CDB_FAILURE_INFORMATION,
sizeof( l__CDB_FAILURE_INFORMATION )
);

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
&l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
( PCALLBACK )( StarBurn_Callback ),
0,
0,
4,
0,
32
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get current disc free space
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetDiscFreeSpace(
l__PVOID__CdvdBurnerGrabber,
l__CD_MODE,
&l__LONG__FreeSpaceInLBs,
&l__LARGE_INTEGER__FreeSpaceInUCHARs
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.26 StarBurn_CdvdBurnerGrabber_GetDiscInformation Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetDiscInformation(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PSTARBURN_DISC_INFORMATION p__PSTARBURN_DISC_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PSTARBURN_DISC_INFORMATION p__PSTARBURN_DISC_INFORMATION	Pointer to the disc information.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets disc information of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the TrackAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetDiscInformation can be used to get current disc information on the CD/DVD/Blu-Ray/HD-DVD media to check parameters of the disc for validness (is disc blank, last session status etc).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), STARBURN_DISC_INFORMATION (see page 555), STARBURN_TRACK_INFORMATION (see page 563)

Example

This example allocates CdvdBurnerGrabber object, gets disc information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
```

```
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
STARBURN_DISC_INFORMATION l__STARBURN_DISC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare disc information buffer
RtlZeroMemory(
    &l__STARBURN_DISC_INFORMATION,
    sizeof( l__STARBURN_DISC_INFORMATION )
    );

// Try to get current disc information
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetDiscInformation(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PSTARBURN_DISC_INFORMATION )( &l__STARBURN_DISC_INFORMATION )
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
```

```
}
```

2.1.27 StarBurn_CdvdBurnerGrabber_GetDiscUsedSpace Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetDiscUsedSpace(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PLARGE_INTEGER p__PLARGE_INTEGER__UsedSpace
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PLARGE_INTEGER p__PLARGE_INTEGER__UsedSpace	Pointer to the used space of disc.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets disc used space of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDiscInformation (see page 62), StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94), StarBurn_CdvdBurnerGrabber_GetTOCInformation (see page 92), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), STARBURN_DISC_INFORMATION (see page 555), STARBURN_TRACK_INFORMATION (see page 563), STARBURN_TOC_INFORMATION

Example

This example allocates CdvdBurnerGrabber object, gets used disc space and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
```

```
LARGE_INTEGER l__LARGE_INTEGER__UsedSpace;

l__LARGE_INTEGER__UsedSpace.QuadPart = 0;

// Prepare exception text buffer
RtlZeroMemory(
&l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText )
);

// Prepare CDB failure information
RtlZeroMemory(
&l__CDB_FAILURE_INFORMATION,
sizeof( l__CDB_FAILURE_INFORMATION )
);

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
&l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
( PCALLBACK )( StarBurn_Callback ),
0,
0,
4,
0,
32
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}


// Try to get current disc used space
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetDiscUsedSpace(
l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
&l__LARGE_INTEGER__UsedSpace
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.28 StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__DVDProtectionSystem
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__DVDProtectionSystem	Pointer to UCHAR that will keep DVD protection system in case of success.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets DVD protection system of inserted DVD media into DVD device.

Remarks

Please see the GrabDisc sample to see how to grab DVD disc into MDS format and how StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem can be used to determine protection system of currently inserted DVD disc.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem

Example

This example allocates CdvdBurnerGrabber object, gets protection system and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
```

```
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__DVDProtectionSystem = DVD_PROTECTION_NONE;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get inserted disc type, supposed to be DVD...

// Try to get DVD protection system
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__DVDProtectionSystem
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.29 StarBurn_CdvdBurnerGrabber_GetDVDRegionMask Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetDVDRegionMask(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__DVDRegionMask
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__DVDRegionMask	Pointer to the variable to receive DVD region mask.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets DVD region mask from DVD media inserted to CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.

Remarks

Please see the GrabDisc sample that will demonstrate how DVD region mask could be get from DVD media with the help of the StarBurn_CdvdBurnerGrabber_GetDVDRegionMask API call.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33), StarBurn_CdvdBurnerGrabber_GetRPC (see page 81), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, gets DVD region and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
```

```
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__DVDRegionMask;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get DVD region mask from DVD media inserted to device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetDVDRegionMask(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__DVDRegionMask
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.30 StarBurn_CdvdBurnerGrabber_GetInsertedDiscType Function

C++

```
__stdcall STARBURN_IMPEX_API DISC_TYPE StarBurn_CdvdBurnerGrabber_GetInsertedDiscType(
    IN PVOID p__PVOID__CdvdBurnerGrabber
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)(...) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)(...) API calls.

Returns

Inserted disc type. This function cannot fail.

Description

This function returns inserted to CD/DVD/Blu-Ray/HD-DVD burner device disc type. This data can be used to report it to the user and to select the stream type to record.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what use can be taken from inserted disc type information.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, retrieves inserted disc type and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
DISC_TYPE l__DISC_TYPE = DISC_TYPE_UNKNOWN;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
```

```
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Get inserted disc type
l__DISC_TYPE = StarBurn_CdvdBurnerGrabber_GetInsertedDiscType( l__PVOID__CdvdBurnerGrabber
);

// Do something with CdvdBurnerGrabber device object and it's inserted disc type here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.31 StarBurn_CdvdBurnerGrabber_GetLastAddress Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetLastAddress(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PULONG p__PULONG__LBA
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PULONG p__PULONG__LBA	Pointer to the variable to receive last recorded address.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets last recorded address from CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object. Such an address could be used for already recorded session import with the UDF mastering.

Remarks

Please see UDF2ImportBurn sample that will demonstrate how UDF file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetLastAddress can be used to get last recorded address.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_GetLastTrack (see page 73)

Example

This example allocates CdvdBurnerGrabber object, gets last recorded address and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
ULONG l__ULONG__LastRecordedLBA = 0;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );
```

```
// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get last recorded address
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetLastAddress(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__ULONG__LastRecordedLBA
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.32 StarBurn_CdvdBurnerGrabber_GetLastTrack Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetLastTrack(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__LastTrack
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__LastTrack	Pointer to the variable to receive last track number.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be

EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets last recorded track from CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object. Such a track number could be used for already recorded session import.

Remarks

Please see TrackAtOnceFromTreeWithImport sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetLastTrack can be used to get last recorded track number.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_ImportTrack

Example

This example allocates CdvdBurnerGrabber object, gets last recorded track number and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__LastRecordedTrackNumber;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

```
// Try to get last recorded track number
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetLastTrack(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__LastRecordedTrackNumber
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.33 StarBurn_CdvdBurnerGrabber_GetMechanicalOptions Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_GetMechanicalOptions(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PBOOLEAN p__PBOOLEAN__IsLockUnLockSupported,
    OUT PBOOLEAN p__PBOOLEAN__IsLocked,
    OUT PBOOLEAN p__PBOOLEAN__IsLoadEjectSupported,
    OUT PUCHAR p__PUCHAR__LoadingMechanismType
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PBOOLEAN p__PBOOLEAN__IsLockUnLockSupported	Pointer to the variable to receive is device capable of programmatically lock/unlock.
OUT PBOOLEAN p__PBOOLEAN__IsLocked	Pointer to the variable to receive is media inside device locked right now.
OUT PBOOLEAN p__PBOOLEAN__IsLoadEjectSupported	Pointer to the variable to receive is device capable of programmatically load/eject.

OUT PUCHAR p__PUCHAR__LoadingMechanismType	Pointer to the variable to receive device loading mechanism type.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets mechanical options (is device capable of programmatically load/eject and lock/unlock etc) from CD/DVD/Blu-Ray/HD-DVD burner device object.

Remarks

Please see the TrackAtOnceFromFile sample that will demonstrate how ISO9660 or Joliet file system image can be burn to the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetMechanicalOptions() can be used to determine device media control capabilities.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166), StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see page 168), StarBurn_CdvdBurnerGrabber_GetMediaTrayStatus (see page 77)

Example

This example allocates CdvdBurnerGrabber object, gets mechanical options and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
BOOLEAN l__BOOLEAN__IsLockUnLockSupported = FALSE;
BOOLEAN l__BOOLEAN__IsLocked = FALSE;
BOOLEAN l__BOOLEAN__IsLoadEjectSupported = FALSE;
UCHAR l__UCHAR__LoadingMechanismType = 0x00;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
```

```
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get media and tray statuses
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetMechanicalOptions(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__BOOLEAN__IsLockUnLockSupported,
    &l__BOOLEAN__IsLocked,
    &l__BOOLEAN__IsLoadEjectSupported,
    &l__UCHAR__LoadingMechanismType
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.34 StarBurn_CdvdBurnerGrabber_GetMediaTrayStatus Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetMediaTrayStatus(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PBOOLEAN p__PBOOLEAN__IsMediaPresent,
    OUT PBOOLEAN p__PBOOLEAN__IsDoorOrTrayOpen
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.

OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PBOOLEAN p__PBOOLEAN__IsMediaPresent	Is media inserted (TRUE) or not (FALSE).
OUT PBOOLEAN p__PBOOLEAN__IsDoorOrTrayOpen	Is tray or door opened (TRUE) or closed (FALSE).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets media (inserted or not) and tray (opened or closed) statuses from CD/DVD/Blu-Ray/HD-DVD burner device object.

Remarks

Please see the TrackAtOnceFromFile sample that will demonstrate how ISO9660 or Joliet file system image can be burn to the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetMediaTrayStatus can be used to determine media and tray status.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166), StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see page 168)

Example

This example allocates CdvdBurnerGrabber object, gets media and tray statuses and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
BOOLEAN l__BOOLEAN__IsMediaPresent;
BOOLEAN l__BOOLEAN__IsDoorOrTrayOpen;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
```

```
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get media and tray statuses
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetMediaTrayStatus(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__BOOLEAN__IsMediaPresent,
    &l__BOOLEAN__IsDoorOrTrayOpen
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.35 StarBurn_CdvdBurnerGrabber_GetNumberOfSystemDescriptors Function

C++

```
__stdcall STARBURN_IMPEX_API LONG StarBurn_CdvdBurnerGrabber_GetNumberOfSystemDescriptors(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    IN PUCHAR p__PUCHAR__SystemStructures,
    IN LONG p__LONG__SystemStructuresSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
IN PUCHAR p__PUCHAR__SystemStructures	Pointer to system structures pre-read into memory buffer.
IN LONG p__LONG__SystemStructuresSizeInLBs	System structures size in logical blocks.

Returns

Number of system descriptors. This code cannot fail.

Description

This function gets number of system descriptors in system structures stream. This value is used for updating head of the image when creating signle track session import code.

Remarks

Please see BuildImageWithImportFromFile sample that will demonstrate how ISO9660 or Joliet file system image can be imported from hard disk ISO disk image and how StarBurn_CdvdBurnerGrabber_GetNumberOfSystemDescriptors(...) can be used to get number of system descriptors when updating system image head.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31)

Example

This example allocates CdvdBurnerGrabber object, gets number of system descriptors and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber = NULL;
LONG l__LONG__NumberOfSystemDescriptors = 0;
PUCHAR l__PUCHAR__SystemStructures = 0;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Allocate enough memory for system structures in l__PUCHAR__SystemStructures here...

// Read system structures from the optical disc to l__PUCHAR__SystemStructures buffer
here...

// Try to get number of system descriptors
l__LONG__NumberOfSystemDescriptors =
StarBurn_CdvdBurnerGrabber_GetNumberOfSystemDescriptors(
```

```
    l__PVOID__CdvdBurnerGrabber,
    l__PUCHAR__SystemStructures,
    SYSTEM_STRUCTURES_SIZE_IN_LOGICAL_BLOCKS
    );

// Do something with l__LONG__NumberOfSystemDescriptors here...

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.36 StarBurn_CdvdBurnerGrabber_GetRPC Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetRPC(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__TypeCode,
    OUT PUCHAR p__PUCHAR__NumberOfVendorResetsAvailable,
    OUT PUCHAR p__PUCHAR__NumberOfUserControlledChangesAvailable,
    OUT PUCHAR p__PUCHAR__RegionMask,
    OUT PUCHAR p__PUCHAR__RPCScheme
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__TypeCode	Pointer to the variable to receive type code.
OUT PUCHAR p__PUCHAR__NumberOfVendorResetsAvailable	Pointer to the variable to receive number of vendor-controlled region code resets.
OUT PUCHAR p__PUCHAR__NumberOfUserControlledChangesAvailable	Pointer to the variable to receive number of user-controlled region code changes.
OUT PUCHAR p__PUCHAR__RegionMask	Pointer to the variable to receive "hardware" region code mask.
OUT PUCHAR p__PUCHAR__RPCScheme	Pointer to the variable to receive RPC management scheme.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets RPC (region play code) and some additional DVD region management information from "DVD part" of the CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.

Remarks

Please see GrabDisc sample that will demonstrate how DVD region management information could be get from DVD device with the help of the StarBurn_CdvdBurnerGrabber_GetRPC API call.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetDVDRegionMask (see page 68)

Example

This example allocates CdvdBurnerGrabber object, gets DVD region management information from CD/DVD/Blu-Ray/HD-DVD device object and destroys it after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__TypeCode;
UCHAR l__UCHAR__NumberOfVendorResetsAvailable;
UCHAR l__UCHAR__NumberOfUserControlledChangesAvailable;
UCHAR l__UCHAR__RegionMask;
UCHAR l__UCHAR__RPCScheme;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get DVD region management information from DVD device
```

```
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetRPC(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__TypeCode,
    &l__UCHAR__NumberOfVendorResetsAvailable,
    &l__UCHAR__NumberOfUserControlledChangesAvailable,
    &l__UCHAR__RegionMask,
    &l__UCHAR__RPCScheme
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.37 StarBurn_CdvdBurnerGrabber_GetSpeeds Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetSpeeds(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PULONG p__PULONG__CurrentReadSpeed,
    OUT PULONG p__PULONG__MaximumReadSpeed,
    OUT PULONG p__PULONG__CurrentWriteSpeed,
    OUT PULONG p__PULONG__MaximumWriteSpeed
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PULONG p__PULONG__CurrentReadSpeed	Pointer to ULONG that will receive current read speed.
OUT PULONG p__PULONG__MaximumReadSpeed	Pointer to ULONG that will receive maximum read speed.
OUT PULONG p__PULONG__CurrentWriteSpeed	Pointer to ULONG that will receive current write speed.
OUT PULONG p__PULONG__MaximumWriteSpeed	Pointer to ULONG that will receive maximum write speed.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets current read and write and top supported read and write speeds on CD/DVD/Blu-Ray/HD-DVD burner device. This speeds later will be used to set current speeds during all read and write operations.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how StarBurn_CdvdBurnerGrabber_GetSpeeds can be used to determine current and top supported CD/DVD/Blu-Ray/HD-DVD device read and write speeds and how StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153) can be used to set top supported speeds.

[WARNING! It's required to have media inserted for this function to work properly!]

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, gets all (current and top supported) read and write speeds and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
ULONG l__ULONG__CurrentReadSpeed;
ULONG l__ULONG__MaximumReadSpeed;
ULONG l__ULONG__CurrentWriteSpeed;
ULONG l__ULONG__MaximumWriteSpeed;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
```

```
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get current and top supported CD/DVD/Blu-Ray/HD-DVD read/write speeds on the
device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetSpeeds(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__ULONG__CurrentReadSpeed,
    &l__ULONG__MaximumReadSpeed,
    &l__ULONG__CurrentWriteSpeed,
    &l__ULONG__MaximumWriteSpeed
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and it's speeds here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.38
StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats
Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PBOOLEAN p__PBOOLEAN__IsCDRRead,
    OUT PBOOLEAN p__PBOOLEAN__IsCDERead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDROMRead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDRRead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDRAMRead,
    OUT PBOOLEAN p__PBOOLEAN__IsTestWrite,
    OUT PBOOLEAN p__PBOOLEAN__IsCDRWrite,
    OUT PBOOLEAN p__PBOOLEAN__IsCDEWrite,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDRWrite,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDRAMWrite
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)().
OUT PBOOLEAN p__PBOOLEAN__IsCDRRead	Pointer to the boolean variable that will receive is device capable of reading CD-R discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsCDERead	Pointer to the boolean variable that will receive is device capable of reading CD-RW discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDROMRead	Pointer to the boolean variable that will receive is device capable of reading DVD-ROM discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDRRead	Pointer to the boolean variable that will receive is device capable of reading DVD-R discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDRAMRead	Pointer to the boolean variable that will receive is device capable of reading DVD-RAM discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsTestWrite	Pointer to the boolean variable that will receive is device capable of test recording or not.
OUT PBOOLEAN p__PBOOLEAN__IsCDRWrite	Pointer to the boolean variable that will receive is device capable of writing CD-R discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsCDEWrite	Pointer to the boolean variable that will receive is device capable of writing CD-RW discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDRWrite	Pointer to the boolean variable that will receive is device capable of writing DVD-R discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDRAMWrite	Pointer to the boolean variable that will receive is device capable of writing DVD-RAM discs or not.

Returns

None. This function cannot fail.

Description

This function returns CD/DVD/Blu-Ray/HD-DVD burner device object extended information. This data can be used to report the capabilities of the device to the user.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what use can be taken from CdvdBurnerGrabber device extended information.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDeviceInformation (see page 57), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsEx (see page 88)

Example

This example allocates CdvdBurnerGrabber object, retrieves device extended information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
ULONG l__ULONG__CacheBufferSizeInUCHARs;
CHAR l__CHAR__ExceptionText[ 1024 ];
BOOLEAN l__BOOLEAN__IsCDRRead;
BOOLEAN l__BOOLEAN__IsCDERead;
BOOLEAN l__BOOLEAN__IsDVDROMRead;
BOOLEAN l__BOOLEAN__IsDVDRRead;
BOOLEAN l__BOOLEAN__IsDVDRAMRead;
BOOLEAN l__BOOLEAN__IsTestWrite;
BOOLEAN l__BOOLEAN__IsCDRWrite;
BOOLEAN l__BOOLEAN__IsCDEWrite;
```

```
BOOLEAN l__BOOLEAN__IsDVDRWrite;
BOOLEAN l__BOOLEAN__IsDVDRAMWrite;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Get CdvdBurnerGrabber device information here...

// Get CdvdBurnerGrabber device extended information
StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats(
    l__PVOID__CdvdBurnerGrabber,
    &l__BOOLEAN__IsCDRRead,
    &l__BOOLEAN__IsCDERead,
    &l__BOOLEAN__IsDVDROMRead,
    &l__BOOLEAN__IsDVDRRead,
    &l__BOOLEAN__IsDVDRAMRead,
    &l__BOOLEAN__IsTestWrite,
    &l__BOOLEAN__IsCDRWrite,
    &l__BOOLEAN__IsCDEWrite,
    &l__BOOLEAN__IsDVDRWrite,
    &l__BOOLEAN__IsDVDRAMWrite
    );

// Do something with CdvdBurnerGrabber device object and it's extended information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.39 StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsEx Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWRead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRRead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWWrite,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWrite
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)().
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWRead	Pointer to the boolean variable that will receive is device capable of reading DVD+RW discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRRead	Pointer to the boolean variable that will receive is device capable of reading DVD+R discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWWrite	Pointer to the boolean variable that will receive is device capable of writing DVD+RW discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWrite	Pointer to the boolean variable that will receive is device capable of writing DVD+R discs or not.

Returns

None. This function cannot fail.

Description

This function returns DVD+R(W) burner device object extended information. This data can be used to report the capabilities of the device to the user.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the DVD+R(W) media and what use can be taken from CdvdBurnerGrabber device extended information.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDeviceInformation (see page 57), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats (see page 85)

Example

This example allocates CdvdBurnerGrabber object, retrieves device extended information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
```

```
ULONG l__ULONG__CacheBufferSizeInUCHARs;
CHAR l__CHAR__ExceptionText[ 1024 ];
BOOLEAN l__BOOLEAN__IsDVDPLUSRWRead;
BOOLEAN l__BOOLEAN__IsDVDPLUSRRead;
BOOLEAN l__BOOLEAN__IsDVDPLUSRWWrite;
BOOLEAN l__BOOLEAN__IsDVDPLUSRWrite;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Get CdvdBurnerGrabber device information here...

// Get CdvdBurnerGrabber device extended information
StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsEx(
    l__PVOID__CdvdBurnerGrabber,
    &l__BOOLEAN__IsDVDPLUSRWRead,
    &l__BOOLEAN__IsDVDPLUSRRead,
    &l__BOOLEAN__IsDVDPLUSRWWrite,
    &l__BOOLEAN__IsDVDPLUSRWrite
    );

// Do something with CdvdBurnerGrabber device object and it's extended information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.40 StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsExEx Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsExEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWRead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRRead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRDLRead,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWWrite,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWrite,
    OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRDLWrite
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)().
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWRead	Pointer to the boolean variable that will receive is device capable of reading DVD+RW discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRRead	Pointer to the boolean variable that will receive is device capable of reading DVD+R discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRDLRead	Pointer to the boolean variable that will receive is device capable of reading DVD+R DL discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWWrite	Pointer to the boolean variable that will receive is device capable of writing DVD+RW discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRWrite	Pointer to the boolean variable that will receive is device capable of writing DVD+R discs or not.
OUT PBOOLEAN p__PBOOLEAN__IsDVDPLUSRDLWrite	Pointer to the boolean variable that will receive is device capable of writing DVD+R DL discs or not.

Returns

None. This function cannot fail.

Description

This function returns DVD+R(W) burner device object extended extended information. This data can be used to report the capabilities of the device to the user.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the DVD+R(W) media and what use can be taken from CdvdBurnerGrabber device extended information.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDeviceInformation (see page 57), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats (see page 85)

Example

This example allocates CdvdBurnerGrabber object, retrieves device extended extended information and destroys the device

object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
ULONG l__ULONG__CacheBufferSizeInUCHARs;
CHAR l__CHAR__ExceptionText[ 1024 ];
BOOLEAN l__BOOLEAN__IsDVDPLUSRWRead;
BOOLEAN l__BOOLEAN__IsDVDPLUSRRead;
BOOLEAN l__BOOLEAN__IsDVDPLUSRDLRead;
BOOLEAN l__BOOLEAN__IsDVDPLUSRWWrite;
BOOLEAN l__BOOLEAN__IsDVDPLUSRWrite;
BOOLEAN l__BOOLEAN__IsDVDPLUSRDLWrite;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Get CdvdBurnerGrabber device information here...

// Get CdvdBurnerGrabber device extended information
StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsExEx(
    l__PVOID__CdvdBurnerGrabber,
    &l__BOOLEAN__IsDVDPLUSRWRead,
    &l__BOOLEAN__IsDVDPLUSRRead,
    &l__BOOLEAN__IsDVDPLUSRDLRead,
    &l__BOOLEAN__IsDVDPLUSRWWrite,
    &l__BOOLEAN__IsDVDPLUSRWrite,
    &l__BOOLEAN__IsDVDPLUSRDLWrite
    );

// Do something with CdvdBurnerGrabber device object and it's extended information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
```

```
}
```

2.1.41 StarBurn_CdvdBurnerGrabber_GetTOCInformation Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GetTOCInformation(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PTOC_INFORMATION p__PTOC_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PTOC_INFORMATION p__PTOC_INFORMATION	Pointer to the TOC information.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets TOC information of the currently inserted disc in the CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the GrabTrack sample that will demonstrate how ISO9660 or Joliet file system image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetTOCInformation can be used to get current TOC information on the CD/DVD/Blu-Ray/HD-DVD media to determine session information.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571)

Example

This example allocates CdvdBurnerGrabber object, gets TOC information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
```

```
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Try to get current TOC information
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetTOCInformation(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PTOC_INFORMATION )( &l__TOC_INFORMATION )
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.42 StarBurn_CdvdBurnerGrabber_GetTrackInformation Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_GetTrackInformation(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN LONG p__LONG__TrackNumber,
    OUT PSTARBURN_TRACK_INFORMATION p__PSTARBURN_TRACK_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN LONG p__LONG__TrackNumber	Track number to get information about.
OUT PSTARBURN_TRACK_INFORMATION p__PSTARBURN_TRACK_INFORMATION	Pointer to the track information.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets track information of the asked track number on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the TrackAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetTrackInformation can be used to get "invisible" track information on the CD/DVD/Blu-Ray/HD-DVD media to check parameters of the track for validness (is track blank, number of free logical blocks etc).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDiscInformation (see page 62), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), STARBURN_TRACK_INFORMATION (see page 563), STARBURN_DISC_INFORMATION (see page 555)

Example

This example allocates CdvdBurnerGrabber object, gets "invisible" track information and destroys the device object after it's

not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
STARBURN_TRACK_INFORMATION l__STARBURN_TRACK_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare track information buffer
RtlZeroMemory(
    &l__STARBURN_TRACK_INFORMATION,
    sizeof( l__STARBURN_TRACK_INFORMATION )
    );

// Try to get "invisible" track information
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetTrackInformation(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    TRACK_NUMBER_INVISIBLE,
    ( PSTARBURN_TRACK_INFORMATION )( &l__STARBURN_TRACK_INFORMATION )
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and track information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );
```

```
// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.43 StarBurn_CdvdBurnerGrabber_GetTrackInformationEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_GetTrackInformationEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN LONG p__LONG__TrackNumber,
    OUT PSTARBURN_TRACK_INFORMATION_EX p__PSTARBURN_TRACK_INFORMATION_EX
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN LONG p__LONG__TrackNumber	Track number to get information about.
OUT PSTARBURN_TRACK_INFORMATION_EX p__PSTARBURN_TRACK_INFORMATION_EX	Pointer to the track extended information.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function gets track extended information of the asked track number on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the TrackAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94) can be used to get "invisible" track information on the CD/DVD/Blu-Ray/HD-DVD media to check parameters of the track for validness (is track blank, number of free logical blocks etc).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31),

StarBurn_CdvdBurnerGrabber_GetDiscInformation (see page 62), StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), STARBURN_TRACK_INFORMATION_EX (see page 564), STARBURN_TRACK_INFORMATION (see page 563), STARBURN_DISC_INFORMATION (see page 555)

Example

This example allocates CdvdBurnerGrabber object, gets "invisible" track information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
STARBURN_TRACK_INFORMATION_EX l__STARBURN_TRACK_INFORMATION_EX;

// Prepare exception text buffer
RtlZeroMemory(
&l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText )
);

// Prepare CDB failure information
RtlZeroMemory(
&l__CDB_FAILURE_INFORMATION,
sizeof( l__CDB_FAILURE_INFORMATION )
);

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
&l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
( PCALLBACK )( StarBurn_Callback ),
0,
0,
4,
0,
32
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare track information buffer
RtlZeroMemory(
&l__STARBURN_TRACK_INFORMATION_EX,
sizeof( l__STARBURN_TRACK_INFORMATION_EX )
);

// Try to get "invisible" track information
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetTrackInformationEx(
l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
TRACK_NUMBER_INVISIBLE,
( PSTARBURN_TRACK_INFORMATION_EX )( &l__STARBURN_TRACK_INFORMATION_EX )
);

// Check for correct reply
```

```
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and track information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.44 StarBurn_CdvdBurnerGrabber_GrabCD Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabCD(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__FileName,
    IN READ_MODE p__READ_MODE,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
    IN ULONG p__ULONG__ReadReportDelayInSeconds,
    IN BOOLEAN p__BOOLEAN__IsDisableHardwareErrorCorrection
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__FileName	Pointer to file name disc will be stored to (basic name, other one will be generated from it).
IN READ_MODE p__READ_MODE	Read mode (raw, raw + PQ sub-channel or raw + raw P-W sub-channel).
IN LONG p__LONG__Retries	Number of retries on bad block hit.
IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore	Is bad block ignore.
IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced	Is single LB transfer forced.
IN ULONG p__ULONG__ReadReportDelayInSeconds	Read report delay in seconds.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format.

Remarks

Please see the GrabDisc sample that will demonstrate how whole disc image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GrabCD can be used to grab disc from the CD/DVD/Blu-Ray/HD-DVD media.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabTrack (see page 110)

Example

This example allocates CdvdBurnerGrabber object, grabs disc and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it...
```

```
// Try to grab the disc with 2 retries on bad block hit and single read status report per
every 1 second, also use fast software error correction
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GrabCD(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "disc01",
    READ_MODE_RAW,
    NUMBER_OF_READ_RETRIES,
    TRUE,
    FALSE,
    READ_REPORT_DELAY_IN_SECONDS,
    TRUE
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.45 StarBurn_CdvdBurnerGrabber_GrabDVD Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabDVD(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__FileName,
    IN LONG p__LONG__SplitFileSizeInMBs,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
    IN ULONG p__ULONG__ReadReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

IN PCHAR p__PCHAR__FileName	Pointer to file name disc will be stored to (basic name, other one will be generated from it).
IN LONG p__LONG__SplitFileSizeInMBs	Split file size in megabytes.
IN LONG p__LONG__Retries	Number of retries on bad block hit.
IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore	Is bad block ignore.
IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced	Is single LB transfer forced.
IN ULONG p__ULONG__ReadReportDelayInSeconds	Read report delay in seconds.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files.

Remarks

Please see the GrabDisc sample that will demonstrate how whole disc image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GrabDVD can be used to grab disc from the CD/DVD/Blu-Ray/HD-DVD media. For now only DVD media can be grabbed with this [Split] variant of grabbing code.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabTrack (see page 110)

Example

This example allocates CdvdBurnerGrabber object, grabs disc and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
```

```
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it...

// Try to grab the disc with 2 retries on bad block hit and single read status report per
every 1 second, store all the CD/DVD/Blu-Ray/HD-DVD
// image into series of files 1024MB (1GB) each
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GrabDVD(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "disc01",
    1024, // 1GB
    NUMBER_OF_READ_RETRIES,
    TRUE,
    FALSE,
    READ_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.46 StarBurn_CdvdBurnerGrabber_GrabDVDEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabDVDEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PSTARBURN_CREATEFILE_CALLBACK p__PSTARBURN_CREATEFILE_CALLBACK,
    IN PSTARBURN_SET_FILE_POINTER_CALLBACK p__PSTARBURN_SET_FILE_POINTER_CALLBACK,
```

```
    IN PSTARBURN_READFILE_CALLBACK p__PSTARBURN_READFILE_CALLBACK,
    IN PSTARBURN_WRITEFILE_CALLBACK p__PSTARBURN_WRITEFILE_CALLBACK,
    IN PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK p__PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK,
    IN PSTARBURN_CLOSEHANDLE_CALLBACK p__PSTARBURN_CLOSEHANDLE_CALLBACK,
    IN PSTARBURN_DELETEFILE_CALLBACK p__PSTARBURN_DELETEFILE_CALLBACK,
    IN PCHAR p__PCHAR__FileName,
    IN LONG p__LONG__SplitFileSizeInMBs,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
    IN ULONG p__ULONG__ReadReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PSTARBURN_CREATEFILE_CALLBACK p__PSTARBURN_CREATEFILE_CALLBACK	Callback for "create file"
IN PSTARBURN_SET_FILE_POINTER_CALLBACK p__PSTARBURN_SET_FILE_POINTER_CALLBACK	Callback for "set file pointer"
IN PSTARBURN_READFILE_CALLBACK p__PSTARBURN_READFILE_CALLBACK	Callback for "read file"
IN PSTARBURN_WRITEFILE_CALLBACK p__PSTARBURN_WRITEFILE_CALLBACK	Callback for "write file"
IN PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK p__PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK	Callback for "flush file buffer"
IN PSTARBURN_CLOSEHANDLE_CALLBACK p__PSTARBURN_CLOSEHANDLE_CALLBACK	Callback for "close handle"
IN PSTARBURN_DELETEFILE_CALLBACK p__PSTARBURN_DELETEFILE_CALLBACK	Callback for "delete file"
IN PCHAR p__PCHAR__FileName	Pointer to file name disc will be stored to (basic name, other one will be generated from it).
IN LONG p__LONG__SplitFileSizeInMBs	Split file size in megabytes.
IN LONG p__LONG__Retries	Number of retries on bad block hit.
IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore	Is bad block ignore.
IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced	Is single LB transfer forced.
IN ULONG p__ULONG__ReadReportDelayInSeconds	Read report delay in seconds.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function grabs disc image (extended) from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files. StarBurn_CdvdBurnerGrabber_GrabDVDEx function for all I/O files operations use user's callback functions.

Remarks

Please see the GrabDiscEx sample that will demonstrate how whole disc image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GrabDVDEx can be used to grab disc from the CD/DVD/Blu-Ray/HD-DVD media. For now only DVD media can be grabbed with this [Split] variant of grabbing code.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabTrackEx (see page 112)

Example

This example allocates CdvdBurnerGrabber object, grabs disc and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
&l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText )
);

// Prepare CDB failure information
RtlZeroMemory(
&l__CDB_FAILURE_INFORMATION,
sizeof( l__CDB_FAILURE_INFORMATION )
);

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
&l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
( PCALLBACK )( StarBurn_Callback ),
0,
0,
4,
0,
32
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
&l__TOC_INFORMATION,
sizeof( l__TOC_INFORMATION )
);

// Get TOC information here and analyze it...

// Try to grab the disc with 2 retries on bad block hit and single read status report per
every 1 second, store all the CD/DVD/Blu-Ray/HD-DVD
// image into series of files 1024MB (1GB) each
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GrabDVD(
l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
(PSTARBURN_CREATEFILE_CALLBACK)CallbackCreateFile,
```

```
(PSTARBURN_SET_FILE_POINTER_CALLBACK)CallbackSetFilePointer,
(PSTARBURN_READFILE_CALLBACK)CallbackReadFile,
(PSTARBURN_WRITEFILE_CALLBACK)CallbackWriteFile,
(PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK )CallbackFlushFileBuffers,
(PSTARBURN_CLOSEHANDLE_CALLBACK)CallbackCloseHandle,
(PSTARBURN_DELETEFILE_CALLBACK)CallbackDeleteFile,
"disc01",
1024, // 1GB
NUMBER_OF_READ_RETRIES,
TRUE,
FALSE,
READ_REPORT_DELAY_IN_SECONDS
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.47 StarBurn_CdvdBurnerGrabber_GrabDVDFast Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabDVDFast(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PSTARBURN_CREATEFILE_CALLBACK p__PSTARBURN_CREATEFILE_CALLBACK,
    IN PSTARBURN_SET_FILE_POINTER_CALLBACK p__PSTARBURN_SET_FILE_POINTER_CALLBACK,
    IN PSTARBURN_READFILE_CALLBACK p__PSTARBURN_READFILE_CALLBACK,
    IN PSTARBURN_WRITEFILE_CALLBACK p__PSTARBURN_WRITEFILE_CALLBACK,
    IN PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK p__PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK,
    IN PSTARBURN_CLOSEHANDLE_CALLBACK p__PSTARBURN_CLOSEHANDLE_CALLBACK,
    IN PSTARBURN_DELETEFILE_CALLBACK p__PSTARBURN_DELETEFILE_CALLBACK,
    IN PCHAR p__PCHAR__FileName,
    IN LONG p__LONG__SplitFileSizeInMBs,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
    IN ULONG p__ULONG__ReadReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.

IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PSTARBURN_CREATEFILE_CALLBACK p__PSTARBURN_CREATEFILE_CALLBACK	Callback for "create file"
IN PSTARBURN_SET_FILE_POINTER_CALLBACK p__PSTARBURN_SET_FILE_POINTER_CALLBACK	Callback for "set file pointer"
IN PSTARBURN_READFILE_CALLBACK p__PSTARBURN_READFILE_CALLBACK	Callback for "read file"
IN PSTARBURN_WRITEFILE_CALLBACK p__PSTARBURN_WRITEFILE_CALLBACK	Callback for "write file"
IN PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK p__PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK	Callback for "flush file buffer"
IN PSTARBURN_CLOSEHANDLE_CALLBACK p__PSTARBURN_CLOSEHANDLE_CALLBACK	Callback for "close handle"
IN PSTARBURN_DELETEFILE_CALLBACK p__PSTARBURN_DELETEFILE_CALLBACK	Callback for "delete file"
IN PCHAR p__PCHAR__FileName	Pointer to file name disc will be stored to (basic name, other one will be generated from it).
IN LONG p__LONG__SplitFileSizeInMBs	Split file size in megabytes.
IN LONG p__LONG__Retries	Number of retries on bad block hit.
IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore	Is bad block ignore.
IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced	Is single LB transfer forced.
IN ULONG p__ULONG__ReadReportDelayInSeconds	Read report delay in seconds.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files. Works faster than StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)

Remarks

Please see the GrabDisc sample that will demonstrate how whole disc image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100) can be used to grab disc from the CD/DVD/Blu-Ray/HD-DVD media. For now only DVD media can be grabbed with this [Split] variant of grabbing code.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabTrack (see page 110)

Example

This example allocates CdvdBurnerGrabber object, grabs disc and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
```

```
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it...

// Try to grab the disc with 2 retries on bad block hit and single read status report per
every 1 second, store all the CD/DVD/Blu-Ray/HD-DVD
// image into series of files 1024MB (1GB) each
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GrabDVD(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "disc01",
    1024, // 1GB
    NUMBER_OF_READ_RETRIES,
    TRUE,
    FALSE,
    READ_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
```

```
}
```

2.1.48 StarBurn_CdvdBurnerGrabber_GrabRange Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabRange(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN LONG p__LONG__StartingLBA,
    IN LONG p__LONG__EndingLBA,
    IN PCHAR p__PCHAR__FileName,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
    IN ULONG p__ULONG__ReadReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN LONG p__LONG__StartingLBA	Beginning of the LBA range to grab.
IN LONG p__LONG__EndingLBA	Ending of the LBA range to grab.
IN PCHAR p__PCHAR__FileName	Pointer to file name track will be stored to.
IN LONG p__LONG__Retries	Number of retries.
IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore	Is bad block ignore.
IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced	Is single LB transfer forced.
IN ULONG p__ULONG__ReadReportDelayInSeconds	Read report delay in seconds.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function grabs range of logical blocks from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the GrabTrack sample that will demonstrate how ISO9660 or Joliet file system image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GrabTrack (see page 110) can be used to grab track actually from the CD/DVD/Blu-Ray/HD-DVD media. To grab not whole track but just range of logical blocks TOC must be get first by call to StarBurn_CdvdBurnerGrabber_GetTOCInformation (see page 92) and after this StarBurn_CdvdBurnerGrabber_GrabRange can be used to grab range of logical blocks. Attention! Care should be taken b/s the range is inclusive. So grabbing the range 10 - 100 will take 91 logical blocks from 10 and up to 100.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabCD (see page 98), StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)

Example

This example allocates CdvdBurnerGrabber object, grabs range of logical blocks track and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it...

// Try to grab the range of 1001 logical blocks (range is both sides inclusive) (from TOC
we know that this range 0 = 1000 is
// valid) with 2 retries and read report status every 1 second
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GrabRange(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
```

```
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    0,
    1000,
    "track01range",
    NUMBER_OF_READ_RETRIES,
    TRUE,
    FALSE,
    READ_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.49 StarBurn_CdvdBurnerGrabber_GrabTrack Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabTrack(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN UCHAR p__UCHAR__TrackNumber,
    IN PCHAR p__PCHAR__FileName,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
    IN ULONG p__ULONG__ReadReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN UCHAR p__UCHAR__TrackNumber	Track number to grab.
IN PCHAR p__PCHAR__FileName	Pointer to file name track will be stored to.
IN LONG p__LONG__Retries	Number of retries.
IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore	Is bad block ignore.
IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced	Is single LB transfer forced.
IN ULONG p__ULONG__ReadReportDelayInSeconds	Read report delay in seconds.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function grabs track from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the GrabTrack sample that will demonstrate how ISO9660 or Joliet file system image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GrabTrack can be used to grab track actually from the CD/DVD/Blu-Ray/HD-DVD media.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabCD (see page 98), StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)

Example

This example allocates CdvdBurnerGrabber object, grabs first track and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
```

```
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it...

// Try to grab the track (from TOC we know that this is audio track) with 2 retries and
read report status every 1 second
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GrabTrack(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    1,
    "track01.mp3",
    NUMBER_OF_READ_RETRIES,
    TRUE,
    FALSE,
    READ_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.50 StarBurn_CdvdBurnerGrabber_GrabTrackEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabTrackEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PSTARBURN_CREATEFILE_CALLBACK p__PSTARBURN_CREATEFILE_CALLBACK,
    IN PSTARBURN_WRITEFILE_CALLBACK p__PSTARBURN_WRITEFILE_CALLBACK,
    IN PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK p__PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK,
    IN PSTARBURN_CLOSEHANDLE_CALLBACK p__PSTARBURN_CLOSEHANDLE_CALLBACK,
    IN PSTARBURN_DELETEFILE_CALLBACK p__PSTARBURN_DELETEFILE_CALLBACK,
    IN UCHAR p__UCHAR__TrackNumber,
    IN PCHAR p__PCHAR__FileName,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
```

```
    IN ULONG p__ULONG__ReadReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PSTARBURN_CREATEFILE_CALLBACK p__PSTARBURN_CREATEFILE_CALLBACK	Callback for "create file"
IN PSTARBURN_WRITEFILE_CALLBACK p__PSTARBURN_WRITEFILE_CALLBACK	Callback for "write file"
IN PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK p__PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK	Callback for "flush file buffer"
IN PSTARBURN_CLOSEHANDLE_CALLBACK p__PSTARBURN_CLOSEHANDLE_CALLBACK	Callback for "close handle"
IN PSTARBURN_DELETEFILE_CALLBACK p__PSTARBURN_DELETEFILE_CALLBACK	Callback for "delete file"
IN UCHAR p__UCHAR__TrackNumber	Track number to grab.
IN PCHAR p__PCHAR__FileName	Pointer to file name track will be stored to.
IN LONG p__LONG__Retries	Number of retries.
IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore	Is bad block ignore.
IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced	Is single LB transfer forced.
IN ULONG p__ULONG__ReadReportDelayInSeconds	Read report delay in seconds.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function grabs track (extended) from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device. StarBurn_CdvdBurnerGrabber_GrabTrackEx function for all I/O files operations use user's callback functions.

Remarks

Please see the GrabTrackEx sample that will demonstrate how ISO9660 or Joliet file system image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_GrabTrackEx can be used to grab track actually from the CD/DVD/Blu-Ray/HD-DVD media.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabDVDEx (see page 102), StarBurn_CdvdBurnerGrabber_GrabDVDFast (see page 105)

Example

This example allocates CdvdBurnerGrabber object, grabs first track and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
```

```
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
&l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText )
);

// Prepare CDB failure information
RtlZeroMemory(
&l__CDB_FAILURE_INFORMATION,
sizeof( l__CDB_FAILURE_INFORMATION )
);

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
&l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
( PCALLBACK )( StarBurn_Callback ),
0,
0,
4,
0,
32
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
&l__TOC_INFORMATION,
sizeof( l__TOC_INFORMATION )
);

// Get TOC information here and analyze it...

// Try to grab the track (from TOC we know that this is audio track) with 2 retries and
read report status every 1 second
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GrabTrackEx(
l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
(PSTARBURN_CREATEFILE_CALLBACK)CallbackCreateFile,
(PSTARBURN_WRITEFILE_CALLBACK)CallbackWriteFile,
(PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK )CallbackFlushFileBuffers,
(PSTARBURN_CLOSEHANDLE_CALLBACK)CallbackCloseHandle,
(PSTARBURN_DELETEFILE_CALLBACK)CallbackDeleteFile,
1,
"track01.mp3",
NUMBER_OF_READ_RETRIES,
TRUE,
FALSE,
READ_REPORT_DELAY_IN_SECONDS
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
```

```
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.51 StarBurn_CdvdBurnerGrabber_GrabTrackUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_GrabTrackUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN UCHAR p__UCHAR__TrackNumber,
    IN PWCHAR p__PWCHAR__FileName,
    IN LONG p__LONG__Retries,
    IN BOOLEAN p__BOOLEAN__IsBadBlockIgnore,
    IN BOOLEAN p__BOOLEAN__IsSingleLBTransferForced,
    IN ULONG p__ULONG__ReadReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_GrabTrackUnicode.

2.1.52 StarBurn_CdvdBurnerGrabber_IsDiscBlank Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_IsDiscBlank(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PBOOLEAN p__PBOOLEAN__IsBlank
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.

OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PBOOLEAN p__PBOOLEAN__IsBlank	Pointer to the BOOLEAN will be set to TRUE if the disc is blank and FALSE otherwise.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function detect blank state of currently inserted disc in CD/DVD/Blu-Ray/HD-DVD burner device.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetDiscInformation (see page 62), StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94), StarBurn_CdvdBurnerGrabber_GetTOCInformation (see page 92), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), STARBURN_DISC_INFORMATION (see page 555), STARBURN_TRACK_INFORMATION (see page 563), STARBURN_TOC_INFORMATION

Example

This example allocates CdvdBurnerGrabber object, detect is disc blank and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
BOOLEAN l__BOOLEAN__IsDiscBlank(FALSE);

// Prepare exception text buffer
RtlZeroMemory(
&l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText )
);

// Prepare CDB failure information
RtlZeroMemory(
&l__CDB_FAILURE_INFORMATION,
sizeof( l__CDB_FAILURE_INFORMATION )
);

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
&l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
( PCALLBACK )( StarBurn_Callback ),
0,
0,
4,
0,
```

```
32
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to detect is disc blank
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_IsDiscBlank(
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
&l__BOOLEAN__IsDiscBlank
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.53 StarBurn_CdvdBurnerGrabber_Load Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_Load(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the

exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function loads the media on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the DiscOnceFromTree sample demonstrating how ISO9660 or Joliet file system image can be burnt on the CD media and how StarBurn_CdvdBurnerGrabber_Load can be used to load the CD media to initiate file system mount sequence.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_LoadEx (see page 119), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, ejects the media, loads it and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to eject the media here...

// Try to load the media
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Load(
```

```
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.54 StarBurn_CdvdBurnerGrabber_LoadEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_LoadEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN BOOLEAN p__BOOLEAN__IsWaitRequired
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN BOOLEAN p__BOOLEAN__IsWaitRequired	Should the library wait for CD/DVD/Blu-Ray/HD-DVD drive to become into ready state.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function loads the media on CD/DVD/Blu-Ray/HD-DVD burner device. Unlike StarBurn_CdvdBurnerGrabber_Load (see page 117)(...) this API call allows to control should it return immediately or wait until the drive would become into ready

state.

Remarks

Please see the DiscAtOnceFromTree sample demonstrating how ISO9660 or Joliet file system image can be burnt to the CD media and how StarBurn_CdvdBurnerGrabber_Load (see page 117) can be used to load the CD media to initiate file system mount sequence.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_Load (see page 117), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, ejects the media, loads it (waiting for the drive to become into ready state) and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to eject the media here...

// Try to load the media
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_LoadEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    TRUE // We want to return only after drive would become into ready state
```

```
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.55 StarBurn_CdvdBurnerGrabber_Lock Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_Lock(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function locks the media inside the CD/DVD/Blu-Ray/HD-DVD device.

Remarks

Please see the TrackAtOnceFromFile and TrackAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media. StarBurn_CdvdBurnerGrabber_Lock can be used to lock the CD/DVD/Blu-Ray/HD-DVD media to prevent media removal from the device while burning (or any I/O process) is in progress.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_Release (see page 137)

Example

This example allocates CdvdBurnerGrabber object, locks it and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to lock the media
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Lock(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );
```

```
// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.56 StarBurn_CdvdBurnerGrabber_MSFToLBA Function

C++

```
__stdcall STARBURN_IMPEX_API LONG StarBurn_CdvdBurnerGrabber_MSFToLBA(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    IN UCHAR p__UCHAR__M,
    IN UCHAR p__UCHAR__S,
    IN UCHAR p__UCHAR__F,
    IN BOOLEAN p__BOOLEAN__IsForcePositive
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to CD/DVD/Blu-Ray/HD-DVD burner/grabber object.
IN UCHAR p__UCHAR__M	Minute of MSF address.
IN UCHAR p__UCHAR__S	Second of MSF address.
IN UCHAR p__UCHAR__F	Frame of MSF address.
IN BOOLEAN p__BOOLEAN__IsForcePositive	TRUE if resulting LBA should be forced to be positive or FALSE if it could be negative.

Returns

LBA (logical block address).

Description

This function converts MSF (minute:second:frame) address into the LBA (logical block address).

Remarks

No remarks.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)

Example

Please see DiscAtOnceFromFile sample to see how to use StarBurn_MSFToLBA API call should be used.

2.1.57 StarBurn_CdvdBurnerGrabber_ProbeSupportedReadModes Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
```

```
StarBurn_CdvdBurnerGrabber_ProbeSupportedReadModes(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PBOOLEAN p__PBOOLEAN__IsCooked,
    OUT PBOOLEAN p__PBOOLEAN__IsRaw,
    OUT PBOOLEAN p__PBOOLEAN__IsRawQ,
    OUT PBOOLEAN p__PBOOLEAN__IsRawRawPW,
    OUT PBOOLEAN p__PBOOLEAN__IsPQ,
    OUT PBOOLEAN p__PBOOLEAN__IsRawPW
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
p__BOOLEAN__IsCooked	Is cooked mode supported.
p__BOOLEAN__IsRaw	Is raw mode supported.
p__BOOLEAN__IsRawPQ	Is raw + PQ sub-channel mode supported.
p__BOOLEAN__IsRawRawPW	Is raw + raw P-W sub-channel mode supported.
p__BOOLEAN__IsPQ	Is PQ sub-channel mode supported.
p__BOOLEAN__IsRawPW	Is raw P-W sub-channel mode supported.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function probes supported read modes on the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the GrabDisc sample that will demonstrate how whole disc image can be grabbed from the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_ProbeSupportedReadModes can be used to determine supported read modes.

[WARNING! It's required to have media inserted for this function to work properly!]

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabCD (see page 98), StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)

Example

This example allocates CdvdBurnerGrabber object, probes supported read modes and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
```

```
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
BOOLEAN l__BOOLEAN__IsCooked = FALSE;
BOOLEAN l__BOOLEAN__IsRaw = FALSE;
BOOLEAN l__BOOLEAN__IsRawPQ = FALSE;
BOOLEAN l__BOOLEAN__IsRawRawPW = FALSE;
BOOLEAN l__BOOLEAN__IsPQ = FALSE;
BOOLEAN l__BOOLEAN__IsRawPW = FALSE;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to probe supported read modes
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_ProbeSupportedReadModes(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__BOOLEAN__IsCooked,
    &l__BOOLEAN__IsRaw,
    &l__BOOLEAN__IsRawPQ,
    &l__BOOLEAN__IsRawRawPW,
    &l__BOOLEAN__IsPQ,
    &l__BOOLEAN__IsRawPW
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and supported read modes information
here...

// Destroy the CdvdBurnerGrabber
```

```
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.58 StarBurn_CdvdBurnerGrabber_ProbeSupportedWriteModes Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_ProbeSupportedWriteModes(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PBOOLEAN p__PBOOLEAN__IsTrackAtOnce,
    OUT PBOOLEAN p__PBOOLEAN__IsSessionAtOnce,
    OUT PBOOLEAN p__PBOOLEAN__IsDiscAtOncePQ,
    OUT PBOOLEAN p__PBOOLEAN__IsDiscAtOnceRawPW
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
p__BOOLEAN__IsTrackAtOnce	Is Track-At-Once supported.
p__BOOLEAN__IsSessionAtOnce	Is Session-At-Once supported.
p__BOOLEAN__IsDiscAtOncePQ	Is Disc-At-Once PQ supported.
p__BOOLEAN__IsDiscAtOnceRawPW	Is Disc-At-Once raw PW supported.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function probes supported read modes on the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the burning samples that will demonstrate how disc can be recorded and how

StarBurn_CdvdBurnerGrabber_ProbeSupportedWriteModes can be used to determine supported write modes.

[WARNING! It's required to have media inserted for this function to work properly!]

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184)

Example

This example allocates CdvdBurnerGrabber object, probes supported write modes and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
BOOLEAN l__BOOLEAN__IsTrackAtOnce = FALSE;
BOOLEAN l__BOOLEAN__IsSessionAtOnce = FALSE;
BOOLEAN l__BOOLEAN__IsDiscAtOncePQ = FALSE;
BOOLEAN l__BOOLEAN__IsDiscAtOnceRawPW = FALSE;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to probe supported write modes
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_ProbeSupportedWriteModes(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__BOOLEAN__IsTrackAtOnce,
```

```
    &l__BOOLEAN__IsSessionAtOnce,
    &l__BOOLEAN__IsDiscAtOncePQ,
    &l__BOOLEAN__IsDiscAtOnceRawPW
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and supported write modes information
here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.59 StarBurn_CdvdBurnerGrabber_ReadATIP Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_ReadATIP(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__ATIP,
    IN ULONG p__ULONG__ATIPSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__ATIP	Pointer to the buffer to receive ATIP information.
IN ULONG p__ULONG__ATIPSizeInUCHARs	ATIP information buffer size in UCHARs.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function reads ATIP information from the media currently inserted into CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or

StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.

Remarks

Please see the DiscAtOnceFromTree sample that will demonstrate how ATIP information could be read from the media with the help of the StarBurn_CdvdBurnerGrabber_ReadATIP API call.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, reads ATIP information and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UCHAR l__UCHAR__ATIP[ 4096 ];

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get ATIP information from device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_ReadATIP(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__UCHAR__ATIP,
    sizeof( l__UCHAR__ATIP )
    );

// Check for correct reply
```

```
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.60 StarBurn_CdvdBurnerGrabber_ReadCooked Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_ReadCooked(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    OUT PUCHAR p__PUCHAR__Read,
    IN ULONG p__ULONG__ReadSizeInUCHARs,
    IN LONG p__LONG__LBA,
    IN ULONG p__ULONG__TransferSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
OUT PUCHAR p__PUCHAR__Read	Pointer to data buffer to receive read data to.
IN ULONG p__ULONG__ReadSizeInUCHARs	Read size in unsigned char(s).
IN LONG p__LONG__LBA	Starting LBA to read from.
IN ULONG p__ULONG__TransferSizeInLBs	Whole transfer size in logical block(s).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function reads one or more logical block(s) from the currently inserted CD/DVD/Blu-Ray/HD-DVD in cooked format.

Remarks

Make sure your data buffer is page-aligned, transfer size is less then 64KB and all of the input parameters are valid.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabCD (see page 98), StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100), StarBurn_CdvdBurnerGrabber_GrabRange (see page 108), StarBurn_CdvdBurnerGrabber_ReadLB (see page 132), StarBurn_CdvdBurnerGrabber_ReadRaw (see page 134)

Example

This example allocates CdvdBurnerGrabber object, reads 2 (two) cooked logical blocks and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;
UCHAR l__UCHAR__Buffer[ 8192 ]; // We do allocate buffer on the stack but it's not correct.
Page-aligned memory must be used.

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Prepare the buffer
RtlZeroMemory(
    &l__UCHAR__Buffer,
    sizeof( l__UCHAR__Buffer )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
```

```
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it

// Get currently inserted disc and analyze it

// Try to grab two cooked logical blocks starting from LBA 100 (from TOC we know that
address 100 is valid)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_ReadCooked(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PUCHAR )( &l__UCHAR__Buffer ),    // Pointer to data buffer to receive the payload
    sizeof( l__UCHAR__Buffer ),         // Data buffer size in unsigned char(s)
    100,                                // Start from logical block number 100
    2                                   // 2 logical blocks
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and read sectors here

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.61 StarBurn_CdvdBurnerGrabber_ReadLB Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_ReadLB(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN LONG p__LONG__LBA,
    IN PUCHAR p__PUCHAR__Buffer,
    IN LONG p__ULONG__BufferSize
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

IN LONG p__LONG__LBA	LBA of logical block to read.
IN PUCHAR p__PUCHAR__Buffer	Pointer to output buffer.
IN LONG p__ULONG__BufferSize	Output buffer size.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function reads one logical block from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabCD (see page 98), StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100), StarBurn_CdvdBurnerGrabber_GrabRange (see page 108)

Example

This example allocates CdvdBurnerGrabber object, reads one logical block and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

```
// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it...

// Allocate buffer
UCHAR l__UCHAR__Buffer[4096];

// Try to grab one logical block with address 1000 (from TOC we know that this address 1000
is valid)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_ReadLB(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    1000,
    l__UCHAR__Buffer,
    4096
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and disc information here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.62 StarBurn_CdvdBurnerGrabber_ReadRaw Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_ReadRaw(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN UCHAR p__UCHAR__ExpectedLBType,
    IN LONG p__LONG__LBA,
    IN ULONG p__ULONG__DataBufferSizeInLBs,
    IN UCHAR p__UCHAR__ErrorField,
    IN BOOLEAN p__BOOLEAN__IsEDC_ECC,
    IN BOOLEAN p__BOOLEAN__IsUserData,
    IN UCHAR p__UCHAR__HeaderCodes,
    IN BOOLEAN p__BOOLEAN__IsSYNC,
    IN UCHAR p__UCHAR__SubChannelSelectionBits,
    OUT PUCHAR p__PUCHAR__DataBuffer,
    IN ULONG p__ULONG__DataBufferSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN UCHAR p__UCHAR__ExpectedLBType	Expected logical block type. See EXPECTED_LB_TYPE_XXX constants.
IN LONG p__LONG__LBA	Starting logical block number to read from.
IN ULONG p__ULONG__DataBufferSizeInLBs	Data buffer/request size in logical block(s).
IN UCHAR p__UCHAR__ErrorField	Error format. See ERROR_FIELD_XXX constants.
IN BOOLEAN p__BOOLEAN__IsEDC_ECC	TRUE if you want EDC/ECC data included in the output stream, FALSE if you don't want EDC/ECC.
IN BOOLEAN p__BOOLEAN__IsUserData	TRUE if you want user data (main payload) included in the output stream, FALSE if you don't want user data.
IN UCHAR p__UCHAR__HeaderCodes	Header format. See HEADER_CODE_XXX constants.
IN BOOLEAN p__BOOLEAN__IsSYNC	TRUE if you want SYNC pattern included in the output stream, FALSE if you don't want SYNC pattern.
IN UCHAR p__UCHAR__SubChannelSelectionBits	Sub-channel selection. See SUBCHANNEL_XXX constants.
OUT PUCHAR p__PUCHAR__DataBuffer	Pointer to data buffer to receive the payload.
IN ULONG p__ULONG__DataBufferSizeInUCHARs	Data buffer size in unsigned char(s).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function reads one or more logical block(s) from the currently inserted CD in RAW format.

Remarks

Make sure your data buffer is page-aligned, transfer size is less then 64KB and all of the input parameters are valid.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), TOC_INFORMATION (see page 573), TOC_ENTRY (see page 571), StarBurn_CdvdBurnerGrabber_GrabCD (see page 98), StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100), StarBurn_CdvdBurnerGrabber_GrabRange (see page 108), StarBurn_CdvdBurnerGrabber_ReadLB (see page 132), StarBurn_CdvdBurnerGrabber_ReadCooked (see page 130)

Example

This example allocates CdvdBurnerGrabber object, reads 2 (two) RAW logical blocks and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
TOC_INFORMATION l__TOC_INFORMATION;
UCHAR l__UCHAR__Buffer[ 8192 ]; // We do allocate buffer on the stack but it's not correct.
Page-aligned memory must be used.

// Prepare exception text buffer
RtlZeroMemory(
```

```
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Prepare the buffer
RtlZeroMemory(
    &l__UCHAR__Buffer,
    sizeof( l__UCHAR__Buffer )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare TOC information buffer
RtlZeroMemory(
    &l__TOC_INFORMATION,
    sizeof( l__TOC_INFORMATION )
    );

// Get TOC information here and analyze it

// Get currently inserted disc and analyze it. We can do RAW read from CD media only

// Probe supported read modes as we need to read in the ones drive really supports

// Try to grab two RAW logical blocks starting from LBA 100 (from TOC we know that address
100 is valid)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_ReadRaw(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    EXPECTED_LB_TYPE_ANY,               // Accept any logical blocks
    100,                                // Start from logical block number 100
    2,                                  // 2 logical blocks only
    ERROR_FIELD_NO_ERROR,               // No error data included
    TRUE,                               // YES, we need EDC/ECC data
    TRUE,                               // YES, we need user data
    HEADER_CODE_ALL_HEADERS,            // We need ALL of the headers
    TRUE,                               // YES, we need SYNC pattern
    SUBCHANNEL_RAW_PW,                  // Read RAW P-W sub-channel from the disc
    ( PUCHAR )( &l__UCHAR__Buffer ),    // Pointer to data buffer to receive the payload
    sizeof( l__UCHAR__Buffer )          // Data buffer size in unsigned char(s)
    );
```

```
// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object and read sectors here

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.63 StarBurn_CdvdBurnerGrabber_Release Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_Release(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function releases the media inside the CD/DVD/Blu-Ray/HD-DVD device after the media was locked before.

Remarks

Please see the TrackAtOnceFromFile and TrackAtOnceFromTree sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media. StarBurn_CdvdBurnerGrabber_Lock (see page 121) can be used to lock the CD/DVD/Blu-Ray/HD-DVD media to prevent media removal from the device while burning (or any I/O process) is in progress. StarBurn_CdvdBurnerGrabber_Release can be used to release (unlock) the media that was locked before.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_Lock (see page 121)

Example

This example allocates CdvdBurnerGrabber object, locks it, releases (unlocks) and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to lock media here...

// Try to release (unlock) the media
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Release(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...
```

```
// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.64 StarBurn_CdvdBurnerGrabber_SendOPC Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SendOPC(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function sends OPC (Optimum Power Calibration) for current write speed on CD/DVD/Blu-Ray/HD-DVD burner device and current CD/DVD/Blu-Ray/HD-DVD media. The set laser level will be used later during all write operations.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_SendOPC can be used to set optimum laser level on CD/DVD/Blu-Ray/HD-DVD device for current write speed and CD/DVD/Blu-Ray/HD-DVD media.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, sends optimum power calibration for current write speed and current CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to send optimum power calibration
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SendOPC(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.65 StarBurn_CdvdBurnerGrabber_SessionAtOnce Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SessionAtOnce(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PDISC_LAYOUT p__PDISC_LAYOUT,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PDISC_LAYOUT p__PDISC_LAYOUT	Pointer to disc layout filled with file tree specific data.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is a just ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) or ISO image with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Session-At-Once mode.

Remarks

Please see the SessionAtOnceFromTree sample that will demonstrate how "virtual" ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_SessionAtOnce. SessionAtOnceFromFile sample shows how to burn ISO image.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image from file tree to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__FileTree;
DISC_LAYOUT l__DISC_LAYOUT;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to create ISO9660 or Joliet file system image with help of
StarBurn_ISO9660JolietFileTree_Create and
// StarBurn_ISO9660JolietFileTree_BuildImage, l__PVOID__FileTree used as the pointer to the
object...

// Fill DISC_LAYOUT structure here

RtlZeroMemory(
    &l__DISC_LAYOUT,
    sizeof( l__DISC_LAYOUT )
    );

l__DISC_LAYOUT.m__LONG__NumberOfEntries = 1;
```

```
l__DISC_LAYOUT.m__DISC_LAYOUT_ENTRY[ 0 ].m__PVOID__ISO9660JolietFileTree =
l__PVOID__FileTree;

// Try to record the ISO9660 or Joliet file system image in Session-At-Once mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SessionAtOnce(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__DISC_LAYOUT,
    FALSE,
    FALSE,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.66 StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW

Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__MDS,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
    IN BOOLEAN p__BOOLEAN__IsRawSaoSubManualLeadIn,
    IN BOOLEAN p__BOOLEAN__IsAmplifyWeak
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__MDS	Pointer to MDS file name.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
IN BOOLEAN p__BOOLEAN__IsRawSaoSubManualLeadIn	Should we write lead-in manually (TRUE) or should we rely on the drive doing this (FALSE).
IN BOOLEAN p__BOOLEAN__IsAmplifyWeak	Should we amplify weak patterns (TRUE) or just forward all the data to the drive "AS IS" (FALSE).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records raw image with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in raw Session-At-Once mode.

Remarks

Please see the SessionAtOnceRawRawPW sample that will demonstrate how burn raw image to CD media in RAW SAO + SUB (raw Session-At-Once) mode.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141), StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW

Example

This example allocates CdvdBurnerGrabber object, records raw image to the CD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );
```

```
// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the raw image in RAW SUB Session-At-Once mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "Image.MDS",
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS,
    TRUE,   // Write lead-in manually
    TRUE    // Amplify weak
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.67
StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW Unicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__MDS,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
    IN BOOLEAN p__BOOLEAN__IsRawSaoSubManualLeadIn,
    IN BOOLEAN p__BOOLEAN__IsAmplifyWeak
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWUnicode.

2.1.68
StarBurn_CdvdBurnerGrabber_SessionAtOnceUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_SessionAtOnceUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PDISC_LAYOUT p__PDISC_LAYOUT,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_SessionAtOnceUnicode.

2.1.69 StarBurn_CdvdBurnerGrabber_SetBUP Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SetBUP(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN BOOLEAN p__BOOLEAN__IsBUPEnabled
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN BOOLEAN p__BOOLEAN__IsBUPEnabled	BOOLEAN variable that keeps TRUE if BUP is must be enabled on the CD/DVD/Blu-Ray/HD-DVD device, FALSE if BUP must be disabled.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function sets BUP (Buffer Underrun Protection) on CD/DVD/Blu-Ray/HD-DVD burner device that supports BUP. BUP must be enabled for successful completion of write operations.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what StarBurn_CdvdBurnerGrabber_GetBUP (see page 55) can be used for.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_GetBUP (see page 55), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, gets BUP status and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
```

```
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
BOOLEAN l__BOOLEAN__IsBUPEnabled;
BOOLEAN l__BOOLEAN__IsBUPSupported;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get BUP on the device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_GetBUP(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    &l__BOOLEAN__IsBUPEnabled,
    &l__BOOLEAN__IsBUPSupported
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Check is BUP supported, if yes - try to enable it
if ( l__BOOLEAN__IsBUPSupported == TRUE )
{

// Try to set BUP on the device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SetBUP(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    TRUE
    );

// Check for correct reply
```

```
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.70 StarBurn_CdvdBurnerGrabber_SetCDTextItem Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SetCDTextItem(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    IN CHAR p__CHAR__TrackIndex,
    IN const PCHAR p__PCHAR__Artist,
    IN const PCHAR p__PCHAR__Title,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to CStarBurn_CdvdBurnerGrabber object.
IN CHAR p__CHAR__TrackIndex	1-based track number. Or 0 for the common DISC information. PCHAR p__PCHAR__Artist - Pointer to zero-terminated string with the name of artist/performer/singer. PCHAR p__PCHAR__Title - Pointer to zero-terminated string with the track title/song name.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to buffer for exception text.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.

Returns

Exception number

Description

Sets CD-Text item for the specified CD/DVD/Blu-Ray/HD-DVD device.

Remarks

Please check DiscAtOnceFromFile sample to find out how to use StarBurn_CdvdBurnerGrabber_SetCDTextItem() in the right way. Note, that this function is used to burn CD-TEXT for audio discs using StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile (see page 42).

See Also

StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile (see page 42), StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)

Example

This example allocates CdvdBurnerGrabber object, specifies the CD-Text for a disc and for 3 track and burns the disc.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
DAO_DISC_LAYOUT l__DAO_DISC_LAYOUT;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add files to burn here by filling l__DAO_DISC_LAYOUT information

// Try to set CD-Text information for 3d track
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SetCDTextItem(
    l__PVOID__CdvdBurnerGrabber,
    3,
    "My band",
    "My new song",
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status
    );

// Check for success
if (l__EXCEPTION_NUMBER != EN_SUCCESS)
{
// Handle error here...
}

// Try to set CD-Text information for the whole disc
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SetCDTextItem(
    l__PVOID__CdvdBurnerGrabber,
    0,
    "My band",
```

```
    "My first album",
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status
    );

// Check for success
if (l__EXCEPTION_NUMBER != EN_SUCCESS)
{
// Handle error here...
}

// Try to burn ISO/sound/MDS image to the disc as CDROM XA (MODE2 Form1) or CDDA (CD
digital audio) in Disc-At-Once raw P-W mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    &l__CDB_FAILURE_INFORMATION,
    &l__DAO_DISC_LAYOUT,
    FALSE,
    l__BOOLEAN__IsTestWrite,
    FALSE, // Next session allowed
    FALSE, // Do NOT repair broken sub-channel
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.71 StarBurn_CdvdBurnerGrabber_SetReadSpeed Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SetReadSpeed(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN ULONG p__ULONG__ReadSpeedInKBps
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

IN ULONG p__ULONG__ReadSpeedInKBps	Read speed in kilobytes per second.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function sets read speed for CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object.

Remarks

There are no examples of using StarBurn_CdvdBurnerGrabber_SetReadSpeed however this call is nearly identical to more generic one called StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153). Current one sets only read speed and generic one sets both read and write speeds at the same time.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_GetSpeeds (see page 83)

Example

This example allocates CdvdBurnerGrabber object, sets maximum allowed read speed and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );
```

```
// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to set maximum allowed read speed
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SetReadSpeed(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    0xFFFF
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.72 StarBurn_CdvdBurnerGrabber_SetSpeeds Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SetSpeeds(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN ULONG p__ULONG__ReadSpeed,
    IN ULONG p__ULONG__WriteSpeed
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN ULONG p__ULONG__ReadSpeed	ULONG with read speed to set.
IN ULONG p__ULONG__WriteSpeed	ULONG with write speed to set.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be

EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function sets current read and write speeds on CD/DVD/Blu-Ray/HD-DVD burner device This speeds later will be used during all read and write operations.

Remarks

Please see the TrackAtOnceFromTree, TrackAtOnceFromFile, GrabTrack, GrabDisc samples that will demonstrate how StarBurn_CdvdBurnerGrabber_SetSpeeds can be used for setting CD/DVD/Blu-Ray/HD-DVD device read and write speeds.

[WARNING! It's required to have media inserted for this function to work properly!]

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_CreateEx (see page 33), StarBurn_CdvdBurnerGrabber_SetReadSpeed (see page 151), StarBurn_CdvdBurnerGrabber_GetSpeeds (see page 83)

Example

This example allocates CdvdBurnerGrabber object, sets top supported read and write speeds and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

```
// Try to set top supported CD/DVD/Blu-Ray/HD-DVD read/write speeds on the device
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SetSpeeds(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    0xFFFF,
    0xFFFF
    );

// Check for correct reply
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.73 StarBurn_CdvdBurnerGrabber_StopPlayScan Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_StopPlayScan(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function resets media state inside CD/DVD/Blu-Ray/HD-DVD burner device object. This actions allow to update disc content w/o need to eject the disc. Could be used for multiple burnings w/o disc eject or manual disc data verification.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media. Adding call to StarBurn_CdvdBurnerGrabber_StopPlayScan and some of the manual media read commands would allow faster (maybe better) recorded image verifications then built-in into the library.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172)

Example

This example allocates CdvdBurnerGrabber object, burns something and does manual verification and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Burn image here...

// Try to update device state
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_StopPlayScan(
```

```
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Manually verify recorded image here...

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.74 StarBurn_CdvdBurnerGrabber_SuperVideoCD Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SuperVideoCD(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__MPEG2ImagePathAndFileName,
    IN PCHAR p__PCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__MPEG2ImagePathAndFileName	Pointer to SVCD/MPEG2 image file located in a file and stored on the hard disk.
IN PCHAR p__PCHAR__AlbumIdentifier	Pointer to album identifier.
IN USHORT p__USHORT__NumberOfVolumesInAlbum	Number of volumes in album.
IN USHORT p__USHORT__AlbumSetSequenceNumber	Album set sequence number.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.

IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device.

Remarks

Please see the SuperVideoCD sample that will demonstrate how SVCD/MPEG2 image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_SuperVideoCD with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_VideoCD (see page 203)

Example

This example allocates CdvdBurnerGrabber object, records the SVCD/MPEG2 image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
ULONG l__ULONG__Status;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
```

```
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the SVCD/MPEG2 image (in Track-At-Once mode)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SuperVideoCD(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "C:\MOVIE.MPEG2",
    "Album",
    0x0001,
    0x0001,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.75 StarBurn_CdvdBurnerGrabber_SuperVideoCDEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SuperVideoCDEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN WRITE_MODE p__WRITE_MODE,
    IN PCHAR p__PCHAR__MPEG2ImagePathAndFileName,
    IN PCHAR p__PCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN WRITE_MODE p__WRITE_MODE	Write mode to use.
IN PCHAR p__PCHAR__MPEG2ImagePathAndFileName	Pointer to SVCD/MPEG2 image file located in a file and stored on the hard disk.
IN PCHAR p__PCHAR__AlbumIdentifier	Pointer to album identifier.
IN USHORT p__USHORT__NumberOfVolumesInAlbum	Number of volumes in album.
IN USHORT p__USHORT__AlbumSetSequenceNumber	Album set sequence number.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.

Remarks

Please see the SuperVideoCDEx sample that will demonstrate how SVCD/MPEG2 image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_SuperVideoCDEx with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157), StarBurn_CdvdBurnerGrabber_VideoCD (see page 203), StarBurn_CdvdBurnerGrabber_VideoCDEx (see page 205)

Example

This example allocates CdvdBurnerGrabber object, records the SVCD/MPEG2 image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
```

```
// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the SVCD/MPEG2 image (in Session-At-Once mode)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SuperVideoCDEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    WRITE_MODE_SESSION_AT_ONCE,
    "C:\MOVIE.MPEG2",
    "Album",
    0x0001,
    0x0001,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.76 StarBurn_CdvdBurnerGrabber_SuperVideoCDExEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_SuperVideoCDExEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__MPEG2ImagePathAndFileName,
    IN PCHAR p__PCHAR__TemplatePathAndFileName,
    IN PCHAR p__PCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__MPEG2ImagePathAndFileName	Pointer to SVCD/MPEG2 image file located in a file and stored on the hard disk.
IN PCHAR p__PCHAR__AlbumIdentifier	Pointer to album identifier.
IN USHORT p__USHORT__NumberOfVolumesInAlbum	Number of volumes in album.
IN USHORT p__USHORT__AlbumSetSequenceNumber	Album set sequence number.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
p__WRITE_MODE	Write mode to use.
p__PCHAR__TemplateImagePathAndFileName	Pointer to template file name StarBurn would record as first data track.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.

Remarks

Please see the SuperVideoCDEx sample that will demonstrate how SVCD/MPEG2 image can be burn on the

CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_SuperVideoCDEx (see page 159) with currently set write speed and currently set optimum power calibration. This sample is for legacy call, StarBurn_CdvdBurnerGrabber_SuperVideoCDExEx is basically the same API call except it allows to pass user-defined data track StarBurn would record to SVCD as first track (MPEG2 file is second track on VCD).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157), StarBurn_CdvdBurnerGrabber_VideoCD (see page 203), StarBurn_CdvdBurnerGrabber_VideoCDExEx (see page 208), StarBurn_CdvdBurnerGrabber_SuperVideoCDEx (see page 159)

Example

This example allocates CdvdBurnerGrabber object, records the SVCD/MPEG2 image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more. Custom user file name is passed to have own data track on resulting SVCD.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the SVCD/MPEG2 image (in Session-At-Once mode)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_SuperVideoCDEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
```

```
      &l__ULONG__SystemError,
      &l__CDB_FAILURE_INFORMATION,
      WRITE_MODE_SESSION_AT_ONCE,
      "C:\MOVIE.MPEG2",  // Second MPEG2 track
      "C:\DATA.ISO",     // First DATA track
      "Album",
      0x0001,
      0x0001,
      FALSE,
      WRITE_REPORT_DELAY_IN_SECONDS,
      BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
      );

  // Check for correct reply
  if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
  {
  // Handle error here...
  }

  // Do something with CdvdBurnerGrabber device object here...

  // Destroy the CdvdBurnerGrabber
  StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

  // Just check for pointer (paranoid?)
  if ( l__PVOID__CdvdBurnerGrabber != NULL )
  {
  // Handle error here...
  }
```

2.1.77
StarBurn_CdvdBurnerGrabber_SuperVideoCDExExUnicode Function

C++

```
  __stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
  StarBurn_CdvdBurnerGrabber_SuperVideoCDExExUnicode(
      IN PVOID p__PVOID__CdvdBurnerGrabber,
      OUT PCHAR p__PCHAR__ExceptionText,
      IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
      OUT PULONG p__PULONG__SystemError,
      OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
      IN PWCHAR p__PWCHAR__MPEG2ImagePathAndFileName,
      IN PWCHAR p__PWCHAR__TemplatePathAndFileName,
      IN PWCHAR p__PWCHAR__AlbumIdentifier,
      IN USHORT p__USHORT__NumberOfVolumesInAlbum,
      IN USHORT p__USHORT__AlbumSetSequenceNumber,
      IN BOOLEAN p__BOOLEAN__IsTestWrite,
      IN ULONG p__ULONG__WriteReportDelayInSeconds,
      IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
  );
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDExExUnicode.

2.1.78 StarBurn_CdvdBurnerGrabber_SuperVideoCDExUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_SuperVideoCDExUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN WRITE_MODE p__WRITE_MODE,
    IN PWCHAR p__PWCHAR__MPEG2ImagePathAndFileName,
    IN PWCHAR p__PWCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDExUnicode.

2.1.79 StarBurn_CdvdBurnerGrabber_SuperVideoCDUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_SuperVideoCDUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__MPEG2ImagePathAndFileName,
    IN PWCHAR p__PWCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDUnicode.

2.1.80 StarBurn_CdvdBurnerGrabber_TestUnitReady Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_TestUnitReady(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what StarBurn_CdvdBurnerGrabber_TestUnitReady can be used for. Also it's a good idea to use extended and "double extended" variants of this call (StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see page 168) and StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx (see page 170)) when required.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see page 168), StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx (see page 170)

Example

This example allocates CdvdBurnerGrabber object, tests unit ready and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
```

```
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to test unit ready
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TestUnitReady(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.81 StarBurn_CdvdBurnerGrabber_TestUnitReadyEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_TestUnitReadyEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN ULONG p__ULONG__NumberOfSecondsToWait
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN ULONG p__ULONG__NumberOfSecondsToWait	Number of seconds to wait (not less than the number, maybe more) until the unit would not become into ready state.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations. The only difference between this and simple StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166)() is that this code passed parameter to wait for unit to become ready. And "simple" code uses default timeout of 20 seconds.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166) and it's more extended "clone" StarBurn_CdvdBurnerGrabber_TestUnitReadyEx and "double extended" StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx (see page 170) can be used for.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166), StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx (see page 170)

Example

This example allocates CdvdBurnerGrabber object, tests unit ready for 10 seconds (instead of default 20) and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to test unit ready for 10 seconds
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TestUnitReadyEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    10 // 10 seconds to test unit to become into ready state
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
```

```
// Handle error here...
}
```

2.1.82 StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN ULONG p__ULONG__NumberOfSecondsToWait,
    IN BOOLEAN p__BOOLEAN__IsFastExitUsed
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN ULONG p__ULONG__NumberOfSecondsToWait	Number of seconds to wait (not less than the number, maybe more) for device to become into ready state.
IN BOOLEAN p__BOOLEAN__IsFastExitUsed	Should StarBurn do "fast exit" if there's no disc in drive.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations. This "double extended" variant of the call also can wait for particular amount of seconds and do "fast exit" if there's no disc in drive.

Remarks

Please see the TrackAtOnceFromTree and TrackAtOnceFromFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and what StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx can be used for. Also it's a good idea to check either normal StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166) or basic extended variant of this call called StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see page 168). Some of them may work better in particular situation.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166), StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see

page 168)

Example

This example allocates CdvdBurnerGrabber object, tests unit ready and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to test unit ready
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    5,      // 5 seconds to wait for device to become ready
    TRUE    // Fast exit if there's no disc in drive
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
```

```
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.83 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__ISOSoundImagePathAndFileName,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__ISOSoundImagePathAndFileName	Pointer to ISO9660 or Joliet file system image or sound file located in a file and stored on the hard disk.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be sound file if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the TrackAtOnceFromFile sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the ISO9660 or Joliet file system image in Track-At-Once mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "C:\ISO9660.ISO",
    FALSE,
```

```
    FALSE,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.84
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__SoundImagePathAndFileName,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to CStarBurn_CdvdBurnerGrabber object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB failure information.
IN PWCHAR p__PWCHAR__SoundImagePathAndFileName	Pointer to sound path and file name in Unicode format.
IN BOOLEAN p__BOOLEAN__IsXA	Is this CDROM XA or just ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	Is this test write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	Is next session allowed.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).

IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Exception number

Description

This API call deals with audio tracks with Unicode names only.

2.1.85 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicodeEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicodeEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__SoundImagePathAndFileName,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
    IN USHORT p__USHORT__AudioPauseLengthInLBs,
    IN BOOLEAN p__BOOLEAN__IsAudioPauseChange
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to CStarBurn_CdvdBurnerGrabber object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB failure information.
IN PWCHAR p__PWCHAR__SoundImagePathAndFileName	Pointer to sound path and file name in Unicode format.
IN BOOLEAN p__BOOLEAN__IsXA	Is this CDROM XA or just ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	Is this test write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	Is next session allowed.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
IN USHORT p__USHORT__AudioPauseLengthInLBs	Audio pause length in logical blocks.
IN BOOLEAN p__BOOLEAN__IsAudioPauseChange	Should we change audio pause length.

Returns

Exception number

Description

This API call deals with audio tracks with Unicode names only. Also it allows to set track-to-track pause length.

2.1.86 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__ISOSoundImagePathAndFileName,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
    IN USHORT p__USHORT__AudioPauseLengthInLBs,
    IN BOOLEAN p__BOOLEAN__IsAudioPauseChange
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__ISOSoundImagePathAndFileName	Pointer to ISO9660 or Joliet file system image or sound file located in a file and stored on the hard disk.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
IN USHORT p__USHORT__AudioPauseLengthInLBs	Audio pause length in logical blocks.
IN BOOLEAN p__BOOLEAN__IsAudioPauseChange	Is audio pause change.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be sound file if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the TrackAtOnceFromFile sample that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172) with currently set write speed and currently set optimum power calibration. The same sample uses extended version of the call to control audio pause length.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, records sound track image to the CD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber = NULL;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError = ERROR_SUCCESS;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to sound track image in Track-At-Once mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "C:\TRACK.WAV",
```

```
      FALSE,
      FALSE,
      FALSE,
      WRITE_REPORT_DELAY_IN_SECONDS,
      BUFFER_STATUS_REPORT_DELAY_IN_SECONDS,
      0x0000, // Zero-sized pause between tracks
      TRUE    // Yes, apply pause change
      );

  // Check for correct reply
  if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
  {
  // Handle error here...
  }

  // Do something with CdvdBurnerGrabber device object here...

  // Destroy the CdvdBurnerGrabber
  StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

  // Just check for pointer (paranoid?)
  if ( l__PVOID__CdvdBurnerGrabber != NULL )
  {
  // Handle error here...
  }
```

2.1.87
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileUnicode Function

C++

```
  __stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
  StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileUnicode(
      IN PVOID p__PVOID__CdvdBurnerGrabber,
      OUT PCHAR p__PCHAR__ExceptionText,
      IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
      OUT PULONG p__PULONG__SystemError,
      OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
      IN PWCHAR p__PWCHAR__ISOSoundImagePathAndFileName,
      IN BOOLEAN p__BOOLEAN__IsXA,
      IN BOOLEAN p__BOOLEAN__IsTestWrite,
      IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
      IN ULONG p__ULONG__WriteReportDelayInSeconds,
      IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
  );
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileUnicode.

2.1.88 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemory Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemory(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN OUT PVOID p__PVOID__MemoryRegion,
    IN LARGE_INTEGER p__LARGE_INTEGER__MemoryRegionSizeInUCHARs,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
    IN BOOLEAN p__BOOLEAN__IsAudio
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN OUT PVOID p__PVOID__MemoryRegion	Pointer to memory region containing data.
IN LARGE_INTEGER p__LARGE_INTEGER__MemoryRegionSizeInUCHARs	Memory region size in UCHARs.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
IN BOOLEAN p__BOOLEAN__IsAudio	Is this audio stream (TRUE) or data stream (FALSE).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image or audio stream located in a memory with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the TrackAtOnceFromPipe and TrackAtOnceFromPipeEx API calls to find out how to deal with content generated on-the-fly and not statically stored in the memory.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx (see page 186), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemoryEx (see page 181)

Example

This example allocates CdvdBurnerGrabber object, reads ISO9660 or Joliet file system image to the memory, records it to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PUCHAR l__PUCHAR__MemoryRegion;
LARGE_INTEGER l__LARGE_INTEGER__SizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare file system image or file and keep in memory. Pointer would be
// stored in l__UCHAR__MemoryRegion and memory region size would be stored in
// l__LARGE_INTEGER__SizeInUCHARs variable

// Try to record the ISO9660 or Joliet file system image in Track-At-Once mode
```

```
  l__EXCEPTION_NUMBER =
  StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemory(
      l__PVOID__CdvdBurnerGrabber,
      l__CHAR__ExceptionText,
      sizeof( l__CHAR__ExceptionText ),
      &l__ULONG__SystemError,
      &l__CDB_FAILURE_INFORMATION,
      ( PVOID )( l__PUCHAR__MemoryRegion ),
      l__LARGE_INTEGER__SizeInUCHARs,
      FALSE,
      FALSE,
      FALSE,
      WRITE_REPORT_DELAY_IN_SECONDS,
      BUFFER_STATUS_REPORT_DELAY_IN_SECONDS,
      FALSE // We burn data compilation
      );

  // Check for correct reply
  if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
  {
  // Handle error here...
  }

  // Do something with CdvdBurnerGrabber device object here...

  // Destroy the CdvdBurnerGrabber
  StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

  // Just check for pointer (paranoid?)
  if ( l__PVOID__CdvdBurnerGrabber != NULL )
  {
  // Handle error here...
  }
```

2.1.89 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemoryEx Function

C++

```
  __stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
  StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemoryEx(
      IN PVOID p__PVOID__CdvdBurnerGrabber,
      OUT PCHAR p__PCHAR__ExceptionText,
      IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
      OUT PULONG p__PULONG__SystemError,
      OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
      IN OUT PVOID p__PVOID__MemoryRegion,
      IN LARGE_INTEGER p__LARGE_INTEGER__MemoryRegionSizeInUCHARs,
      IN BOOLEAN p__BOOLEAN__IsXA,
      IN BOOLEAN p__BOOLEAN__IsTestWrite,
      IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
      IN ULONG p__ULONG__WriteReportDelayInSeconds,
      IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
      IN BOOLEAN p__BOOLEAN__IsAudio,
      IN USHORT p__USHORT__AudioPauseLengthInLBs,
      IN BOOLEAN p__BOOLEAN__IsAudioPauseChange
  );
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN OUT PVOID p__PVOID__MemoryRegion	Pointer to memory region containing data.
IN LARGE_INTEGER p__LARGE_INTEGER__MemoryRegionSizeInUCHARs	Memory region size in UCHARs.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
IN BOOLEAN p__BOOLEAN__IsAudio	Is this audio stream (TRUE) or data stream (FALSE).
IN USHORT p__USHORT__AudioPauseLengthInLBs	Audio pause length in logical blocks.
IN BOOLEAN p__BOOLEAN__IsAudioPauseChange	Is audio pause change.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image or audio stream located in a memory with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode. If audio content is recorded this API call can change default Track-At-Once track-to-track pause length from 150 logical blocks (2 seconds) to any passed one.

Remarks

Please see the TrackAtOnceFromPipe and TrackAtOnceFromPipeEx API calls to find out how to deal with content generated on-the-fly and not statically stored in the memory.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx (see page 186), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemory (see page 179)

Example

This example allocates CdvdBurnerGrabber object, reads audio track content to the memory, records it to the CD recordable media and destroys the device object after it's not needed any more. Also it sets track-to-track pause to zero.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber = NULL;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError = ERROR_SUCCESS;
CHAR l__CHAR__ExceptionText[ 1024 ];
```

```
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PUCHAR l__PUCHAR__MemoryRegion = NULL;
LARGE_INTEGER l__LARGE_INTEGER__SizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare audio track and store it to the memory. Pointer would be
// stored in l__UCHAR__MemoryRegion and memory region size would be stored in
// l__LARGE_INTEGER__SizeInUCHARs variable

// Try to record the audio track in Track-At-Once mode and set zero-sized track-to-track
pause
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemoryEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PVOID )( l__PUCHAR__MemoryRegion ),
    l__LARGE_INTEGER__SizeInUCHARs,
    FALSE,
    FALSE,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS,
    TRUE,   // We burn audio compilation
    0,      // Track-to-track pause length of ZERO
    TRUE    // Yes, please alter default pause length
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
```

```
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.90 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN HANDLE p__HANDLE,
    IN LARGE_INTEGER p__LARGE_INTEGER__SizeInUCHARs,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN HANDLE p__HANDLE	Handle to ISO9660 or Joliet file system image or sound file located in a pipe.
IN LARGE_INTEGER p__LARGE_INTEGER__SizeInUCHARs	Pipe contents size in UCHARs.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a pipe with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the TrackAtOnceFromPipeEx API call as it allows to burn not only data but also audio compilations from pipes.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx (see page 186)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
HANDLE l__HANDLE__Pipe = INVALID_HANDLE_VALUE;
LARGE_INTEGER l__LARGE_INTEGER__SizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare file system image or file and keep in pipe

// Try to record the ISO9660 or Joliet file system image in Track-At-Once mode
l__EXCEPTION_NUMBER =
```

```
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__HANDLE__Pipe,
    l__LARGE_INTEGER__SizeInUCHARs,
    FALSE,
    FALSE,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.91 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN HANDLE p__HANDLE,
    IN LARGE_INTEGER p__LARGE_INTEGER__SizeInUCHARs,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
    IN BOOLEAN p__BOOLEAN__IsAudio
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.

OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN HANDLE p__HANDLE	Handle to ISO9660 or Joliet file system image or sound file located in a pipe.
IN LARGE_INTEGER p__LARGE_INTEGER__SizeInUCHARs	Pipe contents size in UCHARs.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
IN BOOLEAN p__BOOLEAN__IsAudio	Is this audio stream (TRUE) or data stream (FALSE).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image or audio stream located in a pipe with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the TrackAtOnceFromPipe API call as it's simplified variant of extended call allowing to burn data compilations only.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
HANDLE l__HANDLE__Pipe = INVALID_HANDLE_VALUE;
LARGE_INTEGER l__LARGE_INTEGER__SizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );
```

```
// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare file system image or file and keep in pipe

// Try to record the ISO9660 or Joliet file system image in Track-At-Once mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__HANDLE__Pipe,
    l__LARGE_INTEGER__SizeInUCHARs,
    FALSE,
    FALSE,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS,
    FALSE // We burn data compilation
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.92 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PVOID p__PVOID__FileTree,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PVOID p__PVOID__FileTree	Pointer to ISO9660 or Joliet file system image located in a file tree.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is a just ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the TrackAtOnceFromTree sample that will demonstrate how "virtual" ISO9660 or Joliet file system image can be

burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example allocates CdvdBurnerGrabber object, records the ISO9660 or Joliet file system image from file tree to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__FileTree;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to create ISO9660 or Joliet file system image with help of
StarBurn_ISO9660JolietFileTree_Create and
// StarBurn_ISO9660JolietFileTree_BuildImage, l__PVOID__FileTree used as the pointer to the
object...

// Try to record the ISO9660 or Joliet file system image in Track-At-Once mode
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__PVOID__FileTree,
```

```
    FALSE,
    FALSE,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.93 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTreeEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTreeEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PVOID p__PVOID__FileTree,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds,
    IN ULONG p__ULONG__Custom_NWA
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PVOID p__PVOID__FileTree	Pointer to ISO9660 or Joliet file system image located in a file tree.
IN BOOLEAN p__BOOLEAN__IsXA	BOOLEAN set to TRUE if this is CDROM XA, FALSE if this is a just ordinary CDROM/CDDA.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.

IN BOOLEAN p__BOOLEAN__IsNextSessionAllowed	BOOLEAN set to TRUE is next session is allowed on this media, FALSE if next session will not be allowed on this media.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
IN ULONG p__ULONG__Custom_NWA	Next Writable Address value.
p__ULONG__Custom_Padding_Size	Custom padding size value.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the TrackAtOnceFromTree sample that will demonstrate how "virtual" ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189).

2.1.94 StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCALLBACK p__PCALLBACK__Callback,
    IN PVOID p__PVOID__CallbackContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	CdvdBurnerGrabber object.
p__PCALLBACK_Callback	Pointer to callback function that receives information about found files.
p__PVOID_CallbackContext	Callback context

Returns

Execution status.

Description

This function parses the UDF file system on disc and calls a callback function for each found file. The information about

LBAs that file resides on is passed to callback function.

Remarks

Works correctly on pre-recorded media (DVD-ROM, BD-ROM). Information that is passed to callback function should be copied and stored on client side. Strings may not be valid outside callback function.

Example

Please see UDFLookup sample as example how to use StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup API call.

2.1.95 StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCALLBACK p__PCALLBACK__Callback,
    IN PVOID p__PVOID__CallbackContext,
    IN ULONG p__ULONG__StartLBA,
    IN ULONG p__ULONG__EndLBA
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	CdvdBurnerGrabber object.
p__PCALLBACK_Callback	Pointer to callback function that receives information about found files.
p__PVOID_CallbackContext	Callback context

Returns

Execution status.

Description

This function parses the UDF file system on disc and calls a callback function for each found file and directory. The information about LBAs that file resides on is passed to callback function.

Remarks

Works correctly on pre-recorded media (DVD-ROM, BD-ROM). Information that is passed to callback function should be copied and stored on client side. Strings may not be valid outside callback function.

Example

Please see UDFLookup sample as example how to use StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) API call.

2.1.96 StarBurn_CdvdBurnerGrabber_VerifyFile Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VerifyFile(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__ImageFileName,
    IN LONG p__LONG__StartingLBA,
    OUT PLONG p__PLONG__FailedLBA,
    IN ULONG p__ULONG__ReportDelayInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__ImageFileName	Pointer to image file name we'll open and read from to compare the data with recorded disc.
IN LONG p__LONG__StartingLBA	LBA (logical block address) used to start recording from.
OUT PLONG p__PLONG__FailedLBA	Pointer to the variable to contain failed LBA (number of sector failed verification process).
IN ULONG p__ULONG__ReportDelayInLBs	Number of logical blocks verified before progress callback would be called.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function verifies recorded file system image on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular image recording (or maybe other image). This information can be used to be sure recorded disc is readable and really contains recorded information.

Remarks

Please see the TrackAtOnceFromFile and VerifyFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_VerifyFile can be used for verification of the recorded content.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_VerifyTreeEx (see page 201), StarBurn_CdvdBurnerGrabber_VerifyTree (see page 199)

Example

This example allocates CdvdBurnerGrabber object, burns image "IMAGE.ISO", verifies it and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
LONG l__LONG__FailedLBA;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Burn prepared file system image "IMAGE.ISO" here

// Try to verify file image
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_VerifyFile(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "IMAGE.ISO",
    0, // Assume we've started to burn from LBA 0
    &l__LONG__FailedLBA,
    1000 // Call progress callback after every 1000 logical blocks processed
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...
```

```
// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.97 StarBurn_CdvdBurnerGrabber_VerifyFileEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VerifyFileEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__ImageFileName,
    IN LONG p__LONG__StartingLBA,
    OUT PLONG p__PLONG__FailedLBA,
    IN ULONG p__ULONG__ReportDelayInLBs,
    IN LONG p__LONG__NumberOfRetries
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__ImageFileName	Pointer to image file name we'll open and read from to compare the data with recorded disc.
IN LONG p__LONG__StartingLBA	LBA (logical block address) used to start recording from.
OUT PLONG p__PLONG__FailedLBA	Pointer to the variable to contain failed LBA (number of sector failed verification process).
IN ULONG p__ULONG__ReportDelayInLBs	Number of logical blocks verified before progress callback would be called.
IN LONG p__LONG__NumberOfRetries	Number of read retries allowed to do on logical block failed to be readen from optical media.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function verifies recorded file system image on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular image recording (or maybe other image). This information can be used to be sure recorded disc is readable and really contains recorded information.

Remarks

Please see the TrackAtOnceFromFile and VerifyFile samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_VerifyFile (see page 194) can be used for verification of the recorded content.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_VerifyTreeEx (see page 201), StarBurn_CdvdBurnerGrabber_VerifyTree (see page 199), StarBurn_CdvdBurnerGrabber_VerifyFile (see page 194)

Example

This example allocates CdvdBurnerGrabber object, burns image "IMAGE.ISO", verifies it and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
LONG l__LONG__FailedLBA;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Burn prepared file system image "IMAGE.ISO" here

// Try to verify file image
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_VerifyFileEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
```

```
    "IMAGE.ISO",
    0, // Assume we've started to burn from LBA 0
    &l__LONG__FailedLBA,
    1000, // Call progress callback after every 1000 logical blocks processed
    10 // Allow up to 10 attempts to read failed logical block
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.98 StarBurn_CdvdBurnerGrabber_VerifyFileExUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_CdvdBurnerGrabber_VerifyFileExUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__ImageFileName,
    IN LONG p__LONG__StartingLBA,
    OUT PLONG p__PLONG__FailedLBA,
    IN ULONG p__ULONG__ReportDelayInLBs,
    IN LONG p__LONG__NumberOfRetries
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_VerifyFileExUnicode.

2.1.99 StarBurn_CdvdBurnerGrabber_VerifyFileUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VerifyFileUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__ImageFileName,
```

```
    IN LONG p__LONG__StartingLBA,
    OUT PLONG p__PLONG__FailedLBA,
    IN ULONG p__ULONG__ReportDelayInLBs
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_VerifyFileUnicode.

2.1.100 StarBurn_CdvdBurnerGrabber_VerifyTree Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VerifyTree(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PVOID p__PVOID__FileTree,
    IN LONG p__LONG__StartingLBA,
    OUT PLONG p__PLONG__FailedLBA,
    IN ULONG p__ULONG__ReportDelayInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PVOID p__PVOID__FileTree	Pointer to file system tree used for recording.
IN LONG p__LONG__StartingLBA	LBA (logical block address) used to start recording from.
OUT PLONG p__PLONG__FailedLBA	Pointer to the variable to contain failed LBA (number of sector failed verification process).
IN ULONG p__ULONG__ReportDelayInLBs	Number of logical blocks verified before progress callback would be called.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function verifies recorded file system tree on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular tree recording. This information can be used to be sure recorded disc is readable and really contains recorded information.

Remarks

Please see the TrackAtOnceFromTree and DiscAtOnceFromTree samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_VerifyTree can

be used for verification of the recorded content. Also it's a good idea to use extended variant of this call StarBurn_CdvdBurnerGrabber_VerifyTreeEx (see page 201) when there's need to know how many retries for single failed logical block could be allowed.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_VerifyTreeEx (see page 201), StarBurn_CdvdBurnerGrabber_VerifyFile (see page 194)

Example

This example allocates CdvdBurnerGrabber object, burns file tree, verifies it and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__FileTree;
LONG l__LONG__FailedLBA;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Create and burn ISO9660 or Joliet file tree here

// Try to verify file tree
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_VerifyTree(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__PVOID__FileTree,
    0, // Assume we've started to burn from LBA 0
    &l__LONG__FailedLBA,
```

```
    1000 // Call progress callback after every 1000 logical blocks processed
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.101 StarBurn_CdvdBurnerGrabber_VerifyTreeEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VerifyTreeEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PVOID p__PVOID__FileTree,
    IN LONG p__LONG__StartingLBA,
    OUT PLONG p__PLONG__FailedLBA,
    IN ULONG p__ULONG__ReportDelayInLBs,
    IN LONG p__LONG__NumberOfRetries
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PVOID p__PVOID__FileTree	Pointer to file system tree used for recording.
IN LONG p__LONG__StartingLBA	LBA (logical block address) used to start recording from.
OUT PLONG p__PLONG__FailedLBA	Pointer to the variable to contain failed LBA (number of sector failed verification process).
IN ULONG p__ULONG__ReportDelayInLBs	Number of logical blocks verified before progress callback would be called.
IN LONG p__LONG__NumberOfRetries	Number of read retries allowed to do on logical block failed to be readen from optical media.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with

appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function verifies recorded file system tree on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular tree recording. This information can be used to be sure recorded disc is readable and really contains recorded information.

Remarks

Please see the TrackAtOnceFromTree and DiscAtOnceFromTree samples that will demonstrate how ISO9660 or Joliet file system image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and how StarBurn_CdvdBurnerGrabber_VerifyTree (see page 199) can be used for verification of the recorded content (extended variant of the call is basically the same except it allows to control number of read retries). Also it's a good idea to use simplified variant of this call StarBurn_CdvdBurnerGrabber_VerifyTree (see page 199) when there's no need to know how many retries for single failed logical block could be allowed.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_VerifyTree (see page 199), StarBurn_CdvdBurnerGrabber_VerifyFile (see page 194)

Example

This example allocates CdvdBurnerGrabber object, burns file tree, verifies it and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__FileTree;
LONG l__LONG__FailedLBA;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

```
// Create and burn ISO9660 or Joliet file tree here

// Try to verify file tree
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_VerifyTreeEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    l__PVOID__FileTree,
    0, // Assume we've started to burn from LBA 0
    &l__LONG__FailedLBA,
    1000 // Call progress callback after every 1000 logical blocks processed,
    10 // Allow up to 10 attempts to read failed logical block
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.102 StarBurn_CdvdBurnerGrabber_VideoCD Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VideoCD(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__MPEG1ImagePathAndFileName,
    IN PCHAR p__PCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.

IN PCHAR p__PCHAR__MPEG1ImagePathAndFileName	Pointer to VCD/MPEG1 image file located in a file and stored on the hard disk.
IN PCHAR p__PCHAR__AlbumIdentifier	Pointer to album identifier.
IN USHORT p__USHORT__NumberOfVolumesInAlbum	Number of volumes in album.
IN USHORT p__USHORT__AlbumSetSequenceNumber	Album set sequence number.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.

Remarks

Please see the VideoCD sample that will demonstrate how VCD/MPEG1 image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_VideoCD with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157)

Example

This example allocates CdvdBurnerGrabber object, records the VCD/MPEG1 image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
```

```
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the VCD/MPEG1 image (in Track-At-Once mode)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_VideoCD(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    "C:\MOVIE.MPEG1",
    "Album",
    0x0001,
    0x0001,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.103 StarBurn_CdvdBurnerGrabber_VideoCDEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VideoCDEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN WRITE_MODE p__WRITE_MODE,
    IN PCHAR p__PCHAR__MPEG1ImagePathAndFileName,
    IN PCHAR p__PCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
```

```
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN WRITE_MODE p__WRITE_MODE	Write mode to use.
IN PCHAR p__PCHAR__MPEG1ImagePathAndFileName	Pointer to VCD/MPEG1 image file located in a file and stored on the hard disk.
IN PCHAR p__PCHAR__AlbumIdentifier	Pointer to album identifier.
IN USHORT p__USHORT__NumberOfVolumesInAlbum	Number of volumes in album.
IN USHORT p__USHORT__AlbumSetSequenceNumber	Album set sequence number.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.

Remarks

Please see the VideoCDEx sample that will demonstrate how VCD/MPEG1 image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_VideoCDEx with currently set write speed and currently set optimum power calibration.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157), StarBurn_CdvdBurnerGrabber_VideoCD (see page 203), StarBurn_CdvdBurnerGrabber_SuperVideoCDEx (see page 159)

Example

This example allocates CdvdBurnerGrabber object, records the VCD/MPEG1 image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
```

```
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the VCD/MPEG1 image (in Session-At-Once mode)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_VideoCDEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    WRITE_MODE_SESSION_AT_ONCE,
    "C:\MOVIE.MPEG1",
    "Album",
    0x0001,
    0x0001,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.104 StarBurn_CdvdBurnerGrabber_VideoCDExEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VideoCDExEx(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PCHAR p__PCHAR__MPEG1ImagePathAndFileName,
    IN PCHAR p__PCHAR__TemplateImagePathAndFileName,
    IN PCHAR p__PCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION	Pointer to CDB_FAILURE_INFORMATION (see page 480) that will be filled with appropriate values.
IN PCHAR p__PCHAR__MPEG1ImagePathAndFileName	Pointer to VCD/MPEG1 image file located in a file and stored on the hard disk.
IN PCHAR p__PCHAR__TemplateImagePathAndFileName	Pointer to template file name StarBurn would record as first data track.
IN PCHAR p__PCHAR__AlbumIdentifier	Pointer to album identifier.
IN USHORT p__USHORT__NumberOfVolumesInAlbum	Number of volumes in album.
IN USHORT p__USHORT__AlbumSetSequenceNumber	Album set sequence number.
IN BOOLEAN p__BOOLEAN__IsTestWrite	BOOLEAN set to TRUE if this is test write, FALSE if this is a real write.
IN ULONG p__ULONG__WriteReportDelayInSeconds	Write report delay in seconds (time between 2 WRITE_PACKET callbacks).
IN ULONG p__ULONG__BufferStatusReportDelayInSeconds	Buffer status report delay in seconds (time between 2 BUFFER_STATUS callbacks).
p__WRITE_MODE	Write mode to use.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.

Remarks

Please see the VideoCDEx sample that will demonstrate how VCD/MPEG1 image can be burn on the CD/DVD/Blu-Ray/HD-DVD media and with the help of StarBurn_CdvdBurnerGrabber_VideoCDEx (see page 205) with

currently set write speed and currently set optimum power calibration. This sample is for legacy call, StarBurn_CdvdBurnerGrabber_VideoCDExEx is basically the same API call except it allows to pass user-defined data track StarBurn would record to VCD as first track (MPEG1 file is second track on VCD).

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153), StarBurn_CdvdBurnerGrabber_SendOPC (see page 139), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157), StarBurn_CdvdBurnerGrabber_VideoCD (see page 203), StarBurn_CdvdBurnerGrabber_SuperVideoCDEx (see page 159), StarBurn_CdvdBurnerGrabber_SuperVideoCDExEx (see page 162), StarBurn_CdvdBurnerGrabber_VideoCDEx (see page 205)

Example

This example allocates CdvdBurnerGrabber object, records the VCD/MPEG1 image to the CD/DVD/Blu-Ray/HD-DVD media and destroys the device object after it's not needed any more. Custom user file name is passed to have own data track on resulting VCD.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to record the VCD/MPEG1 image (in Session-At-Once mode)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_VideoCDExEx(
    l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
```

```
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    WRITE_MODE_SESSION_AT_ONCE,
    "C:\MOVIE.MPEG1",  // Second MPEG1 track
    "C:\DATA.ISO",     // First DATA track
    "Album",
    0x0001,
    0x0001,
    FALSE,
    WRITE_REPORT_DELAY_IN_SECONDS,
    BUFFER_STATUS_REPORT_DELAY_IN_SECONDS
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.105 StarBurn_CdvdBurnerGrabber_VideoCDExExUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VideoCDExExUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__MPEG1ImagePathAndFileName,
    IN PWCHAR p__PWCHAR__TemplateImagePathAndFileName,
    IN PWCHAR p__PWCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_VideoCDExExUnicode.

2.1.106 StarBurn_CdvdBurnerGrabber_VideoCDExUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VideoCDExUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN WRITE_MODE p__WRITE_MODE,
    IN PWCHAR p__PWCHAR__MPEG1ImagePathAndFileName,
    IN PWCHAR p__PWCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_VideoCDExUnicode.

2.1.107 StarBurn_CdvdBurnerGrabber_VideoCDUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_CdvdBurnerGrabber_VideoCDUnicode(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    OUT PCDB_FAILURE_INFORMATION p__PCDB_FAILURE_INFORMATION,
    IN PWCHAR p__PWCHAR__MPEG1ImagePathAndFileName,
    IN PWCHAR p__PWCHAR__AlbumIdentifier,
    IN USHORT p__USHORT__NumberOfVolumesInAlbum,
    IN USHORT p__USHORT__AlbumSetSequenceNumber,
    IN BOOLEAN p__BOOLEAN__IsTestWrite,
    IN ULONG p__ULONG__WriteReportDelayInSeconds,
    IN ULONG p__ULONG__BufferStatusReportDelayInSeconds
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_CdvdBurnerGrabber_VideoCDUnicode.

2.1.108 StarBurn_CorrectISO9660Name_Default Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CorrectISO9660Name_Default(
    IN OUT PCHAR p__PCHAR__Name
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN OUT PCHAR p__PCHAR__Name	Pointer to current name.

Returns

Nothing.

Description

This function generates next ISO9660 file name to make it unique.

Remarks

N/A

See Also

StarBurn_CorrectJolietName_Default (see page 212)

Example

N/A

2.1.109 StarBurn_CorrectJolietName_Default Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_CorrectJolietName_Default(
    IN OUT PWCHAR p__PWCHAR__Name
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN OUT PWCHAR p__PWCHAR__Name	Pointer to current name.

Returns

Nothing.

Description

This function generates next Joliet file name to make it unique.

Remarks

N/A

See Also

StarBurn_CorrectISO9660Name_Default (see page 212)

Example

N/A

2.1.110 StarBurn_CreatePipe Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_CreatePipe(
    OUT PHANDLE p__PHANDLE__ReadPipe,
    OUT PHANDLE p__PHANDLE__WritePipe
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PHANDLE p__PHANDLE__ReadPipe	pointer to the output "read" pipe handle.
OUT PHANDLE p__PHANDLE__WritePipe	pointer to the output "write" pipe handle.

Returns

Win32 execution status.

Description

This function creates pipe for all of the StarBurn pipe operations and returns both "read" and "write" handles for it.

Remarks

Used with all of the StarBurn pipe-related API calls.

See Also

StarBurn_DestroyPipe (see page 215), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx (see page 186)

Example

This example allocates both "read" and "write" handles for the same pipe, does some I/O and closes them both later.

```
// Somewhere in the data region
HANDLE l__HANDLE__ReadPipe = INVALID_HANDLE_VALUE;
HANDLE l__HANDLE__WritePipe = INVALID_HANDLE_VALUE;
ULONG l__ULONG__Status = ERROR_SUCCESS;

// Try to create pipe
l__ULONG__Status =
StarBurn_CreatePipe(
    &l__HANDLE__ReadPipe,
    &l__HANDLE__WritePipe
    );

// Check for success
if ( l__ULONG__Status != ERROR_SUCCESS )
{
// Handle error here...
}

// Do something with the pipe handles here...

// Close both "read" and "write" pipe handles
StarBurn_DestroyPipe(
```

```
    &l__HANDLE__ReadPipe,
    &l__HANDLE__WritePipe
    );
```

2.1.111 StarBurn_Destroy Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_Destroy(
    IN OUT PVOID * p__PPVOID__StarBurnObject
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN OUT PVOID * p__PPVOID__StarBurnObject`	Pointer to pointer to the object that toolkit allocated before.

Returns

None.

Description

This function frees allocated memory in the way of destroying the object that toolkit created internally. This is universal call, it frees all the objects and does not care about object type.

Remarks

This call does not check for passed parameter validness. It's up to user to provide the toolkit with correct pointers to objects. After return from this call the object pointer will be NULL.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_CdvdBurnerGrabber_Create (see page 31)

Example

This example allocates Joliet file tree and destroys it after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    TRUE,
    FILE_TREE_JOLIET
    );
```

```
// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.112 StarBurn_DestroyPipe Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_DestroyPipe(
    IN PHANDLE p__PHANDLE__ReadPipe,
    IN PHANDLE p__PHANDLE__WritePipe
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PHANDLE p__PHANDLE__ReadPipe	pointer to the input "read" pipe handle.
IN PHANDLE p__PHANDLE__WritePipe	pointer to the input "write" pipe handle.

Returns

Nothing.

Description

This function closes both "read" and "write" pipe handles.

Remarks

Used with all of the StarBurn pipe-related API calls.

See Also

StarBurn_CreatePipe (see page 213), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx (see page 186)

Example

This example allocates both "read" and "write" handles for the same pipe, does some I/O and closes them both later.

```
// Somewhere in the data region
HANDLE l__HANDLE__ReadPipe = INVALID_HANDLE_VALUE;
HANDLE l__HANDLE__WritePipe = INVALID_HANDLE_VALUE;
ULONG l__ULONG__Status = ERROR_SUCCESS;

// Try to create pipe
l__ULONG__Status =
StarBurn_CreatePipe(
    &l__HANDLE__ReadPipe,
    &l__HANDLE__WritePipe
    );

// Check for success
if ( l__ULONG__Status != ERROR_SUCCESS )
```

```
{
// Handle error here...
}

// Do something with the pipe handles here...

// Close both "read" and "write" pipe handles
StarBurn_DestroyPipe(
    &l__HANDLE__ReadPipe,
    &l__HANDLE__WritePipe
    );
```

2.1.113 StarBurn_DownShut Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_DownShut(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Execution status. EN_SUCCESS if uninitialization process completed successfully.

Description

This function uninitialize burning toolkit. It's expected to be called after very last function call before exiting from StarBurn client application. It's a good idea to gather the trash before exiting. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used.

Remarks

See samples for details how and when call StarBurn_DownShut() function.

See Also

See samples for details how and when call StarBurn_DownShut() function.

Example

This example just uninitialize toolkit.

```
// Somewhere in the data region
EXCEPTION_NUMBER l__EXCEPTION_NUMBER = EN_SUCCESS;

// Try to uninitialize toolkit
l__EXCEPTION_NUMBER =
StarBurn_DownShut();

// Check for success
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

2.1.114 StarBurn_DVDVideo_Create Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_DVDVideo_Create(
    OUT PVOID * p__PPVOID__DVDVideo,
    IN PCHAR p__PCHAR__VideoTsDirectory,
    IN BOOLEAN p__BOOLEAN__IsPatchingEnabled,
```

```
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCHAR p__PCHAR__VolumeLabel,
    IN PCHAR p__PCHAR__PublisherPreparerName,
    IN PCHAR p__PCHAR__ApplicationName,
    IN LONG p__LONG__Year,
    IN LONG p__LONG__Month,
    IN LONG p__LONG__Day,
    IN LONG p__LONG__Hour,
    IN LONG p__LONG__Minute,
    IN LONG p__LONG__Second,
    IN LONG p__LONG__MilliSecond
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID * p__PPVOID__DVDVideo	Pointer to pointer to resulting DVD-Video file system object.
IN PCHAR p__PCHAR__VideoTsDirectory	Pointer to VIDEO_TS directory with corresponding DVD files.
IN BOOLEAN p__BOOLEAN__IsPatchingEnabled	Should be set to TRUE to allow StarBurn core to modify IFO/BUP files and FALSE otherwise.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to buffer to receive formatted error message in case of exception.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Error text buffer size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error value (if function would return EN_SYSTEM_CALL_FAILED).
IN PCHAR p__PCHAR__VolumeLabel	Pointer to volume label resulting DVD-Video compilation would receive.
IN PCHAR p__PCHAR__PublisherPreparerName	Pointer to publisher and preparer name resulting DVD-Video compilation would receive.
IN PCHAR p__PCHAR__ApplicationName	Pointer to application name resulting DVD-Video compilation would receive.
IN LONG p__LONG__Year	Year which would be set as DVD-Video compilation creation year.
IN LONG p__LONG__Month	Month which would be set as DVD-Video compilation creation month.
IN LONG p__LONG__Day	Day which would be set as DVD-Video compilation creation day.
IN LONG p__LONG__Hour	Hour which would be set as DVD-Video compilation creation hour.
IN LONG p__LONG__Minute	Minute which would be set as DVD-Video compilation creation minute.
IN LONG p__LONG__Second	Second which would be set as DVD-Video compilation creation second.
IN LONG p__LONG__MilliSecond	Millisecond which would be set as DVD-Video compilation creation millisecond.

Returns

Execution status.

Description

This function creates DVD-Video file system object from passed pointer to VIDEO_TS directory with DVD corresponding files.

Remarks

None.

See Also

StarBurn_DVDVideo_Destroy (see page 218), StarBurn_DVDVideo_SeekToBegin (see page 222), StarBurn_DVDVideo_Read (see page 221), StarBurn_DVDVideo_GetSizeInUCHARs (see page 219), StarBurn_DVDVideo_GetTreePointer (see page 219)

Example

See DVDVideoTrackAtOnceFromFileEx and DVDVideoBuildImageEx StarBurn sample applications as an example how to use StarBurn_DVDVideo_Create API call and all of the companion DVD-Video sub-API calls.

2.1.115 StarBurn_DVDVideo_CreateUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_DVDVideo_CreateUnicode(
    OUT PVOID * p__PPVOID__DVDVideo,
    IN CONST WCHAR* p__PWCHAR__VideoTsDirectory,
    IN BOOLEAN p__BOOLEAN__IsPatchingEnabled,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN CONST WCHAR* p__PWCHAR__VolumeLabel,
    IN CONST WCHAR* p__PWCHAR__PublisherPreparerName,
    IN CONST WCHAR* p__PWCHAR__ApplicationName,
    IN LONG p__LONG__Year,
    IN LONG p__LONG__Month,
    IN LONG p__LONG__Day,
    IN LONG p__LONG__Hour,
    IN LONG p__LONG__Minute,
    IN LONG p__LONG__Second,
    IN LONG p__LONG__MilliSecond
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_DVDVideo_CreateUnicode.

2.1.116 StarBurn_DVDVideo_Destroy Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_DVDVideo_Destroy(
    IN PVOID p__PVOID__DVDVideo
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PVOID p__PVOID__DVDVideo`	Pointer to source DVD-Video file system object we need to deallocate.

Returns

Nothing.

Description

This function destroys DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.

Remarks

None.

See Also

StarBurn_DVDVideo_Create (see page 216), StarBurn_DVDVideo_SeekToBegin (see page 222), StarBurn_DVDVideo_Read (see page 221), StarBurn_DVDVideo_GetSizeInUCHARs (see page 219), StarBurn_DVDVideo_GetTreePointer (see page 219)

Example

See DVDVideoTrackAtOnceFromFileEx and DVDVideoBuildImageEx StarBurn sample applications as an example how to use StarBurn_DVDVideo_Destroy API call and all of the companion DVD-Video sub-API calls.

2.1.117 StarBurn_DVDVideo_GetSizeInUCHARs Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_DVDVideo_GetSizeInUCHARs(
    IN PVOID p__PVOID__DVDVideo,
    OUT PLARGE_INTEGER p__PLARGE_INTEGER__SizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PVOID p__PVOID__DVDVideo`	Pointer to source DVD-Video file system object we need to get payload size in UCHARs.
`OUT PLARGE_INTEGER p__PLARGE_INTEGER__SizeInUCHARs`	Pointer to the variable to receive file system payload size in UCHARs.

Returns

Execution status.

Description

This function gets file system content size in UCHARs from DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.

Remarks

None.

See Also

StarBurn_DVDVideo_Create (see page 216), StarBurn_DVDVideo_Destroy (see page 218), StarBurn_DVDVideo_SeekToBegin (see page 222), StarBurn_DVDVideo_Read (see page 221), StarBurn_DVDVideo_GetTreePointer (see page 219)

Example

See DVDVideoTrackAtOnceFromFileEx and DVDVideoBuildImageEx StarBurn sample applications as an example how to use StarBurn_DVDVideo_GetSizeInUCHARs API call and all of the companion DVD-Video sub-API calls.

2.1.118 StarBurn_DVDVideo_GetTreePointer Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_DVDVideo_GetTreePointer(
    IN PVOID p__PVOID__DVDVideo,
    OUT PVOID * p__PPVOID__ISO9660JolietFileTree
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__DVDVideo	Pointer to source DVD-Video file system object we need to get ISO9660 file tree object pointer.
OUT PVOID * p__PPVOID__ISO9660JolietFileTree	Pointer to the variable to receive ISO9660 file tree object pointer.

Returns

Execution status.

Description

This function returns pointer to ISO9660 file tree object embedded to DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.

Remarks

None.

See Also

StarBurn_DVDVideo_Create (see page 216), StarBurn_DVDVideo_Destroy (see page 218), StarBurn_DVDVideo_SeekToBegin (see page 222), StarBurn_DVDVideo_Read (see page 221), StarBurn_DVDVideo_GetSizeInUCHARs (see page 219)

Example

See DVDVideoTrackAtOnceFromFileEx and DVDVideoBuildImageEx StarBurn sample applications as an example how to use StarBurn_DVDVideo_GetTreePointer API call and all of the companion DVD-Video sub-API calls.

2.1.119 StarBurn_DVDVideo_PatchHeader Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_DVDVideo_PatchHeader(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM,
    IN ULONG p__ULONG__BUPLastTouchedRBA,
    IN BOOLEAN p__BOOLEAN__IsMenuVob
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM	Pointer to tree item to process.
IN ULONG p__ULONG__BUPLastTouchedRBA	BUF last touched RBA.
IN BOOLEAN p__BOOLEAN__IsMenuVob	Is menu VOB file present.

Returns

Nothing.

Description

This function patches IFO/BUF file header RBA for BUP.

Example

See the DVDVideoBuildImage example.

2.1.120 StarBurn_DVDVideo_PatchTable Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_DVDVideo_PatchTable(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM,
    IN PDVD_VIDEO_CONTROL_BLOCK p__PDVD_VIDEO_CONTROL_BLOCK
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM	Pointer to tree item to process.
IN PDVD_VIDEO_CONTROL_BLOCK p__PDVD_VIDEO_CONTROL_BLOCK	Pointer to DVD-Video control block.

Returns

Nothing.

Description

This function patches IFO/BUF file table RBAs.

Example

See the DVDVideoBuildImage example.

2.1.121 StarBurn_DVDVideo_Read Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_DVDVideo_Read(
    IN PVOID p__PVOID__DVDVideo,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN LARGE_INTEGER p__LARGE_INTEGER__IoTransferSizeInUCHARs,
    OUT PUCHAR p__PUCHAR__DataBuffer
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__DVDVideo	Pointer to source DVD-Video file system object we need to read data from.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to buffer to receive formatted error message in case of exception.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Error text buffer size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error value (if function would return EN_SYSTEM_CALL_FAILED).
IN LARGE_INTEGER p__LARGE_INTEGER__IoTransferSizeInUCHARs	Number of UCHARs we want to read from file system object.
OUT PUCHAR p__PUCHAR__DataBuffer	Pointer to output data buffer we want file system object data to be placed after read.

Returns

Execution status.

Description

This function reads requested number of UCHARs from DVD-Video file system object (created with StarBurn_DVDVideo_Create (see page 216) API call) from it's current read position.

Remarks

None.

See Also

StarBurn_DVDVideo_Create (see page 216), StarBurn_DVDVideo_Destroy (see page 218), StarBurn_DVDVideo_SeekToBegin (see page 222), StarBurn_DVDVideo_GetSizeInUCHARs (see page 219), StarBurn_DVDVideo_GetTreePointer (see page 219)

Example

See DVDVideoTrackAtOnceFromFileEx and DVDVideoBuildImageEx StarBurn sample applications as an example how to use StarBurn_DVDVideo_Read API call and all of the companion DVD-Video sub-API calls.

2.1.122 StarBurn_DVDVideo_SeekToBegin Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_DVDVideo_SeekToBegin(
    IN PVOID p__PVOID__DVDVideo,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__DVDVideo	Pointer to source DVD-Video file system object we need to rewind.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to buffer to receive formatted error message in case of exception.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Error text buffer size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error value (if function would return EN_SYSTEM_CALL_FAILED).

Returns

Execution status.

Description

This function moves DVD-Video file system object (created with StarBurn_DVDVideo_Create (see page 216) API call) current read position to the very beginning.

Remarks

None.

See Also

StarBurn_DVDVideo_Create (see page 216), StarBurn_DVDVideo_Destroy (see page 218), StarBurn_DVDVideo_Read (see page 221), StarBurn_DVDVideo_GetSizeInUCHARs (see page 219), StarBurn_DVDVideo_GetTreePointer (see page 219)

Example

See DVDVideoTrackAtOnceFromFileEx and DVDVideoBuildImageEx StarBurn sample applications as an example how to use StarBurn_DVDVideo_SeekToBegin API call and all of the companion DVD-Video sub-API calls.

2.1.123 StarBurn_ElTorito_BootCatalogAddSection Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ElTorito_BootCatalogAddSection(
    void * BootCatalog,
    const char * SectionIdentifier,
    unsigned short BootEntriesCount,
    STARBURN_ELTORITO_PLATFORM PlatformIdentifier,
    const char ** BootImageFileNames,
    unsigned char * SystemTypes,
    STARBURN_ELTORITO_MEDIA * EmulationTypes,
    unsigned short * SectorsToLoads,
    unsigned short * LoadSegments
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * BootCatalog	Pointer to pointer to boot catalog object.
const char * SectionIdentifier	Pointer to zero-terminated string with system identifier.
unsigned short BootEntriesCount	Count of boot entries in section
STARBURN_ELTORITO_PLATFORM PlatformIdentifier	A value from STARBURN_ELTORITO_PLATFORM (see page 557) enumeration.
const char ** BootImageFileNames	Pointer to zero-terminated Unicode string with boot image file path.
unsigned char * SystemTypes	array of MBR partition type.
STARBURN_ELTORITO_MEDIA * EmulationTypes	array of STARBURN_ELTORITO_MEDIA (see page 557) enumeration values.
unsigned short * SectorsToLoads	array of Number of sectors to load from boot image.
unsigned short * LoadSegments	array of Load segment

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ElTorito boot catalog (ANSI names).

2.1.124 StarBurn_ElTorito_BootCatalogAddSectionUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ElTorito_BootCatalogAddSectionUnicode(
    void * BootCatalog,
    const unsigned short * SectionIdentifier,
    unsigned short BootEntriesCount,
    STARBURN_ELTORITO_PLATFORM PlatformIdentifier,
    const unsigned short ** BootImageFileNames,
    unsigned char * SystemTypes,
    STARBURN_ELTORITO_MEDIA * EmulationTypes,
    unsigned short * SectorsToLoads,
    unsigned short * LoadSegments
```

```
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * BootCatalog	Pointer to pointer to boot catalog object.
const unsigned short * SectionIdentifier	Pointer to zero-terminated string with system identifier.
unsigned short BootEntriesCount	Count of boot entries in section.
STARBURN_ELTORITO_PLATFORM PlatformIdentifier	A value from STARBURN_ELTORITO_PLATFORM (see page 557) enumeration.
const unsigned short ** BootImageFileNames	Pointer to zero-terminated Unicode string with boot image file path.
unsigned char * SystemTypes	array of MBR partition type.
STARBURN_ELTORITO_MEDIA * EmulationTypes	array of STARBURN_ELTORITO_MEDIA (see page 557) enumeration values.
unsigned short * SectorsToLoads	array of Number of sectors to load from boot image.
unsigned short * LoadSegments	array of Load segment

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ElTorito boot catalog (UNICODE names).

2.1.125 StarBurn_ElTorito_CreateBootCatalog Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ElTorito_CreateBootCatalog(
    void ** BootCatalog,
    const char * BootImageFileName,
    const char * ValidationIdentifier,
    STARBURN_ELTORITO_PLATFORM PlatformIdentifier,
    unsigned char SystemType,
    STARBURN_ELTORITO_MEDIA EmulationType,
    unsigned short SectorsToLoad,
    unsigned short LoadSegment
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** BootCatalog	Pointer to pointer to boot catalog object.
const char * BootImageFileName	Pointer to zero-terminated Unicode string with boot image file path.
const char * ValidationIdentifier	Pointer to zero-terminated string with system identifier.
STARBURN_ELTORITO_PLATFORM PlatformIdentifier	A value from STARBURN_ELTORITO_PLATFORM (see page 557) enumeration
unsigned char SystemType	MBR partition type.
STARBURN_ELTORITO_MEDIA EmulationType	A value from STARBURN_ELTORITO_MEDIA (see page 557) enumeration
unsigned short SectorsToLoad	Number of sectors to load from boot image.
unsigned short LoadSegment	Load segment

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ElTorito boot catalog (ANSI names).

2.1.126 StarBurn_ElTorito_CreateBootCatalogUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ElTorito_CreateBootCatalogUnicode(
    void ** BootCatalog,
    const unsigned short * BootImageFileName,
    const unsigned short * ValidationIdentifier,
    STARBURN_ELTORITO_PLATFORM PlatformIdentifier,
    unsigned char SystemType,
    STARBURN_ELTORITO_MEDIA EmulationType,
    unsigned short SectorsToLoad,
    unsigned short LoadSegment
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** BootCatalog`	Pointer to pointer to boot catalog object.
`const unsigned short * BootImageFileName`	Pointer to zero-terminated Unicode string with boot image file path.
`const unsigned short * ValidationIdentifier`	Pointer to zero-terminated string with system identifier.
`STARBURN_ELTORITO_PLATFORM PlatformIdentifier`	A value from STARBURN_ELTORITO_PLATFORM (see page 557) enumeration
`unsigned char SystemType`	MBR partition type.
`STARBURN_ELTORITO_MEDIA EmulationType`	A value from STARBURN_ELTORITO_MEDIA (see page 557) enumeration
`unsigned short SectorsToLoad`	Number of sectors to load from boot image.
`unsigned short LoadSegment`	Load segment

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ElTorito boot catalog (UNICODE names).

2.1.127 StarBurn_FileClose Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_FileClose(
    PSTARBURN_FILE_OBJECT FileObject
);
```

File

StarBurn.h (see page 662)

Returns

Nothing

Description

This fucntion closes file handle embedded to file object.

2.1.128 StarBurn_FileCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileCreate(
    PSTARBURN_FILE_OBJECT FileObject,
    const char * PathName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to preallocated file object.
const char * PathName	Pointer to file path and name.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates new file from the passed file path and name.

2.1.129 StarBurn_FileInitialize Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_FileInitialize(
    PSTARBURN_FILE_OBJECT FileObject
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to file object.

Returns

Nothing

Description

This function initializes file handle embedded to file object with bad file handle value.

2.1.130 StarBurn_FileOpen Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileOpen(
    PSTARBURN_FILE_OBJECT FileObject,
    const char * PathName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to preallocated file object.
const char * PathName	Pointer to file path and name.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function opens existing file from the passed file path and name.

2.1.131 StarBurn_FileOpenEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileOpenEx(
    PSTARBURN_FILE_OBJECT FileObject,
    const char * PathName,
    BOOL IsWrite
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to preallocated file object.
const char * PathName	Pointer to file path and name.
BOOL IsWrite	Is write access required.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function opens existing file from the passed file path and name.

2.1.132 StarBurn_FileRead Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileRead(
    PSTARBURN_FILE_OBJECT FileObject,
    void * Buffer,
    unsigned long BufferSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to file object.

void * Buffer	Pointer to data buffer to read file contents to.
unsigned long BufferSizeInUCHARs	Buffer size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function reads asked amount of unsigned chars from file at current seek position.

2.1.133 StarBurn_FileReWind Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileReWind(
    PSTARBURN_FILE_OBJECT FileObject
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to file object

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function rewinds file position in unsigned chars to the very beginning for the file.

2.1.134 StarBurn_FileSeek Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileSeek(
    PSTARBURN_FILE_OBJECT FileObject,
    unsigned __int64 PositionInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to file object.
unsigned __int64 PositionInUCHARs	Position in unsigned chars within a file.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function sets file position in unsigned chars to required one.

2.1.135 StarBurn_FileSeekRead Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileSeekRead(
    PSTARBURN_FILE_OBJECT FileObject,
    unsigned __int64 PositionInUCHARs,
    void * Buffer,
    unsigned long BufferSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to file object.
unsigned __int64 PositionInUCHARs	Position in unsigned chars within a file.
void * Buffer	Pointer to data buffer to read file contents to.
unsigned long BufferSizeInUCHARs	Buffer size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function sets file position in unsigned chars to required one and reads requested amounts unsigned chars.

2.1.136 StarBurn_FileSizeInUCHARsGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileSizeInUCHARsGet(
    PSTARBURN_FILE_OBJECT FileObject,
    unsigned __int64 * SizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to file object.
unsigned __int64 * SizeInUCHARs	Pointer to file size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function gets file size in unsigned chars.

2.1.137 StarBurn_FileUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileUnicodeCreate(
    PSTARBURN_FILE_OBJECT FileObject,
    const unsigned short * UnicodePathName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to preallocated file object.
const unsigned short * UnicodePathName	Pointer to UNICODE file path and name.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates new file from the passed UNICODE file path and name.

2.1.138 StarBurn_FileUnicodeOpen Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileUnicodeOpen(
    PSTARBURN_FILE_OBJECT FileObject,
    const unsigned short * UnicodePathName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to preallocated file object.
const unsigned short * UnicodePathName	Pointer to UNICODE file path and name.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function opens existing file from the passed UNICODE file path and name.

2.1.139 StarBurn_FileUnicodeOpenEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileUnicodeOpenEx(
    PSTARBURN_FILE_OBJECT FileObject,
    const unsigned short * UnicodePathName,
    BOOL IsWrite
```

```
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to preallocated file object.
const unsigned short * UnicodePathName	Pointer to UNICODE file path and name.
BOOL IsWrite	Is write access required.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function opens existing file from the passed UNICODE file path and name.

2.1.140 StarBurn_FileWrite Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_FileWrite(
    PSTARBURN_FILE_OBJECT FileObject,
    const void * Buffer,
    unsigned long BufferSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PSTARBURN_FILE_OBJECT FileObject	Pointer to file object.
const void * Buffer	Pointer to data buffer to write to file.
unsigned long BufferSizeInUCHARs	Buffer size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function writes asked amount of unsigned chars to file at current seek position

2.1.141 StarBurn_FindDevice Function

C++

```
__stdcall STARBURN_IMPEX_API LONG StarBurn_FindDevice(
    IN UCHAR p__UCHAR__DeviceType,
    IN BOOLEAN p__BOOLEAN__IsFindFirst,
    IN PCALLBACK p__CALLBACK,
    IN PVOID p__PVOID__CallbackContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN UCHAR p__UCHAR__DeviceType	SCSI device type to find
IN BOOLEAN p__BOOLEAN__IsFindFirst	TRUE if the function will return after locating first device, FALSE to process all the devices.
IN PVOID p__PVOID__CallbackContext	Callback context that will be passed as second parameter into call to p__PCALLBACK.
p__PCALLBACK	Callback function that will be called with FIND_DEVICE CALLBACK_NUMBER (see page 476), callback context and adapter ID and Target ID of the device.

Returns

Number of devices found. If zero - nothing was found.

Description

This function finds device of the specified device type.

Remarks

Please see the FindDevice sample that will demonstrate how to find CD/DVD/Blu-Ray/HD-DVD devices installed in the system.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480)

Example

This example finds first first CD/DVD/Blu-Ray/HD-DVD device, allocates CdvdBurnerGrabber object, and destroys the device object after it's not needed any more.

```
VOID
__stdcall
Find_Callback(
    IN CALLBACK_NUMBER p__CALLBACK_NUMBER,
    IN PVOID p__PVOID__CallbackContext,
    IN PVOID p__PVOID__CallbackSpecial1,
    IN PVOID p__PVOID__CallbackSpecial2
    )
{

PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on passed address (bus ID is always 0 in case of ASPI
and LUN cannot be probed so
// always 0 as well) with 32MB of cache)
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
```

```
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    ( UCHAR )( p__PVOID__CallbackSpecial1) ,
    0,
    ( UCHAR )( p__PVOID__CallbackSpecial2) ,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with CdvdBurnerGrabber device object here...

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}

}

// Somewhere in the code

// Try to find device
if (
StarBurn_FindDevice(
    SCSI_DEVICE_RO_DIRECT_ACCESS,
    TRUE,
    ( PCALLBACK )( Find_Callback ),
    NULL
    ) == 0
)
{
// Handle error here...
}
else
{
// All actions are in the callback function...
}
```

2.1.142 StarBurn_GetAudioFileStreamSizeInUCHARs Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_GetAudioFileStreamSizeInUCHARs(
    IN PCHAR p__PCHAR__AudioFilePathAndName,
    OUT PLONG p__PLONG__StreamSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__AudioFilePathAndName	Pointer to full path and name of supported audio file.
OUT PLONG p__PLONG__StreamSizeInUCHARs	Pointer to the variable to receive audio stream size in UCHARs.

OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error (if function would return EN_SYSTEM_CALL_FAILED).

Returns

Execution status.

Description

This function returns audio stream size in UCHARs for passed supported as source audio file.

Remarks

This call is nearly obsolete and supported only for legacy software. It's *STRONGLY* recommended to use StarWave API calls to deal with compressed and uncompressed audio streams.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_UncompressedFileSupportedIs (see page 362), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileSupportedIs (see page 355), StarBurn_GetAudioFileStreamSizeInUCHARs_Fast (see page 234)

Example

Please see TrackAtOnceFromFile and SessionAtOnceFromFile samples to see how to use StarBurn_IsAudioFileSupported (see page 246) API call and AudioCompressor sample to see how to use StarWave API calls.

2.1.143 StarBurn_GetAudioFileStreamSizeInUCHARs_Fast Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_GetAudioFileStreamSizeInUCHARs_Fast(
    IN PCHAR p__PCHAR__AudioFilePathAndName,
    OUT PLONG p__PLONG__StreamSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__AudioFilePathAndName	Pointer to full path and name of supported audio file.
OUT PLONG p__PLONG__StreamSizeInUCHARs	Pointer to the variable to receive audio stream size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error (if function would return EN_SYSTEM_CALL_FAILED).

Returns

Execution status.

Description

This function returns audio stream size in UCHARs for passed supported as source audio file.

Remarks

This call is nearly obsolete and supported only for legacy software. It's *STRONGLY* recommended to use StarWave API calls to deal with compressed and uncompressed audio streams.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_UncompressedFileSupportedIs (see page 362), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileSupportedIs (see page 355), StarBurn_GetAudioFileStreamSizeInUCHARs (see page 233)

Example

Please see TrackAtOnceFromFile and SessionAtOnceFromFile samples to see how to use StarBurn_IsAudioFileSupported (see page 246) API call and AudioCompressor sample to see how to use StarWave API calls.

2.1.144
StarBurn_GetAudioFileStreamSizeInUCHARs_UnicodeFast Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_GetAudioFileStreamSizeInUCHARs_UnicodeFast(
    IN PWCHAR p__PWCHAR__AudioFilePathAndName,
    OUT PLONG p__PLONG__StreamSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_GetAudioFileStreamSizeInUCHARs_UnicodeFast.

2.1.145 StarBurn_GetAudioFileStreamSizeInUCHARsUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_GetAudioFileStreamSizeInUCHARsUnicode(
    IN PWCHAR p__PWCHAR__AudioFilePathAndName,
    OUT PLONG p__PLONG__StreamSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_GetAudioFileStreamSizeInUCHARsUnicode.

2.1.146 StarBurn_GetBufferUnderrunTimeOutInMs Function

C++

```
__stdcall STARBURN_IMPEX_API LONG StarBurn_GetBufferUnderrunTimeOutInMs(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Buffer underrun timeout in milliseconds.

Description

This function returns buffer underrun timeout in milliseconds. It's amount of time StarBurn core would wait before resubmitting each command to the drive if burning was faster then reading and StarBurn had exhausted software cache it uses for buffering. Modern hardware has BUP (Buffer Underrun Protection) and would perfectly survive in such a condition and keep burning disc still usable.

Remarks

Having large timeout would allow StarBurn to grab more information into software cache, at the same time having timeout small and disc burning speed MUCH faster then reading speed from the source media (say slow hard disk or network) would produce a lot of BUP errors, start-stop cycles and resulting recorded disc quality would be very low. In general it's a good idea not to touch buffer underrun timeout value at all and keep everything AS IS. It's value for "hardcore tuning".

See Also

StarBurn_SetBufferUnderrunTimeOutInMs (see page 332)

Example

There are no samples for StarBurn_GetBufferUnderrunTimeOutInMs(...) API call.

2.1.147 StarBurn_GetDeviceLetter Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_GetDeviceLetter(
    IN PCHAR p__PCHAR__DeviceName,
    OUT PCHAR p__PCHAR__DeviceLetter,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__DeviceName	Device name to return the letter for.
OUT PCHAR p__PCHAR__DeviceLetter	Buffer to receive device root path.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function returns device letter (device root path) for a device name, like the one used in StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)(...) API call.

See Also

StarBurn_CdvdBurnerGrabber_CreateEx (see page 33).

Example

Please see GUI samples as example how to use StarBurn_GetDeviceLetter API call.

2.1.148 StarBurn_GetDeviceLetterUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_GetDeviceLetterUnicode(
    IN PWCHAR p__PCHAR__DeviceName,
    OUT PWCHAR p__PCHAR__DeviceLetter,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_GetDeviceLetterUnicode.

2.1.149 StarBurn_GetDeviceNameByDeviceAddress Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_GetDeviceNameByDeviceAddress(
    IN UCHAR p__UCHAR__DevicePort,
    IN UCHAR p__UCHAR__DeviceBus,
    IN UCHAR p__UCHAR__DeviceID,
    IN UCHAR p__UCHAR__DeviceLUN,
    OUT PCHAR p__PCHAR__DeviceName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN UCHAR p__UCHAR__DevicePort	Device port.
IN UCHAR p__UCHAR__DeviceBus	Device bus.
IN UCHAR p__UCHAR__DeviceID	Device ID.
IN UCHAR p__UCHAR__DeviceLUN	Device LUN.
OUT PCHAR p__PCHAR__DeviceName	Pointer to the buffer device name will be copied to.

Returns

Execution status. EN_SUCCESS if device name was located by device address.

Description

This function gets device name by device address.

Remarks

Good idea is to fill the device name with zeros before submitting it to this function. Buffer must be large enough to hold all the device name.

See Also

StarBurn_GetVersion (see page 244), StarBurn_CdvdBurnerGrabber_Create (see page 31)

Example

This example gets device name by device address.

```
// Somewhere in the data region
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
CHAR l__CHAR__DeviceName[ MAX_PATH ];

// Prepare device name
RtlZeroMemory(
    &l__CHAR__DeviceName,
    sizeof( l__CHAR__DeviceName )
    );

// Try to get device name by device address 0:0:1:0
l__EXCEPTION_NUMBER =
StarBurn_GetDeviceNameByDeviceAddress(
    0,
    1,
    0,
    0,
    ( PCHAR )( &l__CHAR__DeviceName )
    );

// Check for correct reply
```

```
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with device name here...
```

2.1.150 StarBurn_GetDeviceTimeOutByDeviceAddress Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_GetDeviceTimeOutByDeviceAddress(
    IN UCHAR p__UCHAR__DevicePort,
    IN UCHAR p__UCHAR__DeviceBus,
    IN UCHAR p__UCHAR__DeviceID,
    IN UCHAR p__UCHAR__DeviceLUN,
    OUT PULONG p__PULONG__DeviceTimeOutInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN UCHAR p__UCHAR__DevicePort	Device port.
IN UCHAR p__UCHAR__DeviceBus	Device SCSI bus.
IN UCHAR p__UCHAR__DeviceID	Device SCSI ID.
IN UCHAR p__UCHAR__DeviceLUN	Device LUN.
OUT PULONG p__PULONG__DeviceTimeOutInSeconds	Pointer to the variable to receive device timeout in seconds.

Returns

Execution status.

Description

This function gets device timeout in seconds by device SCSI address.

Remarks

This one is for ASPI devices only.

See Also

StarBurn_SetDeviceTimeOutByDeviceAddress (see page 333)

Example

Please see FindDevice sample as example how to use StarBurn_GetDeviceTimeOutByDeviceAddress API call.

2.1.151 StarBurn_GetDVDPadding Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_GetDVDPadding(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Is DVD padding mode enabled (TRUE) or disabled (FALSE).

Description

This function returns is DVD padding mode (when at least 1GB of the data would be recorded to DVD media - required for DVD-Video compilations to work properly on standalone DVD players) enabled (TRUE) or disabled (FALSE).

Remarks

"Padding mode" should be always enabled for DVD-Video compilations (or resulting disc would not be playable on standalone DVD players) and should be always disabled for generic data discs (or quite a lot of disc capacity would be simply wasted).

See Also

StarBurn_SetDVDPadding (see page 333)

Example

There are no examples for StarBurn_GetDVDPadding(...) API call.

2.1.152 StarBurn_GetDVDPLUSRDLCompatibleMode Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_GetDVDPLUSRDLCompatibleMode(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Is DVD+R DL compatible mode enabled (TRUE) or disabled (FALSE).

Description

This function returns is DVD+R DL (Dual Layer) so-called "compatible mode" (when layer would be switched exactly at 1/2 of the data payload recorded on the disc - required for DVD-Video compilations to work properly) enabled (TRUE) or disabled (FALSE).

Remarks

"Compatible mode" should be always enabled for DVD-Video compilations (or resulting disc would not be playable on standalone DVD players) and should be always disabled for generic data discs (or quite a lot of disc capacity would be simply wasted).

See Also

StarBurn_SetDVDPLUSRDLCompatibleMode (see page 334)

Example

There are no examples for StarBurn_GetDVDPLUSRDLCompatibleMode(...) API call.

2.1.153 StarBurn_GetEjectAfterFail Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_GetEjectAfterFail(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Is "eject after fail" mode enabled (TRUE) or disabled (FALSE).

Description

This function returns is so-called "eject after fail" (if during burning process error would happen - should StarBurn eject disc or not) enabled (TRUE) or disabled (FALSE).

Remarks

It's a good idea to keep everything AS IS, however if StarBurn is used with automatic loaders or whatever sometimes it's required to keep manual control over eject process.

See Also

StarBurn_SetEjectAfterFail (see page 335)

Example

There are no examples for StarBurn_GetEjectAfterFail(...) API call.

2.1.154 StarBurn_GetFastReadTOC Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_GetFastReadTOC(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Is "fast read TOC" mode enabled (TRUE) or disabled (FALSE).

Description

This function returns is so-called "fast read TOC" (TOC information is not 100% accurate - to get EXACT track length StarBurn trys to read beginning and end of the track to find sub-channel index switch indicating EXACT track beginning or EXACT track end) mode enabled (TRUE) or disabled (FALSE).

Remarks

It's a good idea to keep everything AS IS, however if StarBurn takes a lot of time to analyze the disc (it can happen when drive hardware error correction cannot be disabled and reading pre-gap and post-gap of the track is DOG SLOW) "fast read TOC" could be switched ON.

See Also

StarBurn_SetFastReadTOC (see page 335)

Example

Please see GrabTrack sample no how to use StarBurn_GetFastReadTOC(...) API call.

2.1.155 StarBurn_GetId Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_GetId(
    OUT PULONG p__PULONG__SystemError,
    OUT PULONG p__PULONG__Version,
    OUT PCHAR p__PCHAR__VersionText,
    IN ULONG p__ULONG__VersionTextSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PULONG p__PULONG__SystemError	Pointer to system error value (if function would return EN_SYSTEM_CALL_FAILED).
OUT PULONG p__PULONG__Version	Pointer to the variable to receive ASPI layer version number.
OUT PCHAR p__PCHAR__VersionText	Pointer to the buffer to receive ASPI layer version text.
IN ULONG p__ULONG__VersionTextSizeInUCHARs	Version text buffer size in UCHARs.

Returns

Execution status.

Description

This function gets ASPI layer ID strings.

Remarks

This one is for Rocket Division Software ASPI layer only.

See Also

There are no companion API calls for this one.

Example

Please see FindDevice sample as example how to use StarBurn_GetId API call.

2.1.156 StarBurn_GetIs64KBIO Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_GetIs64KBIO(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Is "64KB I/O" mode enabled (TRUE) or disabled (FALSE).

Description

This function returns is so-called "64KB I/O" (when StarBurn issues 64KB I/O packets instead of the 32KB ones) currently

enabled (TRUE) or disabled (FALSE).

Remarks

There's no need to play with this parameter unless you really have broken device you will to force 32KB I/O requests for.

See Also

StarBurn_SetIs64KBIO (see page 336)

Example

There are no examples for StarBurn_GetIs64KBIO(...) API call.

2.1.157 StarBurn_GetIsCollisionDetectionDisabled Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_GetIsCollisionDetectionDisabled(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Is collision detection mode disabled (TRUE) or enabled (FALSE).

Description

This function returns is collision detection disabled (TRUE) or enabled (FALSE). If collision detection is enabled - every new file added to ISO9660 or Joliet file tree would be checked to have unique name. If such a file already exist - special callback (collision detection one) would be called so user would be able to either replace or rename new or old file. However if collision detection is disabled - no comparison and no actions would be performed. File would be just added to the destination file system image AS IS, having it's original name.

Remarks

If new image is created from the content stored on the hard disk - there's no way for file names to be the same. So file system image creation process could be speed up quite a lot by turning collision detection OFF (especially if there are a lot of files inside each directory). In other cases it's not recommended to play with this option.

See Also

StarBurn_SetIsCollisionDetectionDisabled (see page 336)

Example

Please see BuildImage sample to find out how collision detection could be disabled and enabled during file system image creation.

2.1.158 StarBurn_GetIsSafeGrabbingEnabled Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_GetIsSafeGrabbingEnabled(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Is "safe grabbing" mode enabled (TRUE) or disabled (FALSE).

Description

This function returns is so-called "safe grabbing" (when StarBurn would mix read commands with checking for device to become ready - required to workaround broken ATAPI-to-USB bridges) enabled (TRUE) or disabled (FALSE).

Remarks

There's no need to play with this parameter unless you really have ATAPI device in external USB enclosure and it hangs under heavy I/O load.

See Also

StarBurn_SetIsSafeGrabbingEnabled (see page 337)

Example

There are no examples for StarBurn_GetIsSafeGrabbingEnabled(...) API call.

2.1.159 StarBurn_GetVersion Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_GetVersion(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

ULONG with the packed date. This function cannot fail.

Description

This function returns a 32-bit unsigned long value that contains packed numbers that represent the year, the month and the date of the toolkit build. So 10th of March, year of 1978 will look like 0x19780310.

Remarks

Actually it's a good idea to check for supported toolkit version before performing any actions.

See Also

Samples for more details about differences between newer and older toolkit versions.

Example

This example checks for the toolkit version.

```
// Somewhere in the data region
ULONG l__ULONG__Version;

// Get toolkit version
l__ULONG__Version =
StarBurn_GetVersion();

// Check for correct number
if ( l__ULONG__Version < SUPPORTED_VERSION_NUMBER )
{
// Handle error here...
}
```

2.1.160 StarBurn_GetVolumeIDs Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_GetVolumeIDs(
    IN PCHAR p__PCHAR__VolumeName,
    OUT PCHAR p__PCHAR__VolumeID
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__VolumeName	ISO9660/Joliet name we'd like to query.
OUT PCHAR p__PCHAR__VolumeID	Pointer to destination buffer volume ID would be copied to.

Returns

Execution status.

Description

This function gets volume ID from ISO9660/Joliet volume name.

Remarks

This one is for ISO9660/Joliet volumes only.

See Also

There are no companion API calls for this one.

Example

Please see BuildImageWithImportFromFile sample as example how to use StarBurn_GetVolumeIDs API call.

2.1.161 StarBurn_ImageFile_UDFFileSystemLookup Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ImageFile_UDFFileSystemLookup(
    IN PCHAR p__PCHAR__ISOFileName,
    IN PCALLBACK p__Callback,
    IN PVOID p__CallbackContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__ISOFileName	File name of the file to parse.
p__PCALLBACK_Callback	Pointer to callback function that receives information about found files.
p__PVOID_CallbackContext	Callback context.

Returns

Execution status.

Description

This function parses the UDF file system inside image file and calls a callback function for each found file. The information about LBAs that file resides on is passed to callback function.

Remarks

Information that is passed to callback function should be copied and stored on client side. Strings may not be valid outside callback function.

Example

Please see UDFLookupISO sample as example how to use StarBurn_ImageFile_UDFFileSystemLookup API call.

2.1.162 StarBurn_ImageFile_UDFFileSystemLookupEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ImageFile_UDFFileSystemLookupEx(
    IN PCHAR p__PCHAR__ISOFileName,
    IN PCALLBACK p__Callback,
    IN PVOID p__CallbackContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__ISOFileName	File name of the file to parse.
p__PCALLBACK_Callback	Pointer to callback function that receives information about found files.
p__PVOID_CallbackContext	Callback context.

Returns

Execution status.

Description

This function parses the UDF file system inside image file and calls a callback function for each found file and directory. The information about LBAs that file resides on is passed to callback function.

Remarks

Information that is passed to callback function should be copied and stored on client side. Strings may not be valid outside callback function.

Example

Please see UDFLookupISO sample as example how to use StarBurn_ImageFile_UDFFileSystemLookup (see page 245) API call.

2.1.163 StarBurn_IsAudioFileSupported Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_IsAudioFileSupported(
    IN LPCSTR p__PCHAR__AudioFilePathAndName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN LPCSTR p__PCHAR__AudioFilePathAndName`	Pointer to full path and name of "suspected" audio file.

Returns

Execution status.

Description

This function returns is currently passed file supported as source audio file.

Remarks

This call is nearly obsolete and supported only for legacy software. It's *STRONGLY* recommended to use StarWave API calls to deal with compressed and uncompressed audio streams (StarBurn_StarWave_UncompressedFileSupportedIs (see page 362) and of course it's companion StarBurn_StarWave_CompressedFileSupportedIs (see page 355)).

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_UncompressedFileSupportedIs (see page 362), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileSupportedIs (see page 355)

Example

Please see TrackAtOnceFromFile and SessionAtOnceFromFile samples to see how to use StarBurn_IsAudioFileSupported API call and AudioCompressor sample to see how to use StarWave API calls.

2.1.164 StarBurn_IsAudioFileSupportedUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_IsAudioFileSupportedUnicode(
    IN LPCWSTR p__PWCHAR__AudioFilePathAndName
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_IsAudioFileSupportedUnicode.

2.1.165 StarBurn_IsLocalDateTimeUsed Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_IsLocalDateTimeUsed();
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_IsLocalDateTimeUsed.

2.1.166 StarBurn_ISO2_Check1999Name Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_ISO2_Check1999Name(
    const char* Name,
    BOOL bIsDirectory
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char* Name	Pointer to zero terminated string with file or directory name.
BOOL bIsDirectory	TRUE - check directory name, FALSE - check file name.

Returns

TRUE if given string is a valid ISO9660:1999 file/directory name. FALSE otherwise.

Description

This function checks if given string valid ISO9660:1999 file (or directory) name

2.1.167 StarBurn_ISO2_CheckJolietName Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_ISO2_CheckJolietName(
    const unsigned short* UnicodeName,
    BOOL bIsDirectory
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short* UnicodeName	Pointer to zero terminated string (Unicode) with file or directory name.
BOOL bIsDirectory	TRUE - check directory name, FALSE - check file name.

Returns

TRUE if given string is a valid ISO9660 Joliet file/directory name. FALSE otherwise.

Description

This function checks if given string valid ISO9660 Joliet file (or directory) name

2.1.168 StarBurn_ISO2_CheckLevel1Name Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_ISO2_CheckLevel1Name(
    const char* Name,
    BOOL bIsDirectory
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char* Name	Pointer to zero terminated string with file or directory name.
BOOL bIsDirectory	TRUE - check directory name, FALSE - check file name.

Returns

TRUE if given string is a valid ISO9660 Level 1 file/directory name. FALSE otherwise.

Description

This function checks if given string valid ISO9660 Level 1 file (or directory) name

2.1.169 StarBurn_ISO2_CheckLevel2Name Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_ISO2_CheckLevel2Name(
    const char* Name,
    BOOL bIsDirectory
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char* Name	Pointer to zero terminated string with file or directory name.
BOOL bIsDirectory	TRUE - check directory name, FALSE - check file name.

Returns

TRUE if given string is a valid ISO9660 Level 2 file/directory name. FALSE otherwise.

Description

This function checks if given string valid ISO9660 Level 2 file (or directory) name

2.1.170 StarBurn_ISO2_CheckRelaxedJolietName Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_ISO2_CheckRelaxedJolietName(
    const unsigned short* UnicodeName,
    BOOL bIsDirectory
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short* UnicodeName	Pointer to zero terminated string (Unicode) with file or directory name.
BOOL bIsDirectory	TRUE - check directory name, FALSE - check file name.

Returns

TRUE if given string is a valid ISO9660 Joliet file/directory name. FALSE otherwise.

Description

This function checks if given string valid ISO9660 Joliet file (or directory) name

with less restrictions: names may contain more than one dot.

2.1.171 StarBurn_ISO2_CreateNewName Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_ISO2_CreateNewName(
    char* Name,
    unsigned long Seed
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
char* Name	Pointer to zero terminated string (ANSI) with file or directory name.
unsigned long Seed	Number that is added to file name.

Returns

TRUE if file name modified successfully. FALSE otherwise.

Description

This function creates new ANSI name by adding ~XX (where XX is a number (Seed)) at the end of file name.

2.1.172 StarBurn_ISO2_DirectoryBrowse Function

C++

```
__stdcall STARBURN_IMPEX_API void * StarBurn_ISO2_DirectoryBrowse(
    void * Directory,
```

```
    void * Last
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Directory	Pointer to ISO9660 directory object.
void * Last	Pointer to last processed kid node (or NULL if we need first kid node in the kids list).

Returns

Pointer to currently processed kid node (or NULL if we've reached end of the kids list).

Description

This function browses ISO9660 directory content.

2.1.173 StarBurn_ISO2_DirectoryCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_DirectoryCreate(
    void ** Directory,
    const char * Name,
    unsigned long GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Directory	Pointer to pointer to directory object.
const char * Name	Pointer to zero-terminated directory name.
unsigned long GUID	GUID value.
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 directory.

2.1.174 StarBurn_ISO2_DirectoryDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_ISO2_DirectoryDestroy(
    void * Directory
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Directory`	Pointer to directory object to destroy.

Returns

Nothing

Description

This function destroys created ISO9660 directory object.

2.1.175 StarBurn_ISO2_DirectoryProcess Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_DirectoryProcess(
    PVOID * NewDir,
    const char * Name,
    const unsigned short * UnicodePathName,
    unsigned long * GUID,
    void * Parent,
    PSTARBURN_ISO2_PROGRESS_CALLBACK ProgressCallback,
    PVOID ProgressCallbackContext,
    PSTARBURN_ISO2_COLLISION_CALLBACK CollisionCallback,
    PVOID CollisionCallbackContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`const char * Name`	Directory name itself (zero-terminated string).
`const unsigned short * UnicodePathName`	Directory path and name (zero-terminated Unicode string).
`unsigned long * GUID`	GUID value
`void * Parent`	Pointer to parent directory.
`PSTARBURN_ISO2_PROGRESS_CALLBACK ProgressCallback`	Callback function that is called on addition of file/directory
`PVOID ProgressCallbackContext`	Callback context passed to ProgressCallback
`PSTARBURN_ISO2_COLLISION_CALLBACK CollisionCallback`	Callback function that is called on name collision
`PVOID CollisionCallbackContext`	Callback context passed to CollisionCallback

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single ISO9660 directory.

2.1.176 StarBurn_ISO2_DirectoryQuery Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_ISO2_DirectoryQuery(
    void * Directory,
    STARBURN_ISO9660_DIRECTORY_INFO * Info
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`STARBURN_ISO9660_DIRECTORY_INFO * Info`	Pointer to ISO9660 directory information structure.
`File`	Pointer to ISO9660 directory object.

Returns

Nothing

Description

This function do query for ISO9660 file properties.

2.1.177 StarBurn_ISO2_DirectoryRootCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_DirectoryRootCreate(
    void ** Directory,
    unsigned long GUID,
    unsigned char Type
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** Directory`	Pointer to pointer to directory object.
`unsigned long GUID`	GUID value
`unsigned char Type`	Type of created tree.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates root ISO9660 directory.

Remarks

Type can be one of the following values:

STARBURN_ISO9660_TYPE_LEVEL1 (see page 643) - ISO 9660 Level 1

STARBURN_ISO9660_TYPE_LEVEL2 (see page 644) - ISO 9660 Level 2

STARBURN_ISO9660_TYPE_1999 (see page 643) - ISO 9660:1990

STARBURN_ISO9660_TYPE_JOLIET (see page 643) - ISO 9660 with Joliet extension

STARBURN_ISO9660_TYPE_JOLIET_RELAXED (see page 643) - ISO 9660 with Joliet extension (less restrictions)

2.1.178 StarBurn_ISO2_DirectoryUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_DirectoryUnicodeCreate(
```

```
    void ** Directory,
    const unsigned short * UnicodeName,
    unsigned long GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Directory	Pointer to pointer to directory object.
const unsigned short * UnicodeName	Pointer to zero-terminated directory name (UNICODE format).
unsigned long GUID	GUID value.
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UNICODE ISO9660 directory.

2.1.179 StarBurn_ISO2_DirectoryUnicodeProcess Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_DirectoryUnicodeProcess(
    PVOID * NewDir,
    const unsigned short * UnicodeName,
    const unsigned short * UnicodePathName,
    unsigned long * GUID,
    void * Parent,
    PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK ProgressUnicodeCallback,
    PVOID ProgressCallbackContext,
    PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK CollisionUnicodeCallback,
    PVOID CollisionCallbackContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short * UnicodeName	Directory name itself (zero-terminated string in UNICODE format).
const unsigned short * UnicodePathName	Directory path and name (zero-terminated string in UNICODE format).
unsigned long * GUID	GUID value
void * Parent	Pointer to parent directory.
PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK ProgressUnicodeCallback	Callback function that is called on addition of file/directory
PVOID ProgressCallbackContext	Callback context passed to ProgressCallback
PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK CollisionUnicodeCallback	Callback function that is called on name collision
PVOID CollisionCallbackContext	Callback context passed to CollisionCallback

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single UNICODE ISO9660 directory.

2.1.180 StarBurn_ISO2_FileCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileCreate(
    void ** File,
    const char * Name,
    const unsigned short * UnicodePathName,
    unsigned long GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** File	Pointer to pointer to file object.
const char * Name	Pointer to zero-terminated name.
const unsigned short * UnicodePathName	Pointer to zero-terminated path and name (Unicode format).
unsigned long GUID	GUID value
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 file.

2.1.181 StarBurn_ISO2_FileDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_ISO2_FileDestroy(
    void * File
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * File	Pointer to file object to destroy.

Returns

Nothing

Description

This function destroys created ISO9660 file object.

2.1.182 StarBurn_ISO2_FileDirectoryDateTimeGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileDirectoryDateTimeGet(
    void * FileDirectory,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to file or directory object.
`STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to output file date and time.

Returns

Operation status

Description

This function gets date and time from created ISO9660 file or directory object.

2.1.183 StarBurn_ISO2_FileDirectoryDateTimeSet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileDirectoryDateTimeSet(
    void * FileDirectory,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to file or directory object.
`STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to input file date and time.

Returns

Nothing

Description

This function sets date and time on created ISO9660 file or directory object.

2.1.184 StarBurn_ISO2_FileDirectoryNameSet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileDirectoryNameSet(
    void * FileDirectory,
    const char * Name
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to ISO9660 file or directory object.
`const char * Name`	Pointer to new file or directory name.

Returns

Operation status

Description

This function set new name for ISO9660 file or directory object.

2.1.185 StarBurn_ISO2_FileDirectoryUnicodeNameSet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileDirectoryUnicodeNameSet(
    void * FileDirectory,
    const unsigned short * UnicodeName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to ISO9660 file or directory object.
`const unsigned short * UnicodeName`	Pointer to new file or directory UNICODE name.

Returns

Nothing

Description

This function set new UNICODE name for ISO9660 file or directory object.

2.1.186 StarBurn_ISO2_FileMemoryCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileMemoryCreate(
    void ** File,
    const char * Name,
    void * MemoryRegion,
    unsigned long MemoryRegionSizeInUCHARs,
    unsigned long GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** File`	Pointer to pointer to file object.
`const char * Name`	Pointer to zero-terminated name.
`void * MemoryRegion`	Pointer to memory region containing file contents.
`unsigned long MemoryRegionSizeInUCHARs`	Memory region size in unsigned chars.
`unsigned long GUID`	GUID value
`void * Parent`	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 memory file.

2.1.187 StarBurn_ISO2_FileMemoryUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileMemoryUnicodeCreate(
    void ** File,
    const unsigned short * UnicodeName,
    void * MemoryRegion,
    unsigned long MemoryRegionSizeInUCHARs,
    unsigned long GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** File`	Pointer to pointer to file object.
`const unsigned short * UnicodeName`	Pointer to zero-terminated name (Unicode).
`void * MemoryRegion`	Pointer to memory region containing file contents.
`unsigned long MemoryRegionSizeInUCHARs`	Memory region size in unsigned chars.
`unsigned long GUID`	GUID value
`void * Parent`	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates Unicode ISO9660 memory file.

2.1.188 StarBurn_ISO2_FileQuery Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_ISO2_FileQuery(
    void * File,
    STARBURN_ISO9660_FILE_INFO * Info
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * File`	Pointer to ISO9660 file object.
`STARBURN_ISO9660_FILE_INFO * Info`	Pointer to ISO9660 file information structure.

Returns

Nothing

Description

This function do query for ISO9660 file properties.

Remarks

Some structure fields are valid only for imported or non-imported nodes. Refer to structure description.

2.1.189 StarBurn_ISO2_FileSetAttributes Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_ISO2_FileSetAttributes(
    void* File,
    unsigned long Attributes
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void* File`	Pointer to ISO9660 file object.
`unsigned long Attributes`	File attributes bit mask.

Returns

Nothing.

Description

This function sets attributes for ISO9660 file.

2.1.190 StarBurn_ISO2_FileUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_FileUnicodeCreate(
    void ** File,
    const unsigned short * UnicodeName,
    const unsigned short * UnicodePathName,
    unsigned long GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** File	Pointer to pointer to file object.
const unsigned short * UnicodeName	Pointer to zero-terminated name (UNICODE format).
const unsigned short * UnicodePathName	Pointer to zero-terminated path and name (UNICODE format).
unsigned long GUID	GUID value
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UNICODE ISO9660 file.

2.1.191 StarBurn_ISO2_ImpVolumeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_ImpVolumeCreate(
    void ** Volume,
    void * Root,
    unsigned long * GUID,
    PSTARBURN_ISO9660_READ_CALLBACK ImportCallback,
    void * CallbackContext,
    unsigned long VolumeBeginLBA,
    PSTARBURN_ISO2_COLLISION_CALLBACK CollisionCallback,
    const void * CollisionContext,
    PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK UnicodeCollisionCallback,
    const void * UnicodeCollisionContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to ISO9660 import volume object.
void * Root	Pointer to directory where contents should be added. Pointer to root directory.
unsigned long * GUID	GUID value GUID value
PSTARBURN_ISO9660_READ_CALLBACK ImportCallback	Pointer to import callback.
void * CallbackContext	Pointer to callback context.
PSTARBURN_ISO2_COLLISION_CALLBACK CollisionCallback	Callback function for name collisions handling (for ANSI names).
const void * CollisionContext	User context for CollisionCallback.
PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK UnicodeCollisionCallback	Callback function for name collisions handling (for Unicode names).
const void * UnicodeCollisionContext	User context for UnicodeCollisionCallback.
UnicodePathName	Directory path and name (zero-terminated string in UNICODE format).
Callback	Pointer to ISO9660 UNICODE progress callback.
Context	Pointer to context value.
DateTime	Pointer to date and time structure.
VolumeSizeInLBs	Volume size in logical blocks.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

STARBURN_IMPEX_API (see page 639)

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes contents of single UNICODE ISO9660 directory.

__stdcall StarBurn_ISO2_DirectoryContentsUnicodeProcess(const unsigned short *UnicodePathName, unsigned long *GUID, void *Root, PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK (see page 587) Callback, const void *Context, STARBURN_ISO2_FILE_DATE_TIME *DateTime);

This function creates ISO9660 import volume.

2.1.192 StarBurn_ISO2_ImpVolumeDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_ISO2_ImpVolumeDestroy(
    void * Volume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Volume	Pointer to ISO9660 import volume object to destroy.

Returns

Nothing.

Description

This function destroys created ISO9660 import volume object.

2.1.193 StarBurn_ISO2_ImpVolumeImport Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_ImpVolumeImport(
    void * Volume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Volume	Pointer to the ISO9660 import volume object.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function imports ISO9660 import volume content.

2.1.194 StarBurn_ISO2_ISO9660FileTreeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_ISO9660FileTreeCreate(
    void ** ISO9660FileTree,
    void * ISO9660Volume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** ISO9660FileTree	Pointer to pointer to the object that toolkit will set to the ISO 9660 File Tree object it will allocate.
void * ISO9660Volume	Pointer to ISO9660 volume object that toolkit allocated before.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function create ISO image from already created ISO9660 volume.

2.1.195 StarBurn_ISO2_VolumeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreate(
    void ** Volume,
    void * Root,
    const char * Name,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    const char * SystemIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const char * Name	Pointer to zero-terminated string with volume name (Volume Identifier).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
const char * SystemIdentifier	Pointer to zero-terminated string with system identifier.

const char * VolumeSetIdentifier	Pointer to zero-terminated string with volume set identifier.
const char * PublisherIdentifier	Pointer to zero-terminated string with publisher identifier.
const char * DataPreparerIdentifier	Pointer to zero-terminated string with data preparer identifier.
const char * ApplicationIdentifier	Pointer to zero-terminated string with application identifier.
const char * CopyrightFileIdentifier	Pointer to zero-terminated string with copyright file identifier.
const char * AbstractFileIdentifier	Pointer to zero-terminated string with abstract file identifier.
const char * BibliographicFileIdentifier	Pointer to zero-terminated string with bibliographic file identifier.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 volume.

2.1.196 StarBurn_ISO2_VolumeCreateBoot Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreateBoot(
    void ** Volume,
    void * Root,
    const char * Name,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    const unsigned short * BootImageFileName,
    unsigned char EmulationType,
    unsigned short SectorsToLoad,
    unsigned short LoadSegment,
    const char * SystemIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const char * Name	Pointer to zero-terminated string with volume name (Volume Identifier).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
const unsigned short * BootImageFileName	Pointer to zero-terminated Unicode string with boot image file path.
unsigned char EmulationType	Media type of bootable volume.
unsigned short SectorsToLoad	Number of sectors to load from boot image.
unsigned short LoadSegment	Load segment
const char * SystemIdentifier	Pointer to zero-terminated string with system identifier.
const char * VolumeSetIdentifier	Pointer to zero-terminated string with volume set identifier.
const char * PublisherIdentifier	Pointer to zero-terminated string with publisher identifier.
const char * DataPreparerIdentifier	Pointer to zero-terminated string with data preparer identifier.
const char * ApplicationIdentifier	Pointer to zero-terminated string with application identifier.

const char * CopyrightFileIdentifier	Pointer to zero-terminated string with copyright file identifier.
const char * AbstractFileIdentifier	Pointer to zero-terminated string with abstract file identifier.
const char * BibliographicFileIdentifier	Pointer to zero-terminated string with bibliographic file identifier.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 bootable volume (ANSI names).

2.1.197 StarBurn_ISO2_VolumeCreateBootElTorito Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreateBootElTorito(
    void ** Volume,
    void * Root,
    const char * Name,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    void * BootCatalog,
    const char * SystemIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const char * Name	Pointer to zero-terminated string with volume name (Volume Identifier).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
void * BootCatalog	Pointer to ElTorito boot catalog
const char * SystemIdentifier	Pointer to zero-terminated string with system identifier.
const char * VolumeSetIdentifier	Pointer to zero-terminated string with volume set identifier.
const char * PublisherIdentifier	Pointer to zero-terminated string with publisher identifier.
const char * DataPreparerIdentifier	Pointer to zero-terminated string with data preparer identifier.
const char * ApplicationIdentifier	Pointer to zero-terminated string with application identifier.
const char * CopyrightFileIdentifier	Pointer to zero-terminated string with copyright file identifier.
const char * AbstractFileIdentifier	Pointer to zero-terminated string with abstract file identifier.
const char * BibliographicFileIdentifier	Pointer to zero-terminated string with bibliographic file identifier.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 bootable volume (ANSI names).

2.1.198 StarBurn_ISO2_VolumeCreateEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreateEx(
    void ** Volume,
    void * Root,
    const char * Name,
    const unsigned short * UnicodeName,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    const char * SystemIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier,
    const unsigned short * UnicodeSystemIdentifier,
    const unsigned short * UnicodeVolumeSetIdentifier,
    const unsigned short * UnicodePublisherIdentifier,
    const unsigned short * UnicodeDataPreparerIdentifier,
    const unsigned short * UnicodeApplicationIdentifier,
    const unsigned short * UnicodeCopyrightFileIdentifier,
    const unsigned short * UnicodeAbstractFileIdentifier,
    const unsigned short * UnicodeBibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const char * Name	Pointer to zero-terminated string with volume name (Volume Identifier).
const unsigned short * UnicodeName	Pointer to zero-terminated string with volume name (in UNICODE).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
const char * SystemIdentifier	Pointer to zero-terminated string with system identifier.
const char * VolumeSetIdentifier	Pointer to zero-terminated string with volume set identifier.
const char * PublisherIdentifier	Pointer to zero-terminated string with publisher identifier.
const char * DataPreparerIdentifier	Pointer to zero-terminated string with data preparer identifier.
const char * ApplicationIdentifier	Pointer to zero-terminated string with application identifier.
const char * CopyrightFileIdentifier	Pointer to zero-terminated string with copyright file identifier.
const char * AbstractFileIdentifier	Pointer to zero-terminated string with abstract file identifier.
const char * BibliographicFileIdentifier	Pointer to zero-terminated string with bibliographic file identifier.
const unsigned short * UnicodeSystemIdentifier	Pointer to zero-terminated Unicode string with system identifier.
const unsigned short * UnicodeVolumeSetIdentifier	Pointer to zero-terminated Unicode string with volume set identifier.
const unsigned short * UnicodePublisherIdentifier	Pointer to zero-terminated Unicode string with publisher identifier.
const unsigned short * UnicodeDataPreparerIdentifier	Pointer to zero-terminated Unicode string with data preparer identifier.
const unsigned short * UnicodeApplicationIdentifier	Pointer to zero-terminated Unicode string with application identifier.
const unsigned short * UnicodeCopyrightFileIdentifier	Pointer to zero-terminated Unicode string with copyright file identifier.

const unsigned short * UnicodeAbstractFileIdentifier	Pointer to zero-terminated Unicode string with abstract file identifier.
const unsigned short * UnicodeBibliographicFileIdentifier	Pointer to zero-terminated Unicode string with bibliographic file identifier.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 volume (with both Unicode and ANSI names).

2.1.199 StarBurn_ISO2_VolumeCreateExBoot Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreateExBoot(
    void ** Volume,
    void * Root,
    const char * Name,
    const unsigned short * UnicodeName,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    const unsigned short * BootImageFileName,
    unsigned char EmulationType,
    unsigned short SectorsToLoad,
    unsigned short LoadSegment,
    const char * SystemIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier,
    const unsigned short * UnicodeSystemIdentifier,
    const unsigned short * UnicodeVolumeSetIdentifier,
    const unsigned short * UnicodePublisherIdentifier,
    const unsigned short * UnicodeDataPreparerIdentifier,
    const unsigned short * UnicodeApplicationIdentifier,
    const unsigned short * UnicodeCopyrightFileIdentifier,
    const unsigned short * UnicodeAbstractFileIdentifier,
    const unsigned short * UnicodeBibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const char * Name	Pointer to zero-terminated string with volume name (Volume Identifier).
const unsigned short * UnicodeName	Pointer to zero-terminated string with volume name (in UNICODE).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
const unsigned short * BootImageFileName	Pointer to zero-terminated Unicode string with boot image file path.
unsigned char EmulationType	Media type of bootable volume.
unsigned short SectorsToLoad	Number of sectors to load from boot image.
unsigned short LoadSegment	Load segment
const char * SystemIdentifier	Pointer to zero-terminated string with system identifier.
const char * VolumeSetIdentifier	Pointer to zero-terminated string with volume set identifier.

const char * PublisherIdentifier	Pointer to zero-terminated string with publisher identifier.
const char * DataPreparerIdentifier	Pointer to zero-terminated string with data preparer identifier.
const char * ApplicationIdentifier	Pointer to zero-terminated string with application identifier.
const char * CopyrightFileIdentifier	Pointer to zero-terminated string with copyright file identifier.
const char * AbstractFileIdentifier	Pointer to zero-terminated string with abstract file identifier.
const char * BibliographicFileIdentifier	Pointer to zero-terminated string with bibliographic file identifier.
const unsigned short * UnicodeVolumeSetIdentifier	Pointer to zero-terminated Unicode string with volume set identifier.
const unsigned short * UnicodePublisherIdentifier	Pointer to zero-terminated Unicode string with publisher identifier.
const unsigned short * UnicodeDataPreparerIdentifier	Pointer to zero-terminated Unicode string with data preparer identifier.
const unsigned short * UnicodeApplicationIdentifier	Pointer to zero-terminated Unicode string with application identifier.
const unsigned short * UnicodeCopyrightFileIdentifier	Pointer to zero-terminated Unicode string with copyright file identifier.
const unsigned short * UnicodeAbstractFileIdentifier	Pointer to zero-terminated Unicode string with abstract file identifier.
const unsigned short * UnicodeBibliographicFileIdentifier	Pointer to zero-terminated Unicode string with bibliographic file identifier.
UnicodeSytemIdentifier	Pointer to zero-terminated Unicode string with system identifier.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 bootable volume (with both Unicode and ANSI names).

2.1.200 StarBurn_ISO2_VolumeCreateUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreateUnicode(
    void ** Volume,
    void * Root,
    const unsigned short * Name,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    const unsigned short * SystemIdentifier,
    const unsigned short * VolumeSetIdentifier,
    const unsigned short * PublisherIdentifier,
    const unsigned short * DataPreparerIdentifier,
    const unsigned short * ApplicationIdentifier,
    const unsigned short * CopyrightFileIdentifier,
    const unsigned short * AbstractFileIdentifier,
    const unsigned short * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const unsigned short * Name	Pointer to zero-terminated string with volume name (Volume Identifier).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
const unsigned short * SystemIdentifier	Pointer to zero-terminated Unicode string with system identifier.
const unsigned short * VolumeSetIdentifier	Pointer to zero-terminated Unicode string with volume set identifier.
const unsigned short * PublisherIdentifier	Pointer to zero-terminated Unicode string with publisher identifier.
const unsigned short * DataPreparerIdentifier	Pointer to zero-terminated Unicode string with data preparer identifier.
const unsigned short * ApplicationIdentifier	Pointer to zero-terminated Unicode string with application identifier.
const unsigned short * CopyrightFileIdentifier	Pointer to zero-terminated Unicode string with copyright file identifier.

const unsigned short * AbstractFileIdentifier	Pointer to zero-terminated Unicode string with abstract file identifier.
const unsigned short * BibliographicFileIdentifier	Pointer to zero-terminated Unicode string with bibliographic file identifier.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 volume.

2.1.201 StarBurn_ISO2_VolumeCreateUnicodeBoot Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreateUnicodeBoot(
    void ** Volume,
    void * Root,
    const unsigned short * UnicodeName,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    const unsigned short * BootImageFileName,
    unsigned char EmulationType,
    unsigned short SectorsToLoad,
    unsigned short LoadSegment,
    const unsigned short * SystemIdentifier,
    const unsigned short * VolumeSetIdentifier,
    const unsigned short * PublisherIdentifier,
    const unsigned short * DataPreparerIdentifier,
    const unsigned short * ApplicationIdentifier,
    const unsigned short * CopyrightFileIdentifier,
    const unsigned short * AbstractFileIdentifier,
    const unsigned short * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const unsigned short * UnicodeName	Pointer to zero-terminated Unicode string with volume name.
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
const unsigned short * BootImageFileName	Pointer to zero-terminated Unicode string with boot image file path.
unsigned char EmulationType	Media type of bootable volume.
unsigned short SectorsToLoad	Number of sectors to load from boot image.
unsigned short LoadSegment	Load segment
const unsigned short * SystemIdentifier	Pointer to zero-terminated Unicode string with system identifier.
const unsigned short * VolumeSetIdentifier	Pointer to zero-terminated Unicode string with volume set identifier.
const unsigned short * PublisherIdentifier	Pointer to zero-terminated Unicode string with publisher identifier.
const unsigned short * DataPreparerIdentifier	Pointer to zero-terminated Unicode string with data preparer identifier.
const unsigned short * ApplicationIdentifier	Pointer to zero-terminated Unicode string with application identifier.
const unsigned short * CopyrightFileIdentifier	Pointer to zero-terminated Unicode string with copyright file identifier.
const unsigned short * AbstractFileIdentifier	Pointer to zero-terminated Unicode string with abstract file identifier.
const unsigned short * BibliographicFileIdentifier	Pointer to zero-terminated Unicode string with bibliographic file identifier.

Name	Pointer to zero-terminated ANSI string with volume name (Volume Identifier).

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 bootable volume (Unicode names).

2.1.202 StarBurn_ISO2_VolumeCreateUnicodeBootElTorito Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeCreateUnicodeBootElTorito(
    void ** Volume,
    void * Root,
    const unsigned short * UnicodeName,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned long InitialVolumeSizeInLBs,
    void * BootCatalog,
    const unsigned short * SystemIdentifier,
    const unsigned short * VolumeSetIdentifier,
    const unsigned short * PublisherIdentifier,
    const unsigned short * DataPreparerIdentifier,
    const unsigned short * ApplicationIdentifier,
    const unsigned short * CopyrightFileIdentifier,
    const unsigned short * AbstractFileIdentifier,
    const unsigned short * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const unsigned short * UnicodeName	Pointer to zero-terminated Unicode string with volume name.
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).
void * BootCatalog	pointer to ElTorito boot catalog
const unsigned short * SystemIdentifier	Pointer to zero-terminated Unicode string with system identifier.
const unsigned short * VolumeSetIdentifier	Pointer to zero-terminated Unicode string with volume set identifier.
const unsigned short * PublisherIdentifier	Pointer to zero-terminated Unicode string with publisher identifier.
const unsigned short * DataPreparerIdentifier	Pointer to zero-terminated Unicode string with data preparer identifier.
const unsigned short * ApplicationIdentifier	Pointer to zero-terminated Unicode string with application identifier.
const unsigned short * CopyrightFileIdentifier	Pointer to zero-terminated Unicode string with copyright file identifier.
const unsigned short * AbstractFileIdentifier	Pointer to zero-terminated Unicode string with abstract file identifier.
const unsigned short * BibliographicFileIdentifier	Pointer to zero-terminated Unicode string with bibliographic file identifier.
Name	Pointer to zero-terminated ANSI string with volume name (Volume Identifier).

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates ISO9660 bootable volume (Unicode names).

2.1.203 StarBurn_ISO2_VolumeDescriptorsBufferGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeDescriptorsBufferGet(
    void * Volume,
    const unsigned char ** pBuffer,
    unsigned __int64 * SizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`const unsigned char ** pBuffer`	Pointer to the variable to receive pointer to ISO9660 volume descriptors buffer.
`unsigned __int64 * SizeInUCHARs`	Pointer to the variable to receive ISO9660 volume descriptors size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function get ISO9660 volume descriptors buffer.

Remarks

This buffer remains valid only when the volume object is valid.

It is invalidated after call to StarBurn_ISO2_VolumeDestroy (see page 270)().

It should not be freed or altered.

2.1.204 StarBurn_ISO2_VolumeDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_ISO2_VolumeDestroy(
    void * Volume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to ISO9660 volume object to destroy.

Returns

Nothing

Description

This function destroys created ISO9660 volume object.

2.1.205 StarBurn_ISO2_VolumeSeekRead Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeSeekRead(
    void * Volume,
    void * Buffer,
    unsigned long BufferSizeInLogicalBlocks,
    unsigned long OffsetInLogicalBlocks
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`void * Buffer`	Pointer to data buffer to read data to.
`unsigned long BufferSizeInLogicalBlocks`	Buffer size in logical blocks.
`unsigned long OffsetInLogicalBlocks`	Offset within ISO9660 volume in logical blocks.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function seeks to passed offset and reads requested amount of data.

2.1.206 StarBurn_ISO2_VolumeSizeInUCHARsGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_ISO2_VolumeSizeInUCHARsGet(
    void * Volume,
    unsigned __int64 * SizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`unsigned __int64 * SizeInUCHARs`	Pointer to the variable to receive ISO9660 volume size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function get ISO9660 volume size in unsigned chars.

2.1.207 StarBurn_ISO9660JolietFileTree_Add Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_Add(
    OUT PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCHAR p__PCHAR__RootDirectoryAbsolutePathAndName,
    IN PCHAR p__PCHAR__RootDirectoryNewName,
    IN FILE_TIME p__FILE_TIME,
    IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent,
    OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN PCHAR p__PCHAR__RootDirectoryAbsolutePathAndName	Pointer to directory name to build the ISO9660 or Joliet image from. This parameters can be NULL (but in this case p__PCHAR__RootDirectoryNewName cannot!!!) if virtual (non-present on the disk) directory must be added to the image.
IN PCHAR p__PCHAR__RootDirectoryNewName	Pointer to directory name to be stored in the ISO9660 or Joliet image as root name, NULL to include default name from the previous parameter.
IN FILE_TIME p__FILE_TIME	File time that will be included in file system image (See FILE_TIME (see page 490)).
IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent	Pointer to pointer to the file tree node we'll use as parent. This is the result of either tree walking with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() or StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)(). NULL can be passed instead of the result of StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild	Pointer to pointer to the new created file tree node.

Returns

Execution status. EN_SUCCESS if the tree was reallocated successfully, E_MEMORY_ALLOCATION_FAILED if there is not enough memory to reallocate the tree. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.

Remarks

Please see the BuildImage sample that will demonstrate how progress indication can be build and what parameters in general are passed into the callback function during the operations with file tree object.

See Also

StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), FILE_TREE (see page 490), FILE_TIME (see page 490), StarBurn_ISO9660JolietFileTree_Create (see page 288),

StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_AddEx (see page 274)

Example

This example allocates Joliet file tree and destroys it after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__Root;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get root here
l__PVOID__Root =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct reply
if ( l__PVOID__Root == NULL )
{
// Handle error here, keep in mind that root CAN be NULL if nothing was added to tree
before...
}

// Try to add all from the root directory of the drive D: to already created Joliet file
tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Add(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    "D:\",
    NULL,
    FILE_TIME_LAST_WRITE,
    &l__PVOID__Root,
    &l__PVOID__NewNode
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
```

```
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.208 StarBurn_ISO9660JolietFileTree_AddEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_AddEx(
    OUT PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCHAR p__PCHAR__RootDirectoryAbsolutePathAndName,
    IN PCHAR p__PCHAR__RootDirectoryNewName,
    IN FILE_TIME p__FILE_TIME,
    IN PFILETIME p__PFILETIME,
    IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent,
    OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN PCHAR p__PCHAR__RootDirectoryAbsolutePathAndName	Pointer to directory name to build the ISO9660 or Joliet image from. This parameters can be NULL (but in this case p__PCHAR__RootDirectoryNewName cannot!!!) if virtual (non-present on the disk) directory must be added to the image.
IN PCHAR p__PCHAR__RootDirectoryNewName	Pointer to directory name to be stored in the ISO9660 or Joliet image as root name, NULL to include default name from the previous parameter.
IN FILE_TIME p__FILE_TIME	File time that will be included in file system image (See FILE_TIME (see page 490)).
IN PFILETIME p__PFILETIME	Pointer to FILETIME structure created node would have.
IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent	Pointer to pointer to the file tree node we'll use as parent. This is the result of either tree walking with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() or StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)(). NULL can be passed instead of the result of StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild	Pointer to pointer to the new created file tree node.

Returns

Execution status. EN_SUCCESS if the tree was reallocated successfully, E_MEMORY_ALLOCATION_FAILED if there is not enough memory to reallocate the tree. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.

Remarks

Please see the BuildImage sample that will demonstrate how progress indication can be build and what parameters in general are passed into the callback function during the operations with file tree object and how extended variant of adding call could be used to solve the issue of passing the node with required creation times.

See Also

StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), FILE_TREE (see page 490), FILE_TIME (see page 490), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_Add (see page 272)

Example

This example allocates Joliet file tree with required time for root node and destroys the tree after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__Root;
SYSTEMTIME l__SYSTEMTIME;
FILETIME l__FILETIME;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get root here
l__PVOID__Root =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct reply
if ( l__PVOID__Root == NULL )
{
// Handle error here, keep in mind that root CAN be NULL if nothing was added to tree
before...
```

```
}

// Format system time here

ZeroMemory(
    &l__SYSTEMTIME,
    sizeof( l__SYSTEMTIME )
    );

l__SYSTEMTIME.wYear = 1991;

l__SYSTEMTIME.wMonth = 3;

l__SYSTEMTIME.wDayOfWeek = 1;

l__SYSTEMTIME.wDay = 1;

l__SYSTEMTIME.wHour = 2;

l__SYSTEMTIME.wMinute = 30;

l__SYSTEMTIME.wSecond = 40;

l__SYSTEMTIME.wMilliseconds = 10;

// Covert system time to file time
SystemTimeToFileTime(
    &l__SYSTEMTIME,
    &l__FILETIME
    );

// Try to add all from the root directory of the drive D: to already created Joliet file
tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_AddEx(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    "D:\",
    NULL,
    FILE_TIME_LAST_WRITE,
    &l__FILETIME,
    &l__PVOID__Root,
    &l__PVOID__NewNode
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.209 StarBurn_ISO9660JolietFileTree_AddMemory Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_AddMemory(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PUCHAR p__PUCHAR__MemoryRegion,
    IN LARGE_INTEGER p__LARGE_INTEGER__MemoryRegionSizeInUCHARs,
    IN PCHAR p__PCHAR__Name,
    IN FILE_TIME p__FILE_TIME,
    IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent,
    OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN PUCHAR p__PUCHAR__MemoryRegion	Pointer to memory region to create node from.
IN LARGE_INTEGER p__LARGE_INTEGER__MemoryRegionSizeInUCHARs	Memory region size in UCHARs.
IN FILE_TIME p__FILE_TIME	File time that will be included in file system image (See FILE_TIME (see page 490)).
IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent	Pointer to pointer to the file tree node we'll use as parent. This is the result of either tree walking with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() or StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)(). NULL can be passed instead of the result of StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild	Pointer to pointer to the new created file tree node.
p__PCHAR__RootDirectoryNewName	Pointer to directory name to be stored in the ISO9660 or Joliet image as root name. Cannot be NULL in this case!

Returns

Execution status. EN_SUCCESS if the tree was reallocated successfully, E_MEMORY_ALLOCATION_FAILED if there is not enough memory to reallocate the tree. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function adds the node created from the memory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.

Remarks

Please see the BuildImage sample that will demonstrate how memory node can be added to already created ISO9660 or Joliet file tree, progress indication can be build and what parameters in general are passed into the callback function during the operations with file tree object.

See Also

StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), FILE_TREE (see page 490), FILE_TIME (see page 490), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)

Example

This example allocates Joliet file tree, adds memory node to it and destroys it after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__Root;
UCHAR l__UCHAR__MemoryNode[ 4096 ];
LARGE_INTEGER l__LARGE_INTEGER__MemoryNodeSizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get root here
l__PVOID__Root =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct reply
if ( l__PVOID__Root == NULL )
{
// Handle error here, keep in mind that root CAN be NULL if nothing was added to tree
before...
}

// Try to add all from the root directory of the drive D: to already created Joliet file
tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Add(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    "D:\",
    NULL,
    FILE_TIME_LAST_WRITE,
    &l__PVOID__Root,
    &l__PVOID__NewNode
```

```
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Prepare memory node memory
RtlZeroMemory(
    &l__UCHAR__MemoryNode,
    sizeof( l__UCHAR__MemoryNode )
    );

// Format memory node here and store all required data into l__UCHAR__MemoryNode buffer

// Set size of memory node
l__LARGE_INTEGER__MemoryNodeSizeInUCHARs.QuadPart = sizeof( l__UCHAR__MemoryNode );

// Try to add memory node as "MemoryNodeName" to already created Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_AddMemory(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PUCHAR )( &l__UCHAR__MemoryNode ),
    l__LARGE_INTEGER__MemoryNodeSizeInUCHARs,
    "MemoryNodeName",
    FILE_TIME_LAST_WRITE,
    &l__PVOID__Root,
    &l__PVOID__NewNode
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.210 StarBurn_ISO9660JolietFileTree_AddW Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_AddW(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PWCHAR p__PWCHAR__RootDirectoryOrFileAbsolutePathAndName,
    IN PWCHAR p__PWCHAR__RootDirectoryOrFileNewName,
    IN FILE_TIME p__FILE_TIME,
    IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent,
    OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN FILE_TIME p__FILE_TIME	File time that will be included in file system image (See FILE_TIME (see page 490)).
IN PVOID * p__PPVOID__ISO9660JolietFileTreeNode__Parent	Pointer to pointer to the file tree node we'll use as parent. This is the result of either tree walking with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() or StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)(). NULL can be passed instead of the result of StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode__NewChild	Pointer to pointer to the new created file tree node.
p__PCHAR__RootDirectoryAbsolutePathAndName	Pointer to directory name to build the ISO9660 or Joliet image from. This parameters can be NULL (but in this case p__PCHAR__RootDirectoryNewName cannot!!!) if virtual (non-present on the disk) directory must be added to the image.
p__PCHAR__RootDirectoryNewName	Pointer to directory name to be stored in the ISO9660 or Joliet image as root name, NULL to include default name from the previous parameter.

Returns

Execution status. EN_SUCCESS if the tree was reallocated successfully, E_MEMORY_ALLOCATION_FAILED if there is not enough memory to reallocate the tree. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations. Unlike other tree management calls this one deals with Unicode names (wide char variant of basic call).

Remarks

Please see the BuildImage sample that will demonstrate how progress indication can be build and what parameters in general are passed into the callback function during the operations with file tree object. BuildImage sample shows how to add files using both ANSI and Unicode based naming conventions.

See Also

StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), FILE_TREE (see page 490), FILE_TIME (see page 490), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_AddEx (see page 274)

Example

This example allocates Joliet file tree and destroys it after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__Root;

// Prepare exception text buffer
RtlZeroMemory(
```

```
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to get root here
l__PVOID__Root =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct reply
if ( l__PVOID__Root == NULL )
{
// Handle error here, keep in mind that root CAN be NULL if nothing was added to tree
before...
}

// Try to add all from the root directory of the drive D: to already created Joliet file
tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_AddW(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    L"D:\",
    NULL,
    FILE_TIME_LAST_WRITE,
    &l__PVOID__Root,
    &l__PVOID__NewNode
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.211 StarBurn_ISO9660JolietFileTree_AddZeroFile Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_AddZeroFile(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN LARGE_INTEGER p__LARGE_INTEGER__FileSizeInUCHARs,
    IN PCHAR p__PCHAR__Name,
    IN FILE_TIME p__FILE_TIME,
    IN PVOID* p__PPVOID__ISO9660JolietFileTreeNode__Parent,
    OUT PVOID* p__PPVOID__ISO9660JolietFileTreeNode__NewChild
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN LARGE_INTEGER p__LARGE_INTEGER__FileSizeInUCHARs	Size of the added file.
IN PCHAR p__PCHAR__Name	Pointer to file name.
IN FILE_TIME p__FILE_TIME	File time that will be included in file system image (See FILE_TIME (see page 490)).
IN PVOID* p__PPVOID__ISO9660JolietFileTreeNode__Parent	Pointer to pointer to the file tree node we'll use as parent. This is the result of either tree walking with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() or StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)(). NULL can be passed instead of the result of StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PVOID* p__PPVOID__ISO9660JolietFileTreeNode__NewChild	Pointer to pointer to the new created file tree node.

Returns

Execution status. EN_SUCCESS if the tree was reallocated successfully, E_MEMORY_ALLOCATION_FAILED if there is not enough memory to reallocate the tree. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function adds the new "filled with zeros" file to already created ISO9660 or Joliet file tree. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.

Remarks

Please see the BuildImage sample that will demonstrate how progress indication can be build and what parameters in general are passed into the callback function during the operations with file tree object.

See Also

StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), FILE_TREE (see page 490), FILE_TIME (see page 490), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_AddEx (see page

274), StarBurn_ISO9660JolietFileTree_Add (see page 272)

2.1.212 StarBurn_ISO9660JolietFileTree_BuildImage Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_BuildImage(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN LONG p__LONG__StartingLBA,
    IN LONG p__LONG__TreeLevelToProcess,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN PCHAR p__PCHAR__VolumeSetName,
    IN PCHAR p__PCHAR__PublisherPreparerName,
    IN PCHAR p__PCHAR__ApplicationName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN LONG p__LONG__StartingLBA	Starting LBA to build the image from (it should be the next writable address of the track on the CD/DVD/Blu-Ray/HD-DVD media, 0 for empty disc).
IN LONG p__LONG__TreeLevelToProcess	Top tree level to process, levels that exceed this value will be ignored and not included into the file system image. For now this parameter is ignored and whole tree will be put into the image.
IN BOOLEAN p__BOOLEAN__IsXA	Is CDROM XA or CDROM data (do we need CDROM XA marker to be included to the the image).
IN PCHAR p__PCHAR__VolumeSetName	Pointer to the volume set.
IN PCHAR p__PCHAR__PublisherPreparerName	Pointer to the publisher and preparer name.
p__PCHAR__ApplictionName	Pointer to the application name.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function builds ISO9660 or Joliet file system image from ISO9660 or Joliet file tree (assign all the logical block offsets and allocates and initializes all file system internal structures). Later this tree can be used to store the "virtual" ISO9660 or Joliet image to the file on the disk or to burn this ISO9660 or Joliet image directly to the CD/DVD/Blu-Ray/HD-DVD media.

Remarks

Please see the BuildImage sample that will demonstrate how progress indication can be created and what parameters in general are passed into the callback function during the operations with file tree object. See ISO9660_XXX constants to verify correct string parameter lengths.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_Destroy (see page 214), PCALLBACK (see page

582), EXCEPTION_NUMBER (see page 487), StarBurn_ISO9660JolietFileTree_BuildImageEx (see page 285)

Example

This example allocates Joliet file tree from the directory, builds the image and destroys the file tree after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to build the image, start with LBA 0 and include up to 8 levels into the image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_BuildImage(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    0,
    8,
    FALSE,
    "Volume",
    "Publisher",
    "Application"
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.213 StarBurn_ISO9660JolietFileTree_BuildImageEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_BuildImageEx(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN LONG p__LONG__StartingLBA,
    IN LONG p__LONG__TreeLevelToProcess,
    IN BOOLEAN p__BOOLEAN__IsXA,
    IN PCHAR p__PCHAR__VolumeSetName,
    IN PCHAR p__PCHAR__PublisherPreparerName,
    IN PCHAR p__PCHAR__ApplicationName,
    IN PCHAR p__PCHAR__SystemID,
    IN PCHAR p__PCHAR__CopyrightFile,
    IN PCHAR p__PCHAR__BibliographicFile,
    IN PCHAR p__PCHAR__CreationDate,
    IN PCHAR p__PCHAR__ModificationDate,
    IN PCHAR p__PCHAR__ExpirationDate,
    IN PCHAR p__PCHAR__EffectiveDate
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN LONG p__LONG__StartingLBA	Starting LBA to build the image from (it should be the next writable address of the track on the CD/DVD/Blu-Ray/HD-DVD media, 0 for empty disc).
IN LONG p__LONG__TreeLevelToProcess	Top tree level to process, levels that exceed this value will be ignored and not included into the file system image. For now this parameter is ignored and whole tree will be put into the image.
IN BOOLEAN p__BOOLEAN__IsXA	Is CDROM XA or CDROM data (do we need CDROM XA marker to be included to the the image).
IN PCHAR p__PCHAR__VolumeSetName	Pointer to the volume set.
IN PCHAR p__PCHAR__PublisherPreparerName	Pointer to the publisher and preparer name.
IN PCHAR p__PCHAR__SystemID	Pointer to system ID.
IN PCHAR p__PCHAR__CopyrightFile	Pointer to copyright file.
IN PCHAR p__PCHAR__BibliographicFile	Pointer to bibliographic file.
IN PCHAR p__PCHAR__CreationDate	Pointer to creation date.
IN PCHAR p__PCHAR__ModificationDate	Pointer to modification date.
IN PCHAR p__PCHAR__ExpirationDate	Pointer to expiration date.
IN PCHAR p__PCHAR__EffectiveDate	Pointer to effective date.
p__PCHAR__ApplictionName	Pointer to the application name.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function builds ISO9660 or Joliet file system image from ISO9660 or Joliet file tree (assign all the logical block offsets

and allocates and initializes all file system internal structures). Later this tree can be used to store the "virtual" ISO9660 or Joliet image to the file on the disk or to burn this ISO9660 or Joliet image directly to the CD/DVD/Blu-Ray/HD-DVD media. It's identical to StarBurn_ISO9660JolietFileTree_BuildImage (see page 283) except has some extra parameter to put into file system descriptor.

Remarks

Please see the BuildImage sample that will demonstrate how progress indication can be created and what parameters in general are passed into the callback function during the operations with file tree object. StarBurn_ISO9660JolietFileTree_BuildImageEx can be used instead of StarBurn_ISO9660JolietFileTree_BuildImage (see page 283). See ISO9660_XXX constants to verify correct string parameter lengthes.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), StarBurn_ISO9660JolietFileTree_BuildImage (see page 283)

Example

This example allocates Joliet file tree from the directory, builds the image and destroys the file tree after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to build the image, start with LBA 0 and include up to 8 levels into the image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_BuildImageEx(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    0,
    8,
    FALSE,
    "Volume",
    "Publisher",
    "Application",
    "SystemID",
    "CopyFile",
```

```
    "BiblioFile",
    "CreateDate",
    "ModifyDate",
    "ExpireDate",
    "EffectDate"
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.214 StarBurn_ISO9660JolietFileTree_Cancel Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_ISO9660JolietFileTree_Cancel(
    IN PVOID p__PVOID__ISO9660JolietFileTree
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().

Returns

Nothing. This function cannot fail.

Description

This function cancel ISO9660 or Joliet file tree creation process.

Remarks

Can cancel only the process initiated by call to StarBurn_ISO9660Joliet_Add, StarBurn_ISO9660JolietFileTree_BuildImage (see page 283), StarBurn_ISO9660JolietFileTree_Read (see page 318) and StarBurn_ISO9660JolietFileTree_SeekToBegin (see page 322) time-expensive actions cannot be canceled with this code. Also it must be noted that StarBurn_ISO9660JolietFileTree_Cancel is supposed to be executed from the different thread that the one that calls StarBurn_ISO9660JolietFileTree_Add (see page 272) (original thread will be blocked processing it's call).

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_Read (see page 318), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_SeekToBegin (see page 322)

Example

This example allocates Joliet file tree, initiates tree creation, cancels it and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
```

```
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Cancel tree creation process. Keep in mind that this code must be called from the
different thread
StarBurn_ISO9660JolietFileTree_Cancel( l__PVOID__FileTree );

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.215 StarBurn_ISO9660JolietFileTree_Create Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_Create(
    OUT PVOID * p__PPVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCALLBACK p__PCALLBACK,
    IN PVOID p__PVOID__CallbackContext,
    IN BOOLEAN p__BOOLEAN__IsInvalidKidIgnore,
    IN BOOLEAN p__BOOLEAN__IsLocked,
    IN BOOLEAN p__BOOLEAN__IsLevel2,
    IN FILE_TREE p__FILE_TREE
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID * p__PPVOID__ISO9660JolietFileTree	Pointer to pointer to the object that toolkit will set to the ISO9660 or Joliet file tree object it will allocate.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN PCALLBACK p__PCALLBACK	Pointer to callback function that will be called from inside the toolkit to inform the user about the progress of this operation (See samples how to use this callback for progress indication and other useful things).
IN PVOID p__PVOID__CallbackContext	Pointer to arbitrary context value that will be passed as first PVOID parameter to the called callback function.
IN BOOLEAN p__BOOLEAN__IsInvalidKidIgnore	TRUE to ignore bad kids (files or directories that cannot be accessed), FALSE to abort on every unsuccessful attempt to access such a files or directories.
IN BOOLEAN p__BOOLEAN__IsLocked	TRUE of all files of the file tree will be locked, FALSE if they will not be locked (their file handles will not be opened all the times - FALSE must be specified if running under Win9x OSes that have opened files limitation).
IN BOOLEAN p__BOOLEAN__IsLevel2	TRUE if Level2 ISO9660 names generation is used, FALSE if DOS 8.3 format for names generation is used.
IN FILE_TREE p__FILE_TREE	Type of file tree to build, ISO9660 or Joliet (See FILE_TREE (see page 490) for details).

Returns

Execution status. EN_SUCCESS if the tree was allocated successfully, E_MEMORY_ALLOCATION_FAILED if there is not enough memory to allocate the tree. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function allocates in the memory and initializes ISO9660 or Joliet file tree. Later this file tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be recorder directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate buffering operations on the hard disk.

Remarks

Please see the BuildImage sample that will demonstrate how progress indication can be build and what parameters in general are passed into the callback function during the operations with file tree object.

See Also

StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), FILE_TREE (see page 490), FILE_TIME (see page 490), StarBurn_ISO9660JolietFileTree_Add (see page 272)

Example

This example allocates Joliet file tree and destroys it after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
```

```
    &l__ULONG__Status,
    ( PCALLBACK )( Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    TRUE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.216 StarBurn_ISO9660JolietFileTree_GetAttributes Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_ISO9660JolietFileTree_GetAttributes(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().

Returns

Attributes. This function cannot fail.

Combination of one or more of following attributes is valid (see file attributes values in Windows SDK):

FILE_ATTRIBUTE_ARCHIVE FILE_ATTRIBUTE_COMPRESSED FILE_ATTRIBUTE_DIRECTORY FILE_ATTRIBUTE_NORMAL FILE_ATTRIBUTE_HIDDEN FILE_ATTRIBUTE_READONLY FILE_ATTRIBUTE_SYSTEM FILE_ATTRIBUTE_TEMPORARY

Description

This function returns ISO9660 or Joliet file tree node attributes.

Remarks

Attributes can be used to build correct tree to visualize enumeration of all the kids and walk down the tree.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)

Example

This example allocates Joliet file tree, get root node, gets attributes and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
ULONG l__ULONG__Attributes;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get attributes node here
l__ULONG__Attributes =
StarBurn_ISO9660JolietFileTree_GetAttributes(
    l__PVOID__FileTree,
    l__PVOID__RootNode
    );

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
```

```
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.217 StarBurn_ISO9660JolietFileTree_GetAutoSorting Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_ISO9660JolietFileTree_GetAutoSorting(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().

Returns

Current auto-sorting mode. This function cannot fail.

Description

This function returns ISO9660 or Joliet file tree node sorting mode.

Remarks

Auto sorting mode allows sorting sub-nodes before adding it to the image. It should be turned off if you want to specify your own order of adding files. File order determines how the LBAs would be assigned. Lower LBA would be assigned to the last added file.

See Also

StarBurn_ISO9660JolietFileTree_SetSorting, StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214)

2.1.218 StarBurn_ISO9660JolietFileTree_GetFileSizeInUCHARs Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_ISO9660JolietFileTree_GetFileSizeInUCHARs(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PLARGE_INTEGER p__PLARGE_INTEGER__FileSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)(). The function is valid only for nodes that correspond to files.
OUT PLARGE_INTEGER p__PLARGE_INTEGER__FileSizeInUCHARs	Pointer to the variable to receive ISO9660 or Joliet file tree node file size in UCHARs.

Returns

Execution status.

Description

This function returns ISO9660 or Joliet file tree file size in UCHARs.

Remarks

This call is identical to the StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)(...) except it has a bit different semantics.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)

Example

This example allocates Joliet file tree, get root node, gets file size in UCHARs and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
LARGE_INTEGER l__LARGE_INTEGER__FileSizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );
```

```
// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get file size in UCHARs here
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_GetFileSizeInUCHARs(
    l__PVOID__FileTree,
    l__PVOID__RootNode,
    &l__LARGE_INTEGER__FileSizeInUCHARs
    );

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.219 StarBurn_ISO9660JolietFileTree_GetFirstKid Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_ISO9660JolietFileTree_GetFirstKid(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().

Returns

Pointer to first kid node. NULL if no first kid node present.

Description

This function returns ISO9660 or Joliet file tree node first kid pointer.

Remarks

Kid node can be used to enumerate all the kids and walk down the tree.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)

Example

This example allocates Joliet file tree, get root node, gets first kid and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
PVOID l__PVOID__FirstKidNode;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get first kid node here
l__PVOID__FirstKidNode =
StarBurn_ISO9660JolietFileTree_GetFirstKid(
    l__PVOID__FileTree,
    l__PVOID__RootNode
    );

// Check for correct response
if ( l__PVOID__FirstKidNode == NULL )
{
```

```
// Handle error here...
}

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.220 StarBurn_ISO9660JolietFileTree_GetFullPath Function

C++

```
__stdcall STARBURN_IMPEX_API ISO9660_KIDTYPE StarBurn_ISO9660JolietFileTree_GetFullPath(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PCHAR p__PCHAR__FullName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree current node belongs to.
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to ISO9660 or Joliet file tree node to get ISO9660 time stamp for.
OUT PCHAR p__PCHAR__FullName	Pointer to the buffer to receive full path to ISO9660 or Joliet file tree node.

Returns

Kid type of ISO9660 or Joliet file tree node (where this node was taken from). This call cannot fail.

Description

This function returns full path to ISO9660 or Joliet file tree node.

Remarks

None.

See Also

StarBurn_ISO9660JolietFileTree_GetNodeSystemTime (see page 308), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)

Example

There are no examples for StarBurn_ISO9660JolietFileTree_GetFullPath(...) API call.

2.1.221 StarBurn_ISO9660JolietFileTree_GetKidType Function

C++

```
__stdcall STARBURN_IMPEX_API ISO9660_KIDTYPE StarBurn_ISO9660JolietFileTree_GetKidType(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree current node belongs to.
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to ISO9660 or Joliet file tree node to get ISO9660 time stamp for.

Returns

Kid type of ISO9660 or Joliet file tree node (where this node was taken from). This call cannot fail.

Description

This function returns type of ISO9660 or Joliet file tree node.

Remarks

None.

See Also

StarBurn_ISO9660JolietFileTree_GetNodeSystemTime (see page 308), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)

2.1.222 StarBurn_ISO9660JolietFileTree_GetLevel Function

C++

```
__stdcall STARBURN_IMPEX_API LONG StarBurn_ISO9660JolietFileTree_GetLevel(
    IN PVOID p__PVOID__ISO9660JolietFileTree
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().

Returns

Tree level. This function cannot fail.

Description

This function returns ISO9660 or Joliet file tree level. The number of levels in the created tree.

Remarks

The number of levels ISO9660 or Joliet can handle is limited to 8. Any values that exceed 8 is under your own risk.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_Destroy (see page 214)

Example

This example allocates Joliet file tree, checks tree level and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
LONG l__LONG__TreeLevel;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get tree level here
l__LONG__TreeLevel =
StarBurn_ISO9660JolietFileTree_GetLevel( l__PVOID__FileTree );

// Check for tree level here
if ( l__LONG__TreeLevel > 8 )
{
// Handle error here...
}

// Do something with Joliet file tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
```

```
}
```

2.1.223 StarBurn_ISO9660JolietFileTree_GetNames Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_ISO9660JolietFileTree_GetNames(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PCHAR p__PCHAR__AbsoluteName,
    OUT PCHAR p__PCHAR__LongName,
    OUT PCHAR p__PCHAR__ShortName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PCHAR p__PCHAR__AbsoluteName	Pointer to the array of CHARs MAX_PATH in size that will be filled with absolute name.
OUT PCHAR p__PCHAR__LongName	Pointer to the array of CHARs MAX_PATH in size that will be filled with long name.
OUT PCHAR p__PCHAR__ShortName	Pointer to the array of CHARs MAX_PATH in size that will be filled with short name.

Returns

Nothing. Passed buffers will be filled with the names. This function cannot fail.

Description

This function returns ISO9660 or Joliet file tree node names (long, short and absolute).

Remarks

Names can be used to build correct tree to visualize enumeration of all the kids and walk down the tree.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetAttributes (see page 290), StarBurn_ISO9660JolietFileTree_SetNames (see page 330)

Example

This example allocates Joliet file tree, get root node, gets names and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
CHAR l__CHAR__AbsoluteName[ MAX_PATH ];
CHAR l__CHAR__LongName[ MAX_PATH ];
```

```
CHAR l__CHAR__ShortName[ MAX_PATH ];

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get names here
StarBurn_ISO9660JolietFileTree_GetNames(
    l__PVOID__FileTree,
    l__PVOID__RootNode,
    ( PCHAR )( &l__CHAR__AbsoluteName ),
    ( PCHAR )( &l__CHAR__LongName ),
    ( PCHAR )( &l__CHAR__ShortName )
    );

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.224 StarBurn_ISO9660JolietFileTree_GetNamesEx Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_ISO9660JolietFileTree_GetNamesEx(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
```

```
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PCHAR p__PCHAR__AbsoluteName,
    OUT PCHAR p__PCHAR__LongName,
    OUT PCHAR p__PCHAR__ShortName,
    OUT PCHAR p__PCHAR__FileSystemName,
    OUT PUSHORT p__PUSHORT__UnicodeFileSystemName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PCHAR p__PCHAR__AbsoluteName	Pointer to the array of CHARs MAX_PATH in size that will be filled with absolute name.
OUT PCHAR p__PCHAR__LongName	Pointer to the array of CHARs MAX_PATH in size that will be filled with long name.
OUT PCHAR p__PCHAR__ShortName	Pointer to the array of CHARs MAX_PATH in size that will be filled with short name.
OUT PCHAR p__PCHAR__FileSystemName	Pointer to the array of CHARs MAX_PATH in size that will be filled with ISO9660 name.
OUT PUSHORT p__PUSHORT__UnicodeFileSystemName	Pointer to the array of CHARs MAX_PATH in size that will be filled with Joliet name.

Returns

Nothing. Passed buffers will be filled with the names. This function cannot fail.

Description

This function returns ISO9660 or Joliet file tree node names (long, short and absolute). Also it returns file system dependent names in ANSI and Unicode.

Remarks

Names can be used to build correct tree to visualize enumeration of all the kids and walk down the tree.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetAttributes (see page 290), StarBurn_ISO9660JolietFileTree_SetNames (see page 330)

Example

This example allocates Joliet file tree, get root node, gets names and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
CHAR l__CHAR__AbsoluteName[ MAX_PATH ];
CHAR l__CHAR__LongName[ MAX_PATH ];
CHAR l__CHAR__ShortName[ MAX_PATH ];
CHAR l__CHAR__FileSystemName[ MAX_PATH ];
WCHAR l__WCHAR__FileSystemName[ MAX_PATH ];

// Prepare exception text buffer
RtlZeroMemory(
```

```
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get names here
StarBurn_ISO9660JolietFileTree_GetNamesEx(
    l__PVOID__FileTree,
    l__PVOID__RootNode,
    ( PCHAR )( &l__CHAR__AbsoluteName ),
    ( PCHAR )( &l__CHAR__LongName ),
    ( PCHAR )( &l__CHAR__ShortName ),
    ( PCHAR )( &l__CHAR__FileSystemName ),
    ( PUSHORT )( &l__WCHAR__FileSystemName )
    );

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.225 StarBurn_ISO9660JolietFileTree_GetNamesW Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_ISO9660JolietFileTree_GetNamesW(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PWCHAR p__PWCHAR__AbsoluteName,
```

```
    OUT PWCHAR p__PWCHAR__LongName,
    OUT PWCHAR p__PWCHAR__ShortName
);
```

File

StarBurn.h (see page 662)

Description

StarBurn_ISO9660JolietFileTree_GetNamesW function is not documented

2.1.226 StarBurn_ISO9660JolietFileTree_GetNextKid Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_ISO9660JolietFileTree_GetNextKid(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode__Root,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode__PreviousKid
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PVOID p__PVOID__ISO9660JolietFileTree`	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
`IN PVOID p__PVOID__ISO9660JolietFileTreeNode__Root`	Pointer to the ISO9660 or Joliet file tree node object that is result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid().
`IN PVOID p__PVOID__ISO9660JolietFileTreeNode__PreviousKid`	the result of call to previous call to StarBurn_ISO9660JolietFileTreeNode__GetFirstKid() or StarBurn_ISO9660JolietFileTree_GetNextKid().

Returns

Pointer to next kid node. NULL if no next kid node present.

Description

This function returns ISO9660 or Joliet file tree root node next kid pointer.

Remarks

Kid node can be used to enumerate all the kids and walk down the tree.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)

Example

This example allocates Joliet file tree, get root node, gets first kid and next kid and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
```

```
PVOID l__PVOID__FirstKidNode;
PVOID l__PVOID__NextKidNode;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get first kid node node here
l__PVOID__FirstKidNode =
StarBurn_ISO9660JolietFileTree_GetFirstKid(
    l__PVOID__FileTree,
    l__PVOID__RootNode
    );

// Check for correct response
if ( l__PVOID__FirstKidNode == NULL )
{
// Handle error here...
}

// Get next kid node node here
l__PVOID__NextKidNode =
StarBurn_ISO9660JolietFileTree_GetNextKid(
    l__PVOID__FileTree,
    l__PVOID__RootNode,
    l__PVOID__FirstKidNode
    );

// Check for correct response
if ( l__PVOID__NextKidNode == NULL )
{
// Handle error here...
}

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );
```

```
// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.227 StarBurn_ISO9660JolietFileTree_GetNodeISO9660DateTime Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_ISO9660JolietFileTree_GetNodeISO9660DateTime(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PISO9660_DATE_TIME p__PISO9660_DATE_TIME
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree current node belongs to.
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to ISO9660 or Joliet file tree to get ISO9660 time stamp for.
OUT PISO9660_DATE_TIME p__PISO9660_DATE_TIME	Pointer to ISO9660 time stamp (date and time).

Returns

Execution status.

Description

This function returns ISO9660 or Joliet file tree node ISO9660 time stamp (date and time).

Remarks

None.

See Also

StarBurn_ISO9660JolietFileTree_GetNodeSystemTime (see page 308), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)

Example

There are no examples for StarBurn_ISO9660JolietFileTree_GetNodeISO9660DateTime(...) API call.

2.1.228 StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs Function

C++

```
__stdcall STARBURN_IMPEX_API LONG StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PLONG p__PLONG__HighPartSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PLONG p__PLONG__HighPartSizeInUCHARs	Pointer to the variable to receive high part (32 bits) of the node power (file size) in UCHARs.

Returns

Low part (32 bits) of the node power (file size) in UCHARs. This function cannot fail.

Description

This function returns ISO9660 or Joliet file tree node power (file size) in UCHARs.

Remarks

Node power (file size) in UCHARs can be used to build correct tree to visualize enumeration of all the kids and walk down the tree. This call is identical to the StarBurn_ISO9660JolietFileTree_GetFileSizeInUCHARs (see page 292)(...) except the semantics.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetFileSizeInUCHARs (see page 292)

Example

This example allocates Joliet file tree, get root node, gets node power in UCHARs and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
LARGE_INTEGER l__LARGE_INTEGER__NodePowerInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
```

```
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get node power in UCHARs here
l__LARGE_INTEGER__NodePowerInUCHARs.LowPart =
StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs(
    l__PVOID__FileTree,
    l__PVOID__RootNode,
    ( PLONG )( &l__LARGE_INTEGER__NodePowerInUCHARs.HighPart )
    );

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.229
StarBurn_ISO9660JolietFileTree_GetNodeStartingLBA Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_ISO9660JolietFileTree_GetNodeStartingLBA(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PLONG p__PLONG__StartingLBA
```

```
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
OUT PLONG p__PLONG__StartingLBA	Pointer to the variable to receive ISO9660 or Joliet file tree node starting LBA.

Returns

Execution status.

Description

This function returns ISO9660 or Joliet file tree node starting LBA after the image was built.

Remarks

None.

See Also

StarBurn_ISO9660JolietFileTree_GetNodeSystemTime (see page 308), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)

Example

There are no examples for StarBurn_ISO9660JolietFileTree_GetNodeISO9660DateTime (see page 305)(...) API call.

2.1.230 StarBurn_ISO9660JolietFileTree_GetNodeSystemTime Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER
StarBurn_ISO9660JolietFileTree_GetNodeSystemTime(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    OUT PSYSTEMTIME p__PSYSTEMTIME
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree current node belongs to.
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to ISO9660 or Joliet file tree to get SYSTEMTIME for.
OUT PSYSTEMTIME p__PSYSTEMTIME	Pointer to the output SYSTEMTIME for ISO9660 or Joliet file tree node.

Returns

Execution status.

Description

This function returns ISO9660 or Joliet file tree node SYSTEMTIME.

Remarks

None.

See Also

StarBurn_ISO9660JolietFileTree_GetNodeISO9660DateTime (see page 305), StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)

Example

There are no examples for StarBurn_ISO9660JolietFileTree_GetNodeSystemTime(...) API call.

2.1.231 StarBurn_ISO9660JolietFileTree_GetParent Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_ISO9660JolietFileTree_GetParent(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().

Returns

Pointer to parent node. NULL if no parent node present (this is root node).

Description

This function returns ISO9660 or Joliet file tree node parent pointer.

Remarks

Parent node can be used to enumerate all the kids and walk down the tree.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)

Example

This example allocates Joliet file tree, get root node, gets parent and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
PVOID l__PVOID__ParentNode;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get parent node here
l__PVOID__ParentNode =
StarBurn_ISO9660JolietFileTree_GetParent(
    l__PVOID__FileTree,
    l__PVOID__RootNode
    );

// Check for correct response (MUST be NULL)
if ( l__PVOID__ParentNode == NULL )
{
// Handle error here...
}

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
```

```
{
// Handle error here...
}
```

2.1.232 StarBurn_ISO9660JolietFileTree_GetRoot Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_ISO9660JolietFileTree_GetRoot(
    IN PVOID p__PVOID__ISO9660JolietFileTree
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PVOID p__PVOID__ISO9660JolietFileTree`	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().

Returns

Pointer to root node. NULL if no root present.

Description

This function returns ISO9660 or Joliet file tree root node pointer.

Remarks

Root can be used to enumerate all the root node kids and walk down the tree. Attention! This function will return NULL if nothing was added to the tree by StarBurn_ISO9660JolietFileTree_Add (see page 272)() before.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_Destroy (see page 214)

Example

This example allocates Joliet file tree, get root node and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );
```

```
// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.233 StarBurn_ISO9660JolietFileTree_GetSizeInUCHARs Function

C++

```
__stdcall STARBURN_IMPEX_API LONG StarBurn_ISO9660JolietFileTree_GetSizeInUCHARs(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PLONG p__PLONG__HighPartSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288) and StarBurn_ISO9660JolietFileTree_BuildImage (see page 283) was applied to later.
p__PULONG__HighPartSizeInUCHARs	Pointer to the 32-bit unsigned long that will receive high part of 64-bit image size.

Returns

Built ISO9660 or Joliet file system image size in UCHARs. This function cannot fail.

Description

This function returns ISO9660 or Joliet built file system image size in UCHARs.

Remarks

Used to determine the size of the built file system image so it will be clear will it fit on the CD/DVD/Blu-Ray/HD-DVD media (if burning) or on hard disk (if storing file system image as file on the disk) or not.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_BuildImage (see page 283), StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487)

Example

This example allocates Joliet file tree from the directory, builds file system image, gets file system image size in UCHARs, checks for compatibility and destroys the file tree after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
LARGE_INTEGER l__LARGE_INTEGER__FileSystemImageSizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to build the image, start with LBA 0 and include up to 8 levels into the image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_BuildImage(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    0,
    8,
    FALSE,
    "Volume",
    "Publisher",
    "Application"
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Get file system image size in UCHARs
l__LARGE_INTEGER__FileSystemImageSizeInUCHARs.LowPart =
StarBurn_ISO9660JolietFileTree_GetSizeInUCHARs(
    l__PVOID__FileTree,
    &l__LARGE_INTEGER__FileSystemImageSizeInUCHARs.HighPart
    );
```

```
// Check for enough space on the media here (either hard disk or CD/DVD/Blu-Ray/HD-DVD)
if ( l__LARGE_INTEGER__FileSystemImageSizeInUCHAR.QuadPart < ... )
{
// Handle error here...
}

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.234 StarBurn_ISO9660JolietFileTree_ImportFile Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_ImportFile(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PCHAR p__PCHAR__ImportFileName,
    IN BOOLEAN p__BOOLEAN__ImportJolietStructures,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)().
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree to import track to.
IN PCHAR p__PCHAR__ImportFileName	Pointer to the file name to import file system structures from.
IN BOOLEAN p__BOOLEAN__ImportJolietStructures	Should we try to import Joliet structures (TRUE) or proceed with ISO9660 only (FALSE).
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text buffer.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function imports file for created ISO9660 or Joliet file tree.

Remarks

Please see the TrackAtOnceFromTreeWithImportFromFile sample that will demonstrate how ISO9660 or Joliet file system

image can be created, existing file imported and whole result burn to the CD/DVD/Blu-Ray/HD-DVD media.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree (see page 40), StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree (see page 46), StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141), StarBurn_CdvdBurnerGrabber_ImportTrack

Example

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__ISO9660JolietFileTree;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Create and add something to ISO9660 file tree here

// Try to import file system structures from the file
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_ImportFile(
    l__PVOID__CdvdBurnerGrabber,
    l__PVOID__ISO9660JolietFileTree,
    "Image.ISO", // Import FS structures from file called "Image.ISO"
    TRUE, // Try to proceed as Joliet
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError
    );
```

```
// Check for correct reply
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.235 StarBurn_ISO9660JolietFileTree_ImportTrack Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_ImportTrack(
    IN PVOID p__PVOID__CdvdBurnerGrabber,
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN UCHAR p__UCHAR__TrackNumber,
    IN BOOLEAN p__BOOLEAN__ImportJolietStructures,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CdvdBurnerGrabber	Pointer to the CdvdBurnerGrabber object that toolkit allocated before with the call to StarBurn_CdvdBurnerGrabber_Create (see page 31)().
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree to import track to.
IN UCHAR p__UCHAR__TrackNumber	Track number to import.
IN BOOLEAN p__BOOLEAN__ImportJolietStructures	Should we try to import Joliet structures (TRUE) or proceed with ISO9660 only (FALSE).
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text buffer.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message. If the exception number will be EN_SCSI_CDB_FAILED, CDB_FAILURE_INFORMATION (see page 480) will be filled with appropriate values (CDB that failed, CDB size, SCSI sense data, SCSI sense data size, SCSI transport, device and host adapter status codes).

Description

This function imports track for created ISO9660 or Joliet file tree.

Remarks

Please see the TrackAtOnceFromTreeWithImport sample that will demonstrate how ISO9660 or Joliet file system image can be created, existing track imported and whole result burn to the CD/DVD/Blu-Ray/HD-DVD media.

If you use your own file order, when auto-sorting option is turned off (see StarBurn_ISO9660JolietFileTree_SetAutoSorting

(see page 325)), boot image will be added to ISO image as a file in root directory. So if you want boot image file to have the least LBA, call StarBurn_ISO9660JolietFileTree_SetBootImage (see page 326) after adding all other files.

[WARNING! It's required to create empty file tree, import existing disc content into it and only after this add new or delete existing file tree nodes. Reversed direction should not be tried!]

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree (see page 40), StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree (see page 46), StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141), StarBurn_CdvdBurnerGrabber_ImportFile

Example

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__ISO9660JolietFileTree;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Create and add something to ISO9660 file tree here

// Try to import file system structures from the track
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_ImportTrack(
    l__PVOID__CdvdBurnerGrabber,
    l__PVOID__ISO9660JolietFileTree,
    0x01, // First track,
    TRUE, // Try to proceed as Joliet
    l__CHAR__ExceptionText,
```

```
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError
    );

// Check for correct reply
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.236 StarBurn_ISO9660JolietFileTree_Read Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_Read(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN LONG p__LONG__IoTransferSizeInUCHARs,
    OUT PUCHAR p__PUCHAR__DataBuffer
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288) and StarBurn_ISO9660FileTree_BuildImage was applied to.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).
IN LONG p__LONG__IoTransferSizeInUCHARs	Number of UCHARs to read from the image from the current offset.
OUT PUCHAR p__PUCHAR__DataBuffer	Pointer to data buffer readen UCHARs will be stored.

Returns

Execution status. EN_SUCCESS if the operation successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function reads ISO9660 or Joliet file system image from current offset into the user's data buffer. ISO9660 or Joliet file tree must have assigned all the logical block offsets and all file system internal structures allocated and initialized. The data that is read can be either stored in the file on the hard disk or directly written to CD/DVD/Blu-Ray/HD-DVD media.

Remarks

Please see the BuildImage sample that will demonstrate how file system image can be stored on the hard disk drive and either burn later to CD/DVD/Blu-Ray/HD-DVD media or used with CD/DVD/Blu-Ray/HD-DVD emulation software packages. Also TrackAtOnceFromTree sample can be checked out to see how ISO9660 or Joliet image can be recorded on the CD/DVD/Blu-Ray/HD-DVD media w/o intermediate buffering on the hard disk.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_BuildImage (see page 283), StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487)

Example

This example allocates Joliet file tree from the directory, builds file system image, reads some data from it and destroys the file tree after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
UCHAR l__UCHAR__DataBuffer[ 4096 ];
LARGE_INTEGER l__LARGE_INTEGER__NumberOfUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to build the image, start with LBA 0 and include up to 8 levels into the image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_BuildImage(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    0,
    8
    FALSE,
    "Volume",
    "Publisher",
    "Application"
    );

// Prepare data buffer
RtlZeroMemory(
    &l__UCHAR__DataBuffer,
    sizeof( l__UCHAR__DataBuffer )
    );

// Set transfer size
l__LARGE_INTEGER__NumberOfUCHARs.QuadPart = sizeof( l__UCHAR__DataBuffer );

// Try to read
l__EXCEPTION_NUMBER =
```

```
StarBurn_ISO9660JolietFileTree_Read(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    l__LARGE_INTEGER__NumberOfUCHARs.LowPart,
    ( PUCHAR )( &l__UCHAR__DataBuffer )
    );

// Check for success
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with data buffer here...

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.237 StarBurn_ISO9660JolietFileTree_Remove Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_ISO9660JolietFileTree_Remove(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN OUT PVOID * p__PPVOID__ISO9660JolietFileTreeNode	Pointer to pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().

Returns

TRUE if the node was destroyed, FALSE if node was not destroyed.

Description

This function deletes ISO9660 or Joliet file tree node and all it's kids.

Remarks

You cannot delete root node - FALSE will be returned in this case.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_Add (see page 272)

Example

This example allocates Joliet file tree, get root node, gets first kid, deletes first node and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
PVOID l__PVOID__FirstKidNode;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get first kid node here
l__PVOID__FirstKidNode =
StarBurn_ISO9660JolietFileTree_GetFirstKid(
    l__PVOID__FileTree,
    l__PVOID__RootNode
    );

// Check for correct response
if ( l__PVOID__FirstKidNode == NULL )
{
// Handle error here...
}

// Try to delete first kid
if (
StarBurn_ISO9660JolietFileTree_Remove(
    l__PVOID__ISO9660JolietFileTree,
    &l__PVOID__FirstKidNode
    ) == FALSE
```

```
)
{
// Handle error here...
}

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.238 StarBurn_ISO9660JolietFileTree_SeekToBegin Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_SeekToBegin(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660FileTree_Create or StarBurn_JolietFileTree_Create, StarBurn_ISO9660JolietFileTree_CreateFromRoot and StarBurn_ISO9660FileTree_BuildImage was applied to.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to array of CHARs that will be used to store formatted exception description message.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Size of the array of CHARs used to be formatted exception message storage.
OUT PULONG p__PULONG__SystemError	Pointer to ULONG that will contain the system error (if some will occur).

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function seeks the ISO9660 or Joliet file system image from current offset to the very beginning so read operation will start from the 0 offset of the file system image. ISO9660 or Joliet file tree must have assigned all the logical block offsets and all file system internal structures allocated and initialized.

Remarks

Please see the BuildImage sample that will demonstrate how file system image can be stored on the hard disk drive and either burn later to CD/DVD/Blu-Ray/HD-DVD media or used with CD/DVD/Blu-Ray/HD-DVD emulation software packages. StarBurn_ISO9660JolietFileTree_SeekToBegin is used to rewind the file system image after some part of it was read. Rewinding can be used when the user will want to write the same file system image to the media multiple times.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Read (see page 318), StarBurn_Destroy (see page 214), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487),

Example

This example allocates Joliet file tree from the directory, builds file system image, reads some stuff from it and destroys the file tree after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
UCHAR l__UCHAR__DataBuffer[ 4096 ];
LARGE_INTEGER l__LARGE_INTEGER__NumberOfUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Try to build the image, start with LBA 0 and include up to 8 levels into the image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_BuildImage(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    0,
    8
    FALSE,
    "Volume",
    "Publisher",
    "Application"
    );

// Prepare data buffer
RtlZeroMemory(
    &l__UCHAR__DataBuffer,
    sizeof( l__UCHAR__DataBuffer )
    );

// Set transfer size
l__LARGE_INTEGER__NumberOfUCHARs.QuadPart = sizeof( l__UCHAR__DataBuffer );

// Try to read
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Read(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
```

```
    l__LARGE_INTEGER__NumberOfUCHARs.LowPart,
    ( PUCHAR )( &l__UCHAR__DataBuffer )
    );

// Check for success
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with data buffer here...

// Try to seek to the very beging of the "virtual" file system image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_SeekToBegin(
    l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError
    );

// Check for success
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with image once again...

// Perform actions with Joliet tree here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.239 StarBurn_ISO9660JolietFileTree_SetAttributes Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_ISO9660JolietFileTree_SetAttributes(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    IN ULONG p__ULONG__Attributes
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
IN ULONG p__ULONG__Attributes	New attributes

Returns

None. This function cannot fail.

Description

This function sets ISO9660 or Joliet file tree node attributes.

Remarks

Attributes can be used to build correct tree to visualize enumeration of all the kids and walk down the tree.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)

Example

This example allocates Joliet file tree, get root node, gets attributes and destroys it (tree) after it's not needed any more.

2.1.240 StarBurn_ISO9660JolietFileTree_SetAutoSorting Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_ISO9660JolietFileTree_SetAutoSorting(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    IN BOOLEAN p__BOOLEAN__IsSorting
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
IN BOOLEAN p__BOOLEAN__IsSorting	New auto-sorting mode

Returns

None. This function cannot fail.

Description

This function sets ISO9660 or Joliet file tree node auto sorting mode.

Remarks

Auto sorting mode allows sorting sub-nodes before adding it to the image. It should be turned off if you want to specify your own order of adding files.

File order determines how the LBAs would be assigned. Lower LBA would be assigned to the last added file.

It may be used to ensure that boot image has low LBA. See StarBurn_ISO9660JolietFileTree_SetBootImage (see page 326) or StarBurn_ISO9660JolietFileTree_SetBootImageEx (see page 328).

See Also

StarBurn_ISO9660JolietFileTree_GetSorting, StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214)

2.1.241 StarBurn_ISO9660JolietFileTree_SetBootImage Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_SetBootImage(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PCHAR p__PCHAR__AbsoluteFileName,
    IN UCHAR p__UCHAR__SystemType,
    IN UCHAR p__UCHAR__MediaType,
    IN UCHAR p__UCHAR__SectorsToLoad,
    IN BOOLEAN p__BOOLEAN__HideNode,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree to add boot image to.
IN PCHAR p__PCHAR__AbsoluteFileName	Pointer to absolute file name of the boot image.
IN UCHAR p__UCHAR__SystemType	System type value.
IN UCHAR p__UCHAR__MediaType	Media type value.
IN UCHAR p__UCHAR__SectorsToLoad	Number of sectors to load
IN BOOLEAN p__BOOLEAN__HideNode	Indicates will the node be present in the root folder or not
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text buffer.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function sets boot image for created ISO9660 or Joliet file tree.

Remarks

Please see the TrackAtOnceFromTreeWithBoot sample that will demonstrate how ISO9660 or Joliet file system image can be created, boot image added and whole result burn to the CD/DVD/Blu-Ray/HD-DVD media.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree (see page 40), StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree (see page 46),

StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141)

Example

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__ISO9660JolietFileTree;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Create and add something to ISO9660 file tree here

// Try to set boot image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_SetBootImage(
    l__PVOID__ISO9660JolietFileTree,
    "C:\bootimage.bin", // Here is our boot image we gonna use
    6, // Just stick with 6 here...
    ELTORITO_MEDIA_FLOPPY144,
    1,// set only one sector to load
    FALSE, // don't hide the file
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError
    );

// Check for correct reply
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
```

```
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.242 StarBurn_ISO9660JolietFileTree_SetBootImageEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_ISO9660JolietFileTree_SetBootImageEx(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PCHAR p__PCHAR__AbsoluteFileName,
    IN PCHAR p__PCHAR__BootImageFileName,
    IN ULONG p__ULONG__BootImageFileAttributes,
    IN UCHAR p__UCHAR__SystemType,
    IN UCHAR p__UCHAR__MediaType,
    IN UCHAR p__UCHAR__SectorsToLoad,
    IN USHORT p__USHORT__LoadSegment,
    IN BOOLEAN p__BOOLEAN__IsHideNode,
    IN BOOLEAN p__BOOLEAN__IsBootInformationPatch,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to ISO9660 or Joliet file tree to add boot image to.
IN PCHAR p__PCHAR__AbsoluteFileName	Pointer to absolute file name of the boot image.
IN PCHAR p__PCHAR__BootImageFileName	Pointer to boot image file name with that be saved on disc.
IN ULONG p__ULONG__BootImageFileAttributes	Boot image file attributes.
IN UCHAR p__UCHAR__SystemType	System type value.
IN UCHAR p__UCHAR__MediaType	Media type value.
IN UCHAR p__UCHAR__SectorsToLoad	Number of sectors to load
IN USHORT p__USHORT__LoadSegment	Physical address of the memory where will be written code from the first sector of boot image.
IN BOOLEAN p__BOOLEAN__IsHideNode	Indicates will the node be present in the root folder or not
IN BOOLEAN p__BOOLEAN__IsBootInformationPatch	Is recalculate CRC of the boot image.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text buffer.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This extended function sets boot image for created ISO9660 or Joliet file tree.

Remarks

Please see the TrackAtOnceFromTreeWithBoot sample that will demonstrate how ISO9660 or Joliet file system image can be created, boot image added and whole result burn to the CD/DVD/Blu-Ray/HD-DVD media.

If you use your own file order, when auto-sorting option is turned off (see StarBurn_ISO9660JolietFileTree_SetAutoSorting (see page 325)), boot image will be added to ISO image as a file in root directory. So if you want boot image file to have

the least LBA, call StarBurn_ISO9660JolietFileTree_SetBootImage (see page 326) after adding all other files.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_Create (see page 31), PCALLBACK (see page 582), EXCEPTION_NUMBER (see page 487), CDB_FAILURE_INFORMATION (see page 480), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172), StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree (see page 40), StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree (see page 46), StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141)

Example

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
PVOID l__PVOID__ISO9660JolietFileTree;

// Prepare exception text buffer
RtlZeroMemory(
&l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText )
);

// Prepare CDB failure information
RtlZeroMemory(
&l__CDB_FAILURE_INFORMATION,
sizeof( l__CDB_FAILURE_INFORMATION )
);

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
&l__PVOID__CdvdBurnerGrabber,
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError,
&l__CDB_FAILURE_INFORMATION,
( PCALLBACK )( StarBurn_Callback ),
0,
0,
4,
0,
32
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Create and add something to ISO9660 file tree here

// Try to set boot image
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_SetBootImageEx(
l__PVOID__ISO9660JolietFileTree,
"C:\bootimage.bin", // Here is our boot image we gonna use
"boot.img",
0, //Attributes
6, // Just stick with 6 here...
ELTORITO_MEDIA_CUSTOM, //No emulation
4,// set four sector to load
ELTORITO_DEF_LOAD_SEGMENT,
TRUE, // hide the file
```

```
FALSE, //No recalculation CRC
l__CHAR__ExceptionText,
sizeof( l__CHAR__ExceptionText ),
&l__ULONG__SystemError
);

// Check for correct reply
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.243 StarBurn_ISO9660JolietFileTree_SetNames Function

C++

```
__stdcall STARBURN_IMPEX_API BOOLEAN StarBurn_ISO9660JolietFileTree_SetNames(
    IN PVOID p__PVOID__ISO9660JolietFileTree,
    IN PVOID p__PVOID__ISO9660JolietFileTreeNode,
    IN PCHAR p__PCHAR__AbsoluteName,
    IN PCHAR p__PCHAR__LongName,
    IN PCHAR p__PCHAR__ShortName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__ISO9660JolietFileTree	Pointer to the ISO9660 or Joliet file tree object that toolkit allocated during call to StarBurn_ISO9660JolietFileTree_Create (see page 288)().
IN PVOID p__PVOID__ISO9660JolietFileTreeNode	Pointer to the ISO9660 or Joliet file tree node object that is either result of previous tree node kids enumeration with StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)() and StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)() or the result of call to StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)().
IN PCHAR p__PCHAR__AbsoluteName	Pointer to the array of CHARs MAX_PATH in size that will be passed as absolute name.
IN PCHAR p__PCHAR__LongName	Pointer to the array of CHARs MAX_PATH in size that will be passed as long name.
IN PCHAR p__PCHAR__ShortName	Pointer to the array of CHARs MAX_PATH in size that will be passed as short name.

Returns

Nothing. Passed buffers will be copied to the internal file tree node. This function cannot fail. Attention! For now only long file name can be changed (short and absolute file names will be ignored).

Description

This function sets ISO9660 or Joliet file tree node names (long, short and absolute).

Remarks

Names can be used to build correct tree to visualize enumeration of all the kids and walk down the tree. Changing the names can allow to store the files or directories under different names.

See Also

StarBurn_ISO9660JolietFileTree_Create (see page 288), StarBurn_ISO9660JolietFileTree_Add (see page 272), StarBurn_ISO9660JolietFileTree_GetRoot (see page 311), StarBurn_Destroy (see page 214), StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303), StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294), StarBurn_ISO9660JolietFileTree_GetAttributes (see page 290), StarBurn_ISO9660JolietFileTree_GetNames (see page 299)

Example

This example allocates Joliet file tree, get root node, gets names, sets new ones and destroys it (tree) after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__FileTree;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__TreeNodes;
ULONG l__ULONG__SystemError;
CHAR l__CHAR__ExceptionText[ 1024 ];
PVOID l__PVOID__RootNode;
CHAR l__CHAR__AbsoluteName[ MAX_PATH ];
CHAR l__CHAR__LongName[ MAX_PATH ];
CHAR l__CHAR__ShortName[ MAX_PATH ];
INT l__INT__NameSizeInUCHARs;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Try to create Joliet file tree
l__EXCEPTION_NUMBER =
StarBurn_ISO9660JolietFileTree_Create(
    &l__PVOID__FileTree,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    ( PCALLBACK )( StarBurn_Callback ),
    ( PVOID )( &l__LONG__TreeNodes ),
    TRUE,
    FALSE,
    FILE_TREE_JOLIET
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add directories or files with the help of the StarBurn_ISO9660JolietFileTree_Add here...

// Get root node here
l__PVOID__RootNode =
StarBurn_ISO9660JolietFileTree_GetRoot( l__PVOID__FileTree );

// Check for correct response
if ( l__PVOID__RootNode == NULL )
{
// Handle error here...
}

// Get names here
StarBurn_ISO9660JolietFileTree_GetNames(
    l__PVOID__FileTree,
    l__PVOID__RootNode,
    ( PCHAR )( &l__CHAR__AbsoluteName ),
    ( PCHAR )( &l__CHAR__LongName ),
    ( PCHAR )( &l__CHAR__ShortName )
```

```
    );

// Change long name

l__INT__NameSizeInUCHARs = strlen( l__CHAR__LongName );

RtlZeroMemory(
&l__CHAR__LongName,
sizeof( l__CHAR__LongName )
);

strncpy(
l__CHAR__LongName,
"New name",
l__INT__NameSizeInUCHARs
);

// Set names here
StarBurn_ISO9660JolietFileTree_SetNames(
    l__PVOID__FileTree,
    l__PVOID__RootNode,
    ( PCHAR )( &l__CHAR__AbsoluteName ),
    ( PCHAR )( &l__CHAR__LongName ),
    ( PCHAR )( &l__CHAR__ShortName )
    );

// Do something with Joliet file tree root node here...

// Destroy the Joliet file tree
StarBurn_Destroy( &l__PVOID__FileTree );

// Just check for pointer (paranoid?)
if ( l__PVOID__FileTree != NULL )
{
// Handle error here...
}
```

2.1.244 StarBurn_SetBufferUnderrunTimeOutInMs Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetBufferUnderrunTimeOutInMs(
    IN LONG p__LONG__NewBufferUnderrunTimeOutInMs
);
```

File

StarBurn.h (see page 662)

Returns

Nothing.

Description

This function sets buffer underrun timeout in milliseconds. It's amount of time StarBurn core would wait before resubmitting each command to the drive if burning was faster then reading and StarBurn had exhausted software cache it uses for buffering. Modern hardware has BUP (Buffer Underrun Protection) and would perfectly survive in such a condition and keep burning disc still usable.

Remarks

Having large timeout would allow StarBurn to grab more information into software cache, at the same time having timeout small and disc burning speed MUCH faster then reading speed from the source media (say slow hard disk or network) would produce a lot of BUP errors, start-stop cycles and resulting recorded disc quality would be very low. In general it's a good idea not to touch buffer underrun timeout value at all and keep everything AS IS. It's value for "hardcore tuning".

See Also

StarBurn_GetBufferUnderrunTimeOutInMs (see page 236)

Example

There are no samples for StarBurn_SetBufferUnderrunTimeOutInMs(...) API call.

2.1.245 StarBurn_SetDeviceTimeOutByDeviceAddress Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_SetDeviceTimeOutByDeviceAddress(
    IN UCHAR p__UCHAR__DevicePort,
    IN UCHAR p__UCHAR__DeviceBus,
    IN UCHAR p__UCHAR__DeviceID,
    IN UCHAR p__UCHAR__DeviceLUN,
    IN ULONG p__ULONG__DeviceTimeOutInSeconds
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN UCHAR p__UCHAR__DevicePort	Device port.
IN UCHAR p__UCHAR__DeviceBus	Device SCSI bus.
IN UCHAR p__UCHAR__DeviceID	Device SCSI ID.
IN UCHAR p__UCHAR__DeviceLUN	Device LUN.
IN ULONG p__ULONG__DeviceTimeOutInSeconds	New device timeout in seconds.

Returns

Execution status.

Description

This function sets device timeout in seconds by device SCSI address.

Remarks

This one is for ASPI devices only.

See Also

StarBurn_GetDeviceTimeOutByDeviceAddress (see page 239)

Example

Please see FindDevice sample as example how to use StarBurn_SetDeviceTimeOutByDeviceAddress API call.

2.1.246 StarBurn_SetDVDPadding Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetDVDPadding(
    IN BOOLEAN p__BOOLEAN__NewDVDPadding
);
```

File

StarBurn.h (see page 662)

Returns

Nothing.

Description

This function sets DVD padding mode (when at least 1GB of the data would be recorded to DVD media - required for DVD-Video compilations to work properly on standalone DVD players) to enabled (TRUE) or disabled (FALSE).

Remarks

"Padding mode" should be always enabled for DVD-Video compilations (or resulting disc would not be playable on standalone DVD players) and should be always disabled for generic data discs (or quite a lot of disc capacity would be simply wasted).

See Also

StarBurn_GetDVDPadding (see page 239)

Example

There are no examples for StarBurn_SetDVDPadding(...) API call.

2.1.247 StarBurn_SetDVDPLUSRDLCompatibleMode Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetDVDPLUSRDLCompatibleMode(
    IN BOOLEAN p__BOOLEAN__NewDVDPLUSRDLCompatibleMode
);
```

File

StarBurn.h (see page 662)

Returns

Nothing.

Description

This function sets DVD+R DL (Dual Layer) so-called "compatible mode" (when layer would be switched exactly at 1/2 of the data payload recorded on the disc - required for DVD-Video compilations to work properly) to enabled (TRUE) or disabled (FALSE).

Remarks

"Compatible mode" should be always enabled for DVD-Video compilations (or resulting disc would not be playable on standalone DVD players) and should be always disabled for generic data discs (or quite a lot of disc capacity would be simply wasted).

See Also

StarBurn_GetDVDPLUSRDLCompatibleMode (see page 240)

Example

There are no examples for StarBurn_SetDVDPLUSRDLCompatibleMode(...) API call.

2.1.248 StarBurn_SetEjectAfterFail Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetEjectAfterFail(
    IN BOOLEAN p__BOOLEAN__NewEjectAfterFail
);
```

File

StarBurn.h (see page 662)

Returns

Nothing.

Description

This function sets so-called "eject after fail" (if during burning process error would happen - should StarBurn eject disc or not) to enabled (TRUE) or disabled (FALSE).

Remarks

It's a good idea to keep everything AS IS, however if StarBurn is used with automatic loaders or whatever sometimes it's required to keep manual control over eject process.

See Also

StarBurn_GetEjectAfterFail (see page 241)

Example

There are no examples for StarBurn_SetEjectAfterFail(...) API call.

2.1.249 StarBurn_SetFastReadTOC Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetFastReadTOC(
    IN BOOLEAN p__BOOLEAN__NewFastReadTOC
);
```

File

StarBurn.h (see page 662)

Returns

Nothing.

Description

This function sets so-called "fast read TOC" (TOC information is not 100% accurate - to get EXACT track length StarBurn try to read beginning and end of the track to find sub-channel index switch indicating EXACT track beginning or EXACT track end) mode to enabled (TRUE) or disabled (FALSE).

Remarks

It's a good idea to keep everything AS IS, however if StarBurn takes a lot of time to analyze the disc (it can happen when drive hardware error correction cannot be disabled and reading pre-gap and post-gap of the track is DOG SLOW) "fast read TOC" could be switched ON.

See Also

StarBurn_GetFastReadTOC (see page 241)

Example

Please see GrabTrack sample no how to use StarBurn_SetFastReadTOC(...) API call.

2.1.250 StarBurn_SetIs64KBIO Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetIs64KBIO(
    IN BOOLEAN p__BOOLEAN__NewIs64KBIO
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN BOOLEAN p__BOOLEAN__NewIs64KBIO	Current "64KB I/O" enabled (TRUE) or disabled (FALSE).

Returns

Nothing.

Description

This function sets current so-called "64KB I/O" (when StarBurn issues 64KB I/O packets instead of the 32KB ones) to enabled (TRUE) or disabled (FALSE) state.

Remarks

There's no need to play with this parameter unless you really have broken device you will to force 32KB I/O requests for.

See Also

StarBurn_GetIs64KBIO (see page 242)

Example

There are no examples for StarBurn_SetIs64KBIO(...) API call.

2.1.251 StarBurn_SetIsCollisionDetectionDisabled Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetIsCollisionDetectionDisabled(
    IN BOOLEAN p__BOOLEAN__NewIsCollisionDetectionDisabled
);
```

File

StarBurn.h (see page 662)

Returns

Nothing.

Description

This function sets collision detection to disabled (TRUE) or enabled (FALSE). If collision detection is enabled - every new file added to ISO9660 or Joliet file tree would be checked to have unique name. If such a file already exist - special callback (collision detection one) would be called so user would be able to either replace or rename new or old file. However if collision detection is disabled - no comparison and no actions would be performed. File would be just added to the

destination file system image AS IS, having it's original name.

Remarks

If new image is created from the content stored on the hard disk - there's no way for file names to be the same. So file system image creation process could be speed up quite a lot by turning collision detection OFF (especially if there are a lot of files inside each directory). In other cases it's not recommended to play with this option.

See Also

StarBurn_GetIsCollisionDetectionDisabled (see page 243)

Example

Please see BuildImage sample to find out how collision detection could be disabled and enabled during file system image creation.

2.1.252 StarBurn_SetIsSafeGrabbingEnabled Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetIsSafeGrabbingEnabled(
    IN BOOLEAN p__BOOLEAN__NewIsSafeGrabbingEnabled
);
```

File

StarBurn.h (see page 662)

Returns

Nothing.

Description

This function sets so-called "safe grabbing" (when StarBurn would mix read commands with checking for device to become ready - required to workaround broken ATAPI-to-USB bridges) to enabled (TRUE) or disabled (FALSE).

Remarks

There's no need to play with this parameter unless you really have ATAPI device in external USB enclosure and it hangs under heavy I/O load.

See Also

StarBurn_GetIsSafeGrabbingEnabled (see page 243)

Example

There are no examples for StarBurn_SetIsSafeGrabbingEnabled(...) API call.

2.1.253 StarBurn_SetUsingLocalDateTime Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetUsingLocalDateTime();
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_SetUsingLocalDateTime.

2.1.254 StarBurn_SetUsingUTCDateTime Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_SetUsingUTCDateTime();
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_SetUsingUTCDateTime.

2.1.255 StarBurn_sprintf_s Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_sprintf_s(
    char * buffer,
    size_t sizeOfBuffer,
    const char * format,
    ...
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
char * buffer	Storage location for output.
size_t sizeOfBuffer	Maximum number of characters to store.
const char * format	Format-control string.
argument	Optional arguments.

Description

This function wrap runtime sprintf_s function. In Visual Studio 2003 function sprintf_s not present.

2.1.256 StarBurn_SPTD_GetVersion Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_SPTD_GetVersion(
    OUT PUCHAR p__PUCHAR__SptdMajorVersion,
    OUT PUCHAR p__PUCHAR__SptdMinorVersion,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PUCHAR p__PUCHAR__SptdMajorVersion	Pointer to the major SPTD driver version.
OUT PUCHAR p__PUCHAR__SptdMinorVersion	Pointer to the minor SPTD driver version.

OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status. EN_SUCCESS if everythins is OK.

Description

This function checks if the SPTD (SCSI Pass Through Direct) driver is present and gets SPTD driver version.

See Also

StarBurn_Destroy (see page 214), StarBurn_CdvdBurnerGrabber_CreateExEx (see page 35), EXCEPTION_NUMBER (see page 487)

Example

This example just checks for the SPTD presence and queries SPTD driver version.

```
// Somewhere in the data region
EXCEPTION_NUMBER l__EXCEPTION_NUMBER = EN_SUCCESS;
ULONG l__ULONG__SystemError = ERROR_SUCCESS;
UCHAR l__UCHAR__Major = 0;
UCHAR l__UCHAR__Minor = 0;

// Try to get SPTD driver version
l__EXCEPTION_NUMBER =
StarBurn_SPTD_GetVersion(
&l__UCHAR__Major,
&l__UCHAR__Minor,
&l__ULONG__SystemError
);

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with the SPTD here...
```

2.1.257 StarBurn_StarPort_DeviceAddLocal Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceAddLocal(
    IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE,
    IN PCHAR p__PCHAR__StarPortDeviceName,
    OUT PLONG p__PLONG__StarPortDeviceTargetId,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE	StarPort device type (See STARPORT_DEVICE_TYPE (see page 569) enum for more information).
IN PCHAR p__PCHAR__StarPortDeviceName	Pointer to device name we want to add.
OUT PLONG p__PLONG__StarPortDeviceTargetId	Pointer to the variable to receive StarPort device TargetId after device object creation.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function adds local device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.

Remarks

This one is for Enterprise or Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocalEx (see page 340), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceAddRemoteEx (see page 341)

Example

Please see StarPort sample as example how to use StarBurn_StarPort_DeviceAddLocal API call.

2.1.258 StarBurn_StarPort_DeviceAddLocalEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceAddLocalEx(
    IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE,
    IN PCHAR p__PCHAR__StarPortDeviceName,
    OUT PLONG p__PLONG__StarPortDeviceTargetId,
    IN BOOLEAN p__BOOLEAN__Persistent,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE	StarPort device type (See STARPORT_DEVICE_TYPE (see page 569) enum for more information).
IN PCHAR p__PCHAR__StarPortDeviceName	Pointer to device name we want to add.
OUT PLONG p__PLONG__StarPortDeviceTargetId	Pointer to the variable to receive StarPort device TargetId after device object creation.
IN BOOLEAN p__BOOLEAN__Persistent	Is device will be added persistent.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function adds local device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Device can be added persistent.

Remarks

This one is for Enterprise or Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocal (see page 339),StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceAddRemoteEx (see page 341)

Example

Please see StarPort sample as example how to use StarBurn_StarPort_DeviceAddLocal (see page 339) API call.

2.1.259 StarBurn_StarPort_DeviceAddRemote Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceAddRemote(
    IN PCHAR p__PCHAR__TargetAddress,
    IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE,
    IN PCHAR p__PCHAR__StarPortDeviceName,
    OUT PLONG p__PLONG__StarPortDeviceTargetId,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__TargetAddress	Pointer to the IP:port string corresponding to iSCSI target we'd like to add.
IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE	StarPort device type (See STARPORT_DEVICE_TYPE (see page 569) enum for more information).
IN PCHAR p__PCHAR__StarPortDeviceName	Pointer to iSCSI target device name we want to add.
OUT PLONG p__PLONG__StarPortDeviceTargetId	Pointer to the variable to receive StarPort device TargetId after device object creation.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function adds remote device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Device can be mount persistent.

Remarks

This one is for Network license holders only.

See Also

StarBurn_StarPort_DeviceAddRemoteEx (see page 341), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceAddLocalEx (see page 340)

Example

Please see StarPort sample as example how to use StarBurn_StarPort_DeviceAddRemote API call.

2.1.260 StarBurn_StarPort_DeviceAddRemoteEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceAddRemoteEx(
    IN PCHAR p__PCHAR__TargetAddress,
    IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE,
    IN PCHAR p__PCHAR__StarPortDeviceName,
```

```
    OUT PLONG p__PLONG__StarPortDeviceTargetId,
    IN BOOLEAN p__BOOLEAN__Persistent,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__TargetAddress	Pointer to the IP:port string corresponding to iSCSI target we'd like to add.
IN STARPORT_DEVICE_TYPE p__STARPORT_DEVICE_TYPE	StarPort device type (See STARPORT_DEVICE_TYPE (see page 569) enum for more information).
IN PCHAR p__PCHAR__StarPortDeviceName	Pointer to iSCSI target device name we want to add.
OUT PLONG p__PLONG__StarPortDeviceTargetId	Pointer to the variable to receive StarPort device TargetId after device object creation.
IN BOOLEAN p__BOOLEAN__Persistent	Is device will be mount persistent.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function adds remote device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.

Remarks

This one is for Network license holders only.

See Also

StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceAddLocalEx (see page 340)

Example

Please see StarPort sample as example how to use StarBurn_StarPort_DeviceAddRemote (see page 341) API call.

2.1.261 StarBurn_StarPort_DeviceListQueryLocal Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceListQueryLocal(
    OUT PSTARPORT_DEVICE_LIST p__PSTARPORT_DEVICE_LIST,
    IN ULONG p__ULONG__StarPortDeviceListSizeInUCHARs,
    OUT PLONG p__PLONG__NumberOfTargets,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PSTARPORT_DEVICE_LIST p__PSTARPORT_DEVICE_LIST	Pointer to the STARPORT_DEVICE_LIST (see page 568) structure drive would fill with StarPort device list information.
IN ULONG p__ULONG__StarPortDeviceListSizeInUCHARs	True STARPORT_DEVICE_LIST (see page 568) structure size in UCHARs.
OUT PLONG p__PLONG__NumberOfTargets	Pointer to the variable to receive number of targets StarPort would enumerate during this call.

OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function gets device list from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Unlike StarBurn_StarPort_DeviceListQueryRemote (see page 343) API call which retrieves information about REMOTE device list this call gets information about LOCAL device list.

Remarks

This one is for Enterprise or Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_DeviceAddRemote (see page 341)

Example

Please see StarPort sample as example how to use StarBurn_StarPort_DeviceListQueryLocal API call.

2.1.262 StarBurn_StarPort_DeviceListQueryRemote Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceListQueryRemote(
    IN PCHAR p__PCHAR__TargetAddress,
    OUT PSTARPORT_DEVICE_LIST p__PSTARPORT_DEVICE_LIST,
    IN ULONG p__ULONG__StarPortDeviceListSizeInUCHARs,
    OUT PLONG p__PLONG__NumberOfTargets,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__TargetAddress	Pointer to IP:port string which corresponds to the iSCSI target we'd like to query.
OUT PSTARPORT_DEVICE_LIST p__PSTARPORT_DEVICE_LIST	Pointer to the STARPORT_DEVICE_LIST (see page 568) structure drive would fill with StarPort device list information.
IN ULONG p__ULONG__StarPortDeviceListSizeInUCHARs	True STARPORT_DEVICE_LIST (see page 568) structure size in UCHARs.
OUT PLONG p__PLONG__NumberOfTargets	Pointer to the variable to receive number of targets StarPort would enumerate during this call.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function gets device list from remote StarWind iSCSI target. StarWind exports RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) targets. Unlike StarBurn_StarPort_DeviceListQueryLocal (see page 342) API call which retreives information about LOCAL device list this call gets information about REMOTE device list.

Remarks

This one is for Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceAddRemote (see page 341)

Example

Please see StarPort sample as example how to use StarBurn_StarPort_DeviceListQueryRemote API call.

2.1.263 StarBurn_StarPort_DeviceRemove Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceRemove(
    IN LONG p__LONG__StarPortDeviceTargetId,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN LONG p__LONG__StarPortDeviceTargetId	StarPort device TargetId we want to remove from the system.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function removes device from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.

Remarks

This one is for Enterprise or Network license holders only.

See Also

StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemoveEx (see page 344)

Example

Please see StarPort sample as example how to use StarBurn_StarPort_DeviceRemove API call.

2.1.264 StarBurn_StarPort_DeviceRemoveEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_DeviceRemoveEx(
    IN LONG p__LONG__StarPortDeviceTargetId,
    IN BOOLEAN p__BOOLEAN_ForceRemove,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN LONG p__LONG__StarPortDeviceTargetId	StarPort device TargetId we want to remove from the system.
IN BOOLEAN p__BOOLEAN_ForceRemove	FALSE - Eject (safe mode, can fail if the device is in use)
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.
TRUE	Remove (always succeed, but may lead to data corruption if the device is in use)

Returns

Execution status.

Description

This function removes device from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.

Remarks

This one is for Enterprise or Network license holders only.

See Also

StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemove (see page 344)

2.1.265 StarBurn_StarPort_GetDeviceInformation Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_GetDeviceInformation(
    IN LONG p__LONG__StarPortDeviceTargetId,
    OUT PSTARPORT_DEVICE_TYPE p__PSTARPORT_DEVICE_TYPE,
    OUT PCHAR p__PCHAR__VendorID,
    IN ULONG p__ULONG__VendorIDSizeInUCHARs,
    OUT PCHAR p__PCHAR__ProductID,
    IN ULONG p__ULONG__ProductIDSizeInUCHARs,
    OUT PCHAR p__PCHAR__Revision,
    IN ULONG p__ULONG__RevisionSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN LONG p__LONG__StarPortDeviceTargetId	StarPort device TargetId we want to get device letter for.
OUT PSTARPORT_DEVICE_TYPE p__PSTARPORT_DEVICE_TYPE	Pointer to the STARPORT_DEVICE_TYPE (see page 569) variable to store the device type.
OUT PCHAR p__PCHAR__VendorID	Pointer to the buffer to receive vendor ID.
IN ULONG p__ULONG__VendorIDSizeInUCHARs	The size of the buffer to receive vendor ID.
OUT PCHAR p__PCHAR__ProductID	Pointer to the buffer to receive product ID.
IN ULONG p__ULONG__ProductIDSizeInUCHARs	The size of the buffer to receive product ID.
OUT PCHAR p__PCHAR__Revision	Pointer to the buffer to receive revision.
IN ULONG p__ULONG__RevisionSizeInUCHARs	The size of the buffer to receive revision.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function retrieves device type, vendor ID, product ID and revision for StarPort device by its target ID.

Remarks

This one is for Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_GetDeviceLetter (see page 346), StarBurn_StarPort_GetVersion (see page 347).

Example

Please see StarPort GUI sample as example how to use StarBurn_StarPort_GetDeviceInformation API call.

2.1.266 StarBurn_StarPort_GetDeviceLetter Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_GetDeviceLetter(
    IN LONG p__LONG__StarPortDeviceTargetId,
    OUT PCHAR p__PCHAR__DeviceLetter,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN LONG p__LONG__StarPortDeviceTargetId	StarPort device TargetId we want to get device letter for.
OUT PCHAR p__PCHAR__DeviceLetter	Pointer to the CHAR variable to store the device letter.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function retrieves device letter (symbolic link) for StarPort device by its target ID.

Remarks

This one is for Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceListQueryRemote (see page 343)

Example

Please see StarPort GUI sample as example how to use StarBurn_StarPort_GetDeviceLetter API call.

2.1.267 StarBurn_StarPort_GetDeviceSCSIAddress Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_GetDeviceSCSIAddress(
    IN LONG p__LONG__StarPortDeviceTargetId,
    OUT PSCSI_DEVICE_ADDRESS p__PSCSI_DEVICE_ADDRESS,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN LONG p__LONG__StarPortDeviceTargetId	StarPort device TargetId we want to get device letter for.
OUT PSCSI_DEVICE_ADDRESS p__PSCSI_DEVICE_ADDRESS	Pointer to the variable to receive device SCSI address.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function retrieves device SCSI address for StarPort device by its target ID.

Remarks

This one is for Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_GetDeviceLetter (see page 346), StarBurn_StarPort_GetVersion (see page 347), StarBurn_StarPort_GetDeviceLetter (see page 346)

Example

Please see StarPort GUI sample as example how to use StarBurn_StarPort_GetDeviceSCSIAddress API call.

2.1.268 StarBurn_StarPort_GetVersion Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarPort_GetVersion(
    OUT PULONG p__PULONG__BuildVersion,
    OUT PULONG p__PULONG__ApiVersion,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PULONG p__PULONG__BuildVersion	StarPortLite driver build version.

OUT PULONG p__PULONG__ApiVersion	StarPortLite driver API version.
OUT PULONG p__PULONG__SystemError	Pointer to the variable to receive system error if function would return anything except EN_SUCCESS.

Returns

Execution status.

Description

This function checks if the StarPortLite driver is installed and that its API version is equal to the one StarBurn was built with.

Remarks

This one is for Network license holders only.

See Also

StarBurn_StarPort_DeviceAddLocal (see page 339), StarBurn_StarPort_DeviceRemove (see page 344), StarBurn_StarPort_DeviceListQueryLocal (see page 342), StarBurn_StarPort_DeviceAddRemote (see page 341), StarBurn_StarPort_DeviceListQueryRemote (see page 343), StarBurn_StarPort_GetDeviceLetter (see page 346)

Example

Please see StarPort GUI sample as example how to use StarBurn_StarPort_GetVersion API call.

2.1.269
StarBurn_StarWave_CompressedFileReaderObjectBeginSeek Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileReaderObjectBeginSeek(
    IN PVOID p__PVOID__CompressedFileReaderObject
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CompressedFileReaderObject	Pointer to compressed file reader object.

Returns

Execution status.

Description

This function seeks all of the internal object pointers to the very beginning of it so following read operations would start from the very beginning instead of the currently set position.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileSupportedIs (see page 355), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351),

StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileReaderObjectBeginSeek API call.

2.1.270
StarBurn_StarWave_CompressedFileReaderObjectCreate Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileReaderObjectCreate(
    OUT PVOID * p__PPVOID__CompressedFileReaderObject,
    IN PCHAR p__PCHAR__CompressedFileName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID * p__PPVOID__CompressedFileReaderObject	Pointer to pointer to compressed file reader object after it's creation.
IN PCHAR p__PCHAR__CompressedFileName	Pointer to compressed file name to create compressed file reader object from.

Returns

Execution status.

Description

This function creates compressed file reader object from passed compressed audio file name.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileSupportedIs (see page 355) StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileReaderObjectCreate API call.

2.1.271
StarBurn_StarWave_CompressedFileReaderObjectCreateUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG
StarBurn_StarWave_CompressedFileReaderObjectCreateUnicode(
    OUT PVOID * p__PPVOID__CompressedFileReaderObject,
    IN PWCHAR p__PWCHAR__CompressedFileName
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_StarWave_CompressedFileReaderObjectCreateUnicode.

2.1.272
StarBurn_StarWave_CompressedFileReaderObjectDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileReaderObjectDestroy(
    OUT PVOID * p__PPVOID__CompressedFileReaderObject
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`OUT PVOID * p__PPVOID__CompressedFileReaderObject`	Pointer to pointer to compressed file reader object to destroy.

Returns

Execution status.

Description

This function destroys compressed file reader object.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileSupportedIs (see page 355),

StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileReaderObjectDestroy API call.

2.1.273
StarBurn_StarWave_CompressedFileReaderObjectRead Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileReaderObjectRead(
    IN PVOID p__PVOID__CompressedFileReaderObject,
    OUT PVOID p__PVOID__ReadDataBuffer,
    IN ULONG p__ULONG__ReadSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CompressedFileReaderObject	Pointer to compressed file reader object.
OUT PVOID p__PVOID__ReadDataBuffer	Pointer to output buffer to handle just read uncompressed data.
IN ULONG p__ULONG__ReadSizeInUCHARs	Data chunk (uncompressed) to read size in UCHARs.

Returns

Execution status.

Description

This function reads uncompressed payload data chunk from underlying compressed file reader object.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileSupportedIs (see page 355) StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354)

Example

See AudioCompressor StarBurn sample application as an example how to use

StarBurn_StarWave_CompressedFileReaderObjectRead API call.

2.1.274 StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG
StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet(
    IN PVOID p__PVOID__CompressedFileReaderObject,
    OUT PULONG p__PULONG__UncompressedSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CompressedFileReaderObject	Pointer to compressed file reader object.
OUT PULONG p__PULONG__UncompressedSizeInUCHARs	Pointer to the variable to receive uncompressed payload size in UCHARs.

Returns

Execution status.

Description

This function returns uncompressed payload size in UCHARs for underlying compressed file reader object.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileSupportedIs (see page 355) StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet_Fast (see page 353)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet API call.

2.1.275 StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet_Fast Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG
StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet_Fast(
    IN PVOID p__PVOID__CompressedFileReaderObject,
    OUT PULONG p__PULONG__UncompressedSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__CompressedFileReaderObject	Pointer to compressed file reader object.
OUT PULONG p__PULONG__UncompressedSizeInUCHARs	Pointer to the variable to receive uncompressed payload size in UCHARs.

Returns

Execution status.

Description

This function returns uncompressed payload size in UCHARs for underlying compressed file reader object.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileSupportedIs (see page 355) StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354)

2.1.276 StarBurn_StarWave_CompressedFileReaderObjectUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG
StarBurn_StarWave_CompressedFileReaderObjectUnicodeCreate(
    OUT PVOID * p__PPVOID__CompressedFileReaderObject,
    IN PWCHAR p__PWCHAR__CompressedFileName
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_StarWave_CompressedFileReaderObjectUnicodeCreate.

2.1.277 StarBurn_StarWave_CompressedFileRecompress Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileRecompress(
    IN PCHAR p__PCHAR__SourceCompressedFileName,
    IN PCHAR p__PCHAR__DestinationRecompressedFileName,
    IN ULONG p__ULONG__WorkingBufferSizeInUCHARs,
    OUT PULONG p__PULONG__UncompressedSizeInUCHARs,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__Context,
    IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__SourceCompressedFileName	Pointer to source compressed file name.
IN PCHAR p__PCHAR__DestinationRecompressedFileName	Pointer to destination compressed file name.
IN ULONG p__ULONG__WorkingBufferSizeInUCHARs	Working buffer size in UCHARs (recommended to have it equal to STARBURN_STARWAVE_IO_BUFFER_SIZE_IN_UCHARS (see page 648)).
OUT PULONG p__PULONG__UncompressedSizeInUCHARs	Pointer to the variable to receive uncompressed source stream size in UCHARs.
IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK	Pointer to callback function to process progress indication and I/O cancellation.
IN PVOID p__PVOID__Context	Pointer to the context value passed to callback function.
IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION	Compression template.

Returns

Execution status.

Description

This function recompresse already compressed file to new one with another compression.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileSupportedIs (see page 355)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileRecompress API call.

2.1.278 StarBurn_StarWave_CompressedFileRecompressUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileRecompressUnicode(
    IN PWCHAR p__PWCHAR__SourceCompressedFileName,
    IN PWCHAR p__PWCHAR__DestinationRecompressedFileName,
    IN ULONG p__ULONG__WorkingBufferSizeInUCHARs,
    OUT PULONG p__PULONG__UncompressedSizeInUCHARs,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__Context,
    IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_StarWave_CompressedFileRecompressUnicode.

2.1.279 StarBurn_StarWave_CompressedFileSupportedIs Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileSupportedIs(
    IN PCHAR p__PCHAR__CompressedFileName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PCHAR p__PCHAR__CompressedFileName`	Pointer to compressed file name.

Returns

Execution status.

Description

This function checks is file name actually points to supported compressed audio file.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileSupportedIs API call.

2.1.280 StarBurn_StarWave_CompressedFileUncompress Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileUncompress(
    IN PCHAR p__PCHAR__SourceCompressedFileName,
    IN PCHAR p__PCHAR__DestinationUncompressedFileName,
    IN ULONG p__ULONG__WorkingBufferSizeInUCHARs,
    IN BOOLEAN p__BOOLEAN__IsWavHeaderRequired,
    OUT PULONG p__PULONG__UncompressedSizeInUCHARs,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__Context
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PCHAR p__PCHAR__SourceCompressedFileName`	Pointer to source compressed file name.
`IN PCHAR p__PCHAR__DestinationUncompressedFileName`	Pointer to destination uncompressed file name.

IN ULONG p__ULONG__WorkingBufferSizeInUCHARs	Working buffer size in UCHARs (recommended to have it equal to STARBURN_STARWAVE_IO_BUFFER_SIZE_IN_UCHARS (see page 648)).
IN BOOLEAN p__BOOLEAN__IsWavHeaderRequired	Is WAV header required for destination file (WAV would be result) or not (RAW would be result).
OUT PULONG p__PULONG__UncompressedSizeInUCHARs	Pointer to the variable to receive uncompressed payload size in UCHARs.
IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK	Pointer to callback function to process progress indication and I/O cancellation.
IN PVOID p__PVOID__Context	Pointer to the context value passed to callback function.

Returns

Execution status.

Description

This function uncompresses compressed file to either WAV or RAW PCM.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileSupportedIs (see page 355)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileUncompress API call.

2.1.281
StarBurn_StarWave_CompressedFileUncompressUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileUncompressUnicode(
    IN PWCHAR p__PWCHAR__SourceCompressedFileName,
    IN PWCHAR p__PWCHAR__DestinationUncompressedFileName,
    IN ULONG p__ULONG__WorkingBufferSizeInUCHARs,
    IN BOOLEAN p__BOOLEAN__IsWavHeaderRequired,
    OUT PULONG p__PULONG__UncompressedSizeInUCHARs,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__Context
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_StarWave_CompressedFileUncompressUnicode.

2.1.282
StarBurn_StarWave_CompressedFileWriterObjectCreate Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileWriterObjectCreate(
    OUT PVOID * p__PPVOID__CompressedFileWriterObject,
    IN PCHAR p__PCHAR__CompressedFileName,
    IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID * p__PPVOID__CompressedFileWriterObject	Pointer to pointer to compressed file writer object.
IN PCHAR p__PCHAR__CompressedFileName	Pointer to compressed file name.
IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION	Compression template.

Returns

Execution status.

Description

This function creates compressed file writer object.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileSupportedIs (see page 355), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354),

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileWriterObjectCreate API call.

2.1.283
StarBurn_StarWave_CompressedFileWriterObjectCreateUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG
StarBurn_StarWave_CompressedFileWriterObjectCreateUnicode(
    OUT PVOID * p__PPVOID__CompressedFileWriterObject,
    IN PWCHAR p__PWCHAR__CompressedFileName,
    IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_StarWave_CompressedFileWriterObjectCreateUnicode.

2.1.284
StarBurn_StarWave_CompressedFileWriterObjectDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileWriterObjectDestroy(
    OUT PVOID * p__PPVOID__CompressedFileWriterObject
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`OUT PVOID * p__PPVOID__CompressedFileWriterObject`	Pointer to pointer to compressed file writer object.

Returns

Execution status.

Description

This function destroys compressed file writer object.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349),
StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352),
StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348),
StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351),

StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileSupportedIs (see page 355)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_CompressedFileWriterObjectDestroy API call.

2.1.285
StarBurn_StarWave_CompressedFileWriterObjectWrite Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_CompressedFileWriterObjectWrite(
    OUT PVOID p__PVOID__CompressedFileWriterObject,
    IN PVOID p__PVOID__WriterDataBuffer,
    IN ULONG p__ULONG__WriterSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
OUT PVOID p__PVOID__CompressedFileWriterObject	Pointer to compressed file writer object.
IN PVOID p__PVOID__WriterDataBuffer	Pointer to data buffer to write.
p__ULONG__WriteSizeInUCHARs	Data buffer to write size in UCHARs.

Returns

Execution status.

Description

This function writes uncompressed data chunk to compressed file writer object.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileSupportedIs (see page 355), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354)

Example

See AudioCompressor StarBurn sample application as an example how to use

StarBurn_StarWave_CompressedFileWriterObjectWrite API call.

2.1.286 StarBurn_StarWave_UncompressedFileCompress Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_UncompressedFileCompress(
    IN PCHAR p__PCHAR__SourceUncompressedFileName,
    IN PCHAR p__PCHAR__DestinationCompressedFileName,
    IN ULONG p__ULONG__WorkingBufferSizeInUCHARs,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__Context,
    IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR__SourceUncompressedFileName	Pointer to source uncompressed file name.
IN PCHAR p__PCHAR__DestinationCompressedFileName	Pointer to destination compressed file name.
IN ULONG p__ULONG__WorkingBufferSizeInUCHARs	Working buffer size in UCHARs (recommended to have it equal to STARBURN_STARWAVE_IO_BUFFER_SIZE_IN_UCHARS (see page 648)).
IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK	Pointer to callback function to process progress indication and I/O cancellation.
IN PVOID p__PVOID__Context	Pointer to the context value passed to callback function.
IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION	Compression template.

Returns

Execution status.

Description

This function compresses uncompressed file to new one with asked compression.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_CompressedFileSupportedIs (see page 355) StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_UncompressedFileCompress API call.

2.1.287
StarBurn_StarWave_UncompressedFileCompressUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_UncompressedFileCompressUnicode(
    IN PWCHAR p__PWCHAR__SourceUncompressedFileName,
    IN PWCHAR p__PWCHAR__DestinationCompressedFileName,
    IN ULONG p__ULONG__WorkingBufferSizeInUCHARs,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__Context,
    IN STARBURN_STARWAVE_COMPRESSION p__STARBURN_STARWAVE_COMPRESSION
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_StarWave_UncompressedFileCompressUnicode.

2.1.288
StarBurn_StarWave_UncompressedFileSupportedIs Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_UncompressedFileSupportedIs(
    IN PCHAR p__PCHAR__UncompressedFileName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`IN PCHAR p__PCHAR__UncompressedFileName`	Pointer to uncompressed file name.

Returns

Execution status.

Description

This function checks is file name actually points to supported uncompressed audio file.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349),
StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352),
StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348),
StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351),

StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileSupportedIs (see page 355)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_UncompressedFileSupportedIs API call.

2.1.289
StarBurn_StarWave_UncompressedFileSupportedUnicodeIs Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_UncompressedFileSupportedUnicodeIs(
    IN PWCHAR p__PWCHAR__UncompressedFileName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PWCHAR p__PWCHAR__UncompressedFileName	Pointer to uncompressed Unicode file name.

Returns

Execution status.

Description

This function checks is file name actually points to supported uncompressed audio file.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileSupportedIs (see page 355) StarBurn_StarWave_UncompressedFileSupportedUnicodeIsEx (see page 364)

2.1.290 StarBurn_StarWave_UncompressedFileSupportedUnicodeIsEx Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_UncompressedFileSupportedUnicodeIsEx(
    IN PWCHAR p__PWCHAR__UncompressedFileName,
    OUT PULONG p__PULONG__HeaderSizeInUCHARs,
    OUT PULONG p__PULONG__DataSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PWCHAR p__PWCHAR__UncompressedFileName	Pointer to uncompressed Unicode file name.

Returns

Execution status.

Description

This function checks is file name actually points to supported uncompressed audio file.

Remarks

None.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_VersionGet (see page 364), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileSupportedIs (see page 355), StarBurn_StarWave_UncompressedFileSupportedUnicodeIs (see page 363)

2.1.291 StarBurn_StarWave_VersionGet Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave_VersionGet(
    OUT PULONG p__PULONG__StarWaveVersion
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`OUT PULONG p__PULONG__StarWaveVersion`	Pointer to the variable to receive StarWave build version.

Returns

Execution status. This call cannot fail unless wrong pointer is passed as input parameter.

Description

This function returns StarWave library version.

Remarks

Actually StarWave version build number is synchronized with StarBurn build number. So there's no real use in calling this API call any more.

See Also

StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349), StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352), StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348), StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351), StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350), StarBurn_StarWave_UncompressedFileSupportedIs (see page 362), StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358), StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360), StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359), StarBurn_StarWave_UncompressedFileCompress (see page 361), StarBurn_StarWave_CompressedFileUncompress (see page 356), StarBurn_StarWave_CompressedFileRecompress (see page 354), StarBurn_StarWave_CompressedFileSupportedIs (see page 355)

Example

See AudioCompressor StarBurn sample application as an example how to use StarBurn_StarWave_VersionGet API call.

2.1.292 StarBurn_StarWave2_Convert Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_StarWave2_Convert(
    PCHAR p__PCHAR__InputFileName,
    PCHAR p__PCHAR__OutputFileName,
    PSTARWAVE2_COMPRESSION_PROFILE p__PSTARWAVE2_COMPRESSION_PROFILE
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`i_input_file_name`	Pointer to CHAR array with input file name.
`i_output_file_name`	Pointer to CHAR array with output file name.
`ip_convert_settings`	Pointer to conversion settings structure. It depends on file writer.

Returns

True if convertion was successfull. False otherwice.

Description

This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)

Remarks

To use FLAC format you should register StarWaveFlac.dll via StarBurn_StarWave2_RegisterCustomConverter.

Example

See AudioConverter StarBurn sample application as an example how to use StarBurn_StarWave2_ConvertExEx. Use of StarBurn_StarWave2_Convert is similar with obvious changes.

2.1.293 StarBurn_StarWave2_ConvertEx Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_StarWave2_ConvertEx(
    CFuncStarWaveConversionCallback i_callback_function,
    PVOID ip_user_data,
    PCHAR i_input_file_name,
    PCHAR i_output_file_name,
    ULONG i_buffer_size,
    PSTARWAVE2_COMPRESSION_PROFILE p__PSTARWAVE2_COMPRESSION_PROFILE
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`CFuncStarWaveConversionCallback i_callback_function`	Address of callback function for notify about progress and force to stop process of conversion.
`PVOID ip_user_data`	Pointer to data that is passed to callback function
`PCHAR i_input_file_name`	Pointer to CHAR array with input file name.
`PCHAR i_output_file_name`	Pointer to CHAR array with output file name.
`ULONG i_buffer_size`	Size of intermediate buffer. If this parameter is 0 size will be 50000 byte.
`ip_convert_settings`	Pointer to conversion settings structure. It depends on file writer.

Returns

True if convertion was successfull. False otherwice.

Description

This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)

Remarks

To use FLAC format you should register StarWaveFlac.dll via StarBurn_StarWave2_RegisterCustomConverter.

Example

See AudioConverter StarBurn sample application as an example how to use StarBurn_StarWave2_ConvertExEx. Use of StarBurn_StarWave2_ConvertEx is similar with obvious changes.

2.1.294 StarBurn_StarWave2_ConvertExUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_StarWave2_ConvertExUnicode(
    CFuncStarWaveConversionCallback i_callback_function,
    PVOID ip_user_data,
    PWCHAR i_input_file_name,
    PWCHAR i_output_file_name,
    ULONG i_buffer_size,
    PSTARWAVE2_COMPRESSION_PROFILE p__PSTARWAVE2_COMPRESSION_PROFILE
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`CFuncStarWaveConversionCallback i_callback_function`	Address of callback function for notify about progress and force to stop process of conversion.
`PVOID ip_user_data`	Pointer to data that is passed to callback function
`PWCHAR i_input_file_name`	Pointer to WCHAR array with input file name.
`PWCHAR i_output_file_name`	Pointer to WCHAR array with output file name.
`ULONG i_buffer_size`	Size of intermediate buffer. If this parameter is 0 size will be 50000 byte.
`PSTARWAVE2_COMPRESSION_PROFILE p__PSTARWAVE2_COMPRESSION_PROFILE`	Pointer to conversion settings structure. It depends on file writer.

Returns

True if convertion was successfull. False otherwice.

Description

This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)

Remarks

To use FLAC format you should register StarWaveFlac.dll via StarBurn_StarWave2_RegisterCustomConverter.

Example

See AudioConverter StarBurn sample application as an example how to use StarBurn_StarWave2_ConvertEx (see page 366). Use of StarBurn_StarWave2_ConvertExUnicode is similar with obvious changes.

2.1.295 StarBurn_StarWave2_ConvertUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API BOOL StarBurn_StarWave2_ConvertUnicode(
    PWCHAR p__PWCHAR__InputFileName,
    PWCHAR p__PWCHAR__OutputFileName,
    PSTARWAVE2_COMPRESSION_PROFILE p__PSTARWAVE2_COMPRESSION_PROFILE
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`i_input_file_name`	Pointer to WCHAR array with input file name.

i_output_file_name	Pointer to WCHAR array with output file name.
ip_convert_settings	Pointer to conversion settings structure. It depends on file writer.

Returns

True if convertion was successfull. False otherwice.

Description

This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)

Remarks

To use FLAC format you should register StarWaveFlac.dll via StarBurn_StarWave2_RegisterCustomConverter.

Example

See AudioConverter StarBurn sample application as an example how to use StarBurn_StarWave2_ConvertExEx. Use of StarBurn_StarWave2_ConvertUnicode is similar with obvious changes.

2.1.296 StarBurn_StarWave2_EncodeMP3OGGFromWAV Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarWave2_EncodeMP3OGGFromWAV(
    IN PCHAR p__PCHAR_SrcFileName,
    IN PCHAR p__PCHAR_DstFileName,
    IN PSTARBURN_STARWAVE2_INIT_PARAMS p_PSTARBURN_STARWAVE2_INIT_PARAMS,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__CallbackContext,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PCHAR p__PCHAR_SrcFileName	Pointer to source file full path (WAV file)
IN PCHAR p__PCHAR_DstFileName	Pointer to destination file full path (MP3 or OGG file)
IN PSTARBURN_STARWAVE2_INIT_PARAMS p_PSTARBURN_STARWAVE2_INIT_PARAMS	Pointer to initialization structure (See STARBURN_STARWAVE2_INIT_PARAMS (see page 562))
IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK	Pointer to callback function.
IN PVOID p__PVOID__CallbackContext	Pointer to callback context.
OUT PCHAR p__PCHAR__ExceptionText	Error text buffer size in UCHARs.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Error text buffer size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error value (error code WIN32/Vorbis/Nlame). For detailed information see p__PCHAR__ExceptionText.

Returns

Return true if finish successful, otherwise false. If function return false see error code (p__PULONG__SystemError), and error detailed information (p__PCHAR__ExceptionText).

Description

This function compress audio file from WAV (44100Hz, 16, stereo) format to MP3 and OGG format.

Remarks

None.

Example

See AudioCompressor StarBurn sample applications as an example how to use

2.1.297 StarBurn_StarWave2_EncodeMP3OGGFromWAVUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_StarWave2_EncodeMP3OGGFromWAVUnicode(
    IN PWCHAR p__PWCHAR_SrcFileName,
    IN PWCHAR p__PWCHAR_DstFileName,
    IN PSTARBURN_STARWAVE2_INIT_PARAMS p_PSTARBURN_STARWAVE2_INIT_PARAMS,
    IN PSTARBURN_STARWAVE_CALLBACK p__PSTARBURN_STARWAVE_CALLBACK,
    IN PVOID p__PVOID__CallbackContext,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_StarWave2_EncodeMP3OGGFromWAVUnicode.

2.1.298 StarBurn_StarWave2_IsFileSupported Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave2_IsFileSupported(
    PCHAR i_input_file_name
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
i_filename	Pointer to CHAR array with file name.

Returns

Windows error. ERROR_SUCCESS if file is supported. ERROR_NOT_SUPPORTED otherwise.

Description

This function determines audio file by StarWave2 library supported is.

Remarks

None.

Example

```
ULONG res = StarBurn_StarWave2_IsFileSupported("track1.wav");

if(res == ERROR_SUCCESS)
{
    ... //doing smth with file
}
```

2.1.299 StarBurn_StarWave2_IsFileSupportedUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API ULONG StarBurn_StarWave2_IsFileSupportedUnicode(
    PWCHAR i_input_file_name
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
i_filename	Pointer to WCHAR array with file name.

Returns

Windows error. ERROR_SUCCESS if file is supported. ERROR_NOT_SUPPORTED otherwise.

Description

This function determines audio file by StarWave2 library supported is.

Remarks

None.

Example

```
ULONG res = StarBurn_StarWave2_IsFileSupportedUnicode(CA2W("track1.wav"));

if(res == ERROR_SUCCESS)
{
    ... //doing smth with file
}
```

2.1.300 StarBurn_strcpy_s Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_strcpy_s(
    char * strDestination,
    size_t numberOfElements,
    const char * strSource
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
char * strDestination	Location of destination string buffer.

size_t numberOfElements	Size of the destination string buffer.
const char * strSource	Null-terminated source string buffer.

Description

This function wrap runtime strcpy_s function. In Visual Studio 2003 function strcpy_s not present.

2.1.301 StarBurn_swprintf_s Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_swprintf_s(
    wchar_t * buffer,
    size_t sizeOfBuffer,
    const wchar_t * format,
    ...
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
wchar_t * buffer	Storage location for output.
size_t sizeOfBuffer	Maximum number of characters to store.
const wchar_t * format	Format-control string.
argument	Optional arguments.

Description

This function wrap runtime swprintf_s function. In Visual Studio 2003 oriinal swprintf_s not present.

2.1.302 StarBurn_UDF_Add Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDF_Add(
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCHAR p__PCHAR__DirectoryOrFileAbsolutePathAndName,
    IN PCHAR p__PCHAR__DirectoryOrFileNewName,
    IN PVOID p__PVOID__UDF__Parent,
    OUT PVOID * p__PPVOID__UDF__NewChild
);
```

File

StarBurn.h (see page 662)

Returns

Exception number

Description

Creates the UDF Tree Node

2.1.303 StarBurn_UDF_AddUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDF_AddUnicode(
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PWCHAR p__PWCHAR__DirectoryOrFileAbsolutePathAndName,
    IN PWCHAR p__PWCHAR__DirectoryOrFileNewName,
    IN PVOID p__PVOID__UDF__Parent,
    OUT PVOID * p__PPVOID__UDF__NewChild
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_UDF_AddUnicode.

2.1.304 StarBurn_UDF_CleanUp Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF_CleanUp(
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM,
    unsigned long p__ULONG__NumberOfTreeItems
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PUDF_TREE_ITEM p__PUDF_TREE_ITEM	Pointer to allocated and formatted as file UDF tree root node.
unsigned long p__ULONG__NumberOfTreeItems	Number of tree items to process.

Returns

Nothing. This function cannot fail.

Description

This function removes created UDF tree from the pointed UDF tree node sequentially. Processes file formatted UDF nodes and either closes file handles or frees memory (non-cached vs. cached content).

Remarks

Please check DVDVideoBuildImage and DVDVideoTrackAtOnceFromTree samples to find out how to use StarBurn_UDF_CleanUp() in the right way.

See Also

StarBurn_UDF_CreateEx (see page 374), StarBurn_UDF_Destroy (see page 377), StarBurn_UDF_FormatTreeItemAsDirectory (see page 379), StarBurn_UDF_FormatTreeItemAsFile (see page 382)

Example

This example allocates CdvdBurnerGrabber object, creates and destroys UDF tree and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
```

```
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UDF_TREE_ITEM l__UDF_TREE_ITEM__Directory[ 10 ];
UDF_TREE_ITEM l__UDF_TREE_ITEM__File[ 10 ];
UDF_CONTROL_BLOCK l__UDF_CONTROL_BLOCK;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add nodes with StarBurn_UDF_FormatTreeItemAsXxx here

// Create UDF tree with StarBurn_UDF_CreateEx() here

// Do something with the UDF tree here (maybe burn to CdvdBurnerGrabber object)

// Clean up UDF stuff
StarBurn_UDF_CleanUp(
    &l__UDF_TREE_ITEN__File[ 0 ],
    10
    );

// Destroy UDF tree from the root
StarBurn_UDF_Destroy(
    &l__UDF_TREE_ITEM__Directory[ 0 ],
    &l__UDF_CONTROL_BLOCK
    );

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.305 StarBurn_UDF_Create Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDF_Create(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__UDFRoot,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootVideo,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootAudio,
    IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCHAR p__PCHAR__VolumeLabel
);
```

File

StarBurn.h (see page 662)

Description

StarBurn_UDF_Create() function is obsolete. Use StarBurn_UDF_CreatEx() instead of it

2.1.306 StarBurn_UDF_CreateEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDF_CreateEx(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__UDFRoot,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootVideo,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootAudio,
    IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCHAR p__PCHAR__VolumeLabel,
    IN PCHAR p__PCHAR__PublisherPreparerName,
    IN PCHAR p__PCHAR__ApplicationName,
    IN LONG p__LONG__Year,
    IN LONG p__LONG__Month,
    IN LONG p__LONG__Day,
    IN LONG p__LONG__Hour,
    IN LONG p__LONG__Minute,
    IN LONG p__LONG__Second,
    IN LONG p__LONG__MilliSecond
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__UDFRoot	Pointer to UDF root node.
IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootVideo	Pointer to UDF root node for VIDEO_TS directory.
IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootAudio	Pointer to UDF root node for AUDIO_TS directory.
IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK	Pointer to UDF control block.
OUT PCHAR p__PCHAR__ExceptionText	Pointer to exception text.
IN ULONG p__ULONG__ExceptionTextSizeInUCHARs	Exception text size in UCHARs.
OUT PULONG p__PULONG__SystemError	Pointer to system error.
IN PCHAR p__PCHAR__VolumeLabel	Pointer to volume label for UDF image.

IN PCHAR p__PCHAR__PublisherPreparerName	Pointer to publisher and preparer name.
IN PCHAR p__PCHAR__ApplicationName	Pointer to application name.
IN LONG p__LONG__Year	Year of creation.
IN LONG p__LONG__Month	Month of creation.
IN LONG p__LONG__Day	Day of creation.
IN LONG p__LONG__Hour	Hour of creation.
IN LONG p__LONG__Minute	Minute of creation.
IN LONG p__LONG__Second	Second of creation.
IN LONG p__LONG__MilliSecond	Millisecond of creation.

Returns

Execution status. EN_SUCCESS if the operation completed successfully. If the exception number will be EN_SYSTEM_CALL_FAILED, variable that SystemError points to will be filled with system error. If something other then EN_SUCCESS will be returned buffer that ExceptionText will point to will be filled with formatted exception message.

Description

This function creates UDF tree from the allocated data buffers.

Remarks

Please check DVDVideoBuildImage and DVDVideoTrackAtOnceFromTree samples to find out how to use StarBurn_UDF_CreateEx() in the right way.

See Also

StarBurn_UDF_Destroy (see page 377), StarBurn_UDF_CleanUp (see page 372), StarBurn_UDF_FormatTreeItemAsDirectory (see page 379), StarBurn_UDF_FormatTreeItemAsFile (see page 382)

Example

This example allocates CdvdBurnerGrabber object, creates and destroys UDF tree and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UDF_TREE_ITEM l__UDF_TREE_ITEM__Directory[ 10 ];
UDF_CONTROL_BLOCK l__UDF_CONTROL_BLOCK;

unsigned char g__UCHAR__FileSystemHead[ ( UDF_HEAD_SIZE_IN_LOGICAL_BLOCKS *
UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ) ];
unsigned char g__UCHAR__FileSystemTail[ ( UDF_TAIL_SIZE_IN_LOGICAL_BLOCKS *
UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ) ];
unsigned char g__UCHAR__FileSystemStructures[ ( 1024 * UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ) ];

UDF_TREE_ITEM g__UDF_TREE_ITEM__File[ 200 ]; // File[ 0 ] is not used

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
```

```
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add nodes with StarBurn_UDF_FormatTreeItemAsXxx here

// Create UDF tree with StarBurn_UDF_CreateEx() here
l__EXCEPTION_NUMBER =
StarBurn_UDF_CreateEx(
    &l__UDF_TREE_ITEM__Directory[ 1 ], // This is ROOT
    &l__UDF_TREE_ITEM__Directory[ 2 ], // This is VIDEO_TS and not ROOT, for VIDEO_TS
listing
    &l__UDF_TREE_ITEM__Directory[ 3 ], // This is AUDIO_TS and not ROOT, for AUDIO_TS
listing
    &l__UDF_CONTROL_BLOCK,
    ( CHAR * )( &l__CHAR__ExceptionText ),
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    "VolumeLabel",
    "PublisherName",
    "ApplicationName",
    2006,   // Year
    10,     // Month
    20,     // Day,
    12,     // Hour
    10,     // Minute
    0,      // Second
    50      // Millisecond
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with the UDF tree here (maybe burn to CdvdBurnerGrabber object)

// Clean up UDF stuff with StarBurn_UDF_CleanUp() here

// Destroy UDF tree from the root
StarBurn_UDF_Destroy(
    &l__UDF_TREE_ITEM__Directory[ 0 ],
    &l__UDF_CONTROL_BLOCK
    );

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.307 StarBurn_UDF_CreateExUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDF_CreateExUnicode(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__UDFRoot,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootVideo,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootAudio,
    IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PWCHAR p__PWCHAR__VolumeLabel,
    IN PWCHAR p__PWCHAR__PublisherPreparerName,
    IN PWCHAR p__PWCHAR__ApplicationName,
    IN LONG p__LONG__Year,
    IN LONG p__LONG__Month,
    IN LONG p__LONG__Day,
    IN LONG p__LONG__Hour,
    IN LONG p__LONG__Minute,
    IN LONG p__LONG__Second,
    IN LONG p__LONG__MilliSecond
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_UDF_CreateExUnicode.

2.1.308 StarBurn_UDF_CreateUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDF_CreateUnicode(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__UDFRoot,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootVideo,
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__ISO9660UDFBridgeRootAudio,
    IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PWCHAR p__PWCHAR__VolumeLabel
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_UDF_CreateUnicode.

2.1.309 StarBurn_UDF_Destroy Function

C++

```
__stdcall STARBURN_IMPEX_API VOID StarBurn_UDF_Destroy(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Root,
```

```
    IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Root	Pointer to allocated and formatted UDF tree root node.
IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK	Pointer to UDF control block.

Returns

Nothing. This function cannot fail.

Description

This function devastates created UDF tree from the pointed root recursively.

Remarks

Please check DVDVideoBuildImage and DVDVideoTrackAtOnceFromTree samples to find out how to use StarBurn_UDF_Destroy() in the right way.

See Also

StarBurn_UDF_CreateEx (see page 374), StarBurn_UDF_CleanUp (see page 372), StarBurn_UDF_FormatTreeItemAsDirectory (see page 379), StarBurn_UDF_FormatTreeItemAsFile (see page 382)

Example

This example allocates CdvdBurnerGrabber object, creates and destroys UDF tree and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UDF_TREE_ITEM l__UDF_TREE_ITEM__Director[ 10 ];
UDF_CONTROL_BLOCK l__UDF_CONTROL_BLOCK;

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
```

```
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Add nodes with StarBurn_UDF_FormatTreeItemAsXxx here

// Create UDF tree with StarBurn_UDF_CreateEx() here

// Do something with the UDF tree here (maybe burn to CdvdBurnerGrabber object)

// Clean up UDF stuff with StarBurn_UDF_CleanUp() here

// Destroy UDF tree from the root
StarBurn_UDF_Destroy(
    &l__UDF_TREE_ITEM__Directory[ 0 ],
    &l__UDF_CONTROL_BLOCK
    );

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.310 StarBurn_UDF_DestroyNodeAndKids Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDF_DestroyNodeAndKids(
    IN PVOID p__PVOID__UDF__Parent
);
```

File

StarBurn.h (see page 662)

Returns

Exception number

Description

Destroys UDF Tree Node and all its kids

2.1.311 StarBurn_UDF_FormatTreeItemAsDirectory Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF_FormatTreeItemAsDirectory(
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM,
    unsigned long p__ULONG__GUID,
    char * p__PCHAR__Name,
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PUDF_TREE_ITEM p__PUDF_TREE_ITEM	Pointer to UDF tree item to format as file.
unsigned long p__ULONG__GUID	Globally unique identifier.
char * p__PCHAR__Name	Pointer to name of the node to use in UDF volume.
PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Parent	Pointer to parent node.

Returns

Nothing. This function cannot fail.

Description

This function formats UDF tree item as directory.

Remarks

Please check DVDVideoBuildImage and DVDVideoTrackAtOnceFromTree samples to find out how to use StarBurn_UDF_FormatTreeItemAsDirectory() in the right way.

See Also

StarBurn_UDF_Destroy (see page 377), StarBurn_UDF_CleanUp (see page 372), StarBurn_UDF_FormatTreeItemAsFile (see page 382)

Example

This example allocates CdvdBurnerGrabber object, creates and destroys UDF tree and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UDF_TREE_ITEM l__UDF_TREE_ITEM__Directory[ 10 ];
UDF_CONTROL_BLOCK l__UDF_CONTROL_BLOCK;

unsigned char g__UCHAR__FileSystemHead[ ( UDF_HEAD_SIZE_IN_LOGICAL_BLOCKS *
UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ) ];
unsigned char g__UCHAR__FileSystemTail[ ( UDF_TAIL_SIZE_IN_LOGICAL_BLOCKS *
UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ) ];
unsigned char g__UCHAR__FileSystemStructures[ ( 1024 * UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ) ];

UDF_TREE_ITEM g__UDF_TREE_ITEM__File[ 200 ]; // File[ 0 ] is not used

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
```

```
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Format VIDEO_TS directory
StarBurn_UDF_FormatTreeItemAsDirectory(
    &l__UDF_TREE_ITEM__Directory[ 2 ],
    ( ++l__ULONG__GUID ),
    "VIDEO_TS",
    &l__UDF_TREE_ITEM__Directory[ 1 ]
    );

// Format file node 1 as file
l__ULONG__SystemError =
StarBurn_UDF_FormatTreeItemAsFile(
    &g__UDF_TREE_ITEM__File[ 1 ],
    ( ++l__ULONG__GUID ),
    "video_ts.ifo",
    "c:\\VOB\video_ts.ifo",
    &l__UDF_TREE_ITEM__Directory[ 2 ]
    );

// Handle error case here if l__ULONG__SystemError is not NO_ERROR...

// Create UDF tree with StarBurn_UDF_CreateEx() here
l__EXCEPTION_NUMBER =
StarBurn_UDF_CreateEx(
    ( PUCHAR )( &g__UCHAR__FileSystemHead ),
    sizeof( g__UCHAR__FileSystemHead ),
    ( PUCHAR )( &g__UCHAR__FileSystemTail ),
    sizeof( g__UCHAR__FileSystemTail ),
    ( PUCHAR )( &g__UCHAR__FileSystemStructures ),
    &l__UDF_TREE_ITEM__Directory[ 1 ], // This is ROOT
    &l__UDF_TREE_ITEM__Directory[ 2 ], // This is VIDEO_TS and not ROOT, for VIDEO_TS
listing
    &l__UDF_TREE_ITEM__Directory[ 3 ], // This is AUDIO_TS and not ROOT, for AUDIO_TS
listing
    &l__UDF_CONTROL_BLOCK,
    ( CHAR * )( &l__CHAR__ExceptionText ),
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    "VolumeLabel",
    "PublisherName",
    "ApplicationName",
    2006,   // Year
    10,     // Month
    20,     // Day,
    12,     // Hour
    10,     // Minute
    0,      // Second
    50      // Millisecond
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with the UDF tree here (maybe burn to CdvdBurnerGrabber object)

// Clean up UDF stuff with StarBurn_UDF_CleanUp() here

// Destroy UDF tree from the root
StarBurn_UDF_Destroy(
    &l__UDF_TREE_ITEM__Directory[ 0 ],
```

```
    &l__UDF_CONTROL_BLOCK
    );

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.312 StarBurn_UDF_FormatTreeItemAsDirectoryUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF_FormatTreeItemAsDirectoryUnicode(
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM,
    unsigned long p__ULONG__GUID,
    WCHAR * p__PWCHAR__Name,
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Parent
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_UDF_FormatTreeItemAsDirectoryUnicode.

2.1.313 StarBurn_UDF_FormatTreeItemAsFile Function

C++

```
__stdcall STARBURN_IMPEX_API unsigned long StarBurn_UDF_FormatTreeItemAsFile(
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM,
    unsigned long p__ULONG__GUID,
    char * p__PCHAR__Name,
    char * p__PCHAR__FullPath,
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
PUDF_TREE_ITEM p__PUDF_TREE_ITEM	Pointer to UDF tree item to format as file.
unsigned long p__ULONG__GUID	Globally unique identifier.
char * p__PCHAR__Name	Pointer to name of the node to use in UDF volume.
char * p__PCHAR__FullPath	Pointer to file path on the disk.
PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Parent	Pointer to parent node.

Returns

Execution status. NO_ERROR means everything is OK and Win32 system error is returned otherwise.

Description

This function formats UDF tree item as file.

Remarks

Please check DVDVideoBuildImage and DVDVideoTrackAtOnceFromTree samples to find out how to use StarBurn_UDF_FormatTreeItemAsFile() in the right way.

See Also

StarBurn_UDF_Destroy (see page 377), StarBurn_UDF_CleanUp (see page 372), StarBurn_UDF_FormatTreeItemAsDirectory (see page 379)

Example

This example allocates CdvdBurnerGrabber object, creates and destroys UDF tree and destroys the device object after it's not needed any more.

```
// Somewhere in the data region
PVOID l__PVOID__CdvdBurnerGrabber;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER;
ULONG l__ULONG__SystemError;
CDB_FAILURE_INFORMATION l__CDB_FAILURE_INFORMATION;
UDF_TREE_ITEM l__UDF_TREE_ITEM__Directory[ 10 ];
UDF_CONTROL_BLOCK l__UDF_CONTROL_BLOCK;
UDF_TREE_ITEM g__UDF_TREE_ITEM__File[ 200 ]; // File[ 0 ] is not used

// Prepare exception text buffer
RtlZeroMemory(
    &l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText )
    );

// Prepare CDB failure information
RtlZeroMemory(
    &l__CDB_FAILURE_INFORMATION,
    sizeof( l__CDB_FAILURE_INFORMATION )
    );

// Try to create CdvdBurnerGrabber on 0:0:4:0 with 32MB of cache
l__EXCEPTION_NUMBER =
StarBurn_CdvdBurnerGrabber_Create(
    &l__PVOID__CdvdBurnerGrabber,
    l__CHAR__ExceptionText,
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__SystemError,
    &l__CDB_FAILURE_INFORMATION,
    ( PCALLBACK )( StarBurn_Callback ),
    0,
    0,
    4,
    0,
    32
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Format file node 1 as file
l__ULONG__SystemError =
StarBurn_UDF_FormatTreeItemAsFile(
    &g__UDF_TREE_ITEM__File[ 1 ],
    ( ++l__ULONG__GUID ),
    "video_ts.ifo",
    "c:\VOB\video_ts.ifo",
    &l__UDF_TREE_ITEM__Directory[ 2 ]
    );
```

```
// Handle error case here if l__ULONG__SystemError is not NO_ERROR...

// Create UDF tree with StarBurn_UDF_CreateEx() here
l__EXCEPTION_NUMBER =
StarBurn_UDF_CreateEx(
    &l__UDF_TREE_ITEM__Directory[ 1 ], // This is ROOT
    &l__UDF_TREE_ITEM__Directory[ 2 ], // This is VIDEO_TS and not ROOT, for VIDEO_TS
listing
    &l__UDF_TREE_ITEM__Directory[ 3 ], // This is AUDIO_TS and not ROOT, for AUDIO_TS
listing
    &l__UDF_CONTROL_BLOCK,
    ( CHAR * )( &l__CHAR__ExceptionText ),
    sizeof( l__CHAR__ExceptionText ),
    &l__ULONG__Status,
    "VolumeLabel",
    "PublisherName",
    "ApplicationName",
    2006,   // Year
    10,     // Month
    20,     // Day,
    12,     // Hour
    10,     // Minute
    0,      // Second
    50      // Millisecond
    );

// Check for correct reply
if (  l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}

// Do something with the UDF tree here (maybe burn to CdvdBurnerGrabber object)

// Clean up UDF stuff with StarBurn_UDF_CleanUp() here

// Destroy UDF tree from the root
StarBurn_UDF_Destroy(
    &l__UDF_TREE_ITEM__Directory[ 0 ],
    &l__UDF_CONTROL_BLOCK
    );

// Destroy the CdvdBurnerGrabber
StarBurn_Destroy( &l__PVOID__CdvdBurnerGrabber );

// Just check for pointer (paranoid?)
if ( l__PVOID__CdvdBurnerGrabber != NULL )
{
// Handle error here...
}
```

2.1.314 StarBurn_UDF_FormatTreeItemAsFileUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API unsigned long StarBurn_UDF_FormatTreeItemAsFileUnicode(
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM,
    unsigned long p__ULONG__GUID,
    CONST WCHAR * p__PWCHAR__Name,
    CONST WCHAR * p__PWCHAR__FullPath,
    PUDF_TREE_ITEM p__PUDF_TREE_ITEM__Parent
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_UDF_FormatTreeItemAsFileUnicode.

2.1.315 StarBurn_UDF_GetFirstChild Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_UDF_GetFirstChild(
    IN PVOID p__PVOID__UDF__Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__UDF__Parent	Pointer to file tree node

Returns

Pointer to first child node of passed node

Description

This function returns the pointer to first child node

2.1.316 StarBurn_UDF_GetNextSibling Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_UDF_GetNextSibling(
    IN PVOID p__PVOID__UDF__Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__UDF__Parent	Pointer to file tree node

Returns

Pointer to next sibling of passed node

Description

This function returns the pointer to next sibling node

2.1.317 StarBurn_UDF_GetNodeObject Function

C++

```
__stdcall STARBURN_IMPEX_API PUDF_TREE_ITEM StarBurn_UDF_GetNodeObject(
    IN PVOID p__PVOID__UDF__Parent
);
```

File

StarBurn.h (see page 662)

Returns

Pointer to UDF_TREE_ITEM (see page 577) object associated with passed node

Description

Returns the object to current Tree Node

2.1.318 StarBurn_UDF_GetParent Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_UDF_GetParent(
    IN PVOID p__PVOID__UDF__Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__UDF__Parent	Pointer to file tree node

Returns

Pointer to parent node of passed node

Description

This function returns the pointer to next parent node

2.1.319 StarBurn_UDF_GetPreviousSibling Function

C++

```
__stdcall STARBURN_IMPEX_API PVOID StarBurn_UDF_GetPreviousSibling(
    IN PVOID p__PVOID__UDF__Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__UDF__Parent	Pointer to file tree node

Returns

Pointer to next sibling of passed node

Description

This function returns the pointer to previous sibling node

2.1.320 StarBurn_UDF2_DirectoryBrowse Function

C++

```
__stdcall STARBURN_IMPEX_API void * StarBurn_UDF2_DirectoryBrowse(
    void * Directory,
    void * Last
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Directory	Pointer to UDF directory object.
void * Last	Pointer to last processed kid node (or NULL if we need first kid node in the kids list).

Returns

Pointer to currently processed kid node (or NULL if we've reached end of the kids list).

Description

This function browses UDF directory content.

2.1.321 StarBurn_UDF2_DirectoryContentsProcess Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryContentsProcess(
    const char * PathName,
    unsigned long * GUID,
    void * Root,
    PSTARBURN_UDF2_PROGRESS_CALLBACK Callback,
    const void * Context,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char * PathName	Directory path and name (zero-terminated string).
unsigned long * GUID	Pointer to GUID.
void * Root	Pointer to directory where contents should be added.
PSTARBURN_UDF2_PROGRESS_CALLBACK Callback	Pointer to progress callback.
const void * Context	Pointer to context value.
STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to date and time structure.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes contents of single UDF directory.

2.1.322 StarBurn_UDF2_DirectoryContentsProcessEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryContentsProcessEx(
    const char * PathName,
    unsigned long * GUID,
    void * Root,
    PSTARBURN_UDF2_PROGRESS_CALLBACK Callback,
    const void * Context,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    PSTARBURN_UDF2_COLLISION_CALLBACK_EX CollisionCallback,
    const void* CollisionContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char * PathName	Directory path and name (zero-terminated string).
unsigned long * GUID	Pointer to GUID.
void * Root	Pointer to directory where contents should be added.
PSTARBURN_UDF2_PROGRESS_CALLBACK Callback	Pointer to progress callback.
const void * Context	Pointer to context value.
STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to date and time structure.
PSTARBURN_UDF2_COLLISION_CALLBACK_EX CollisionCallback	Pointer to name collision callback
const void* CollisionContext	Pointer to context value.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes contents of single UDF directory.

2.1.323 StarBurn_UDF2_DirectoryContentsUnicodeProcess Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryContentsUnicodeProcess(
    const unsigned short* UnicodePathName,
    unsigned long* GUID,
    void* Root,
    PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK ProgressCallback,
    const void* ProgressContext,
    STARBURN_UDF2_FILE_DATE_TIME* DateTime,
    PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX CollisionCallback,
    const void* CollisionContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short* UnicodePathName	Directory path and name (zero-terminated string in UNICODE format).
unsigned long* GUID	Pointer to GUID.
void* Root	Pointer to directory where contents should be added.
STARBURN_UDF2_FILE_DATE_TIME* DateTime	Pointer to date and time structure.
Callback	Pointer to UDF UNICODE progress callback.
Context	Pointer to context value.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes contents of single UNICODE UDF directory.

2.1.324
StarBurn_UDF2_DirectoryContentsUnicodeProcessEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryContentsUnicodeProcessEx(
    const unsigned short* UnicodePathName,
    unsigned long* GUID,
    void* Root,
    PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK ProgressCallback,
    const void* ProgressContext,
    STARBURN_UDF2_FILE_DATE_TIME* DateTime,
    PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX CollisionCallback,
    const void* CollisionContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short* UnicodePathName	Directory path and name (zero-terminated string in UNICODE format).
unsigned long* GUID	Pointer to GUID.
void* Root	Pointer to directory where contents should be added.
STARBURN_UDF2_FILE_DATE_TIME* DateTime	Pointer to date and time structure.
PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX CollisionCallback	Pointer to name collision callback
const void* CollisionContext	Pointer to context value.
Callback	Pointer to UDF UNICODE progress callback.
Context	Pointer to context value.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes contents of single UNICODE UDF directory.

2.1.325 StarBurn_UDF2_DirectoryCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryCreate(
    void ** Directory,
    const char * Name,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Directory	Pointer to pointer to directory object.
const char * Name	Pointer to zero-terminated directory name.
unsigned long * GUID	Pointer to GUID.
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF directory.

2.1.326 StarBurn_UDF2_DirectoryDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_DirectoryDestroy(
    void * Directory
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Directory	Pointer to directory object to destroy.

Returns

Nothing

Description

This function destroys created UDF directory object.

2.1.327 StarBurn_UDF2_DirectoryProcess Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryProcess(
    const char * Name,
```

```
    const char * PathName,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char * Name	Directory name itself (zero-terminated string).
const char * PathName	Directory path and name (zero-terminated string).
unsigned long * GUID	Pointer to GUID.
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single UDF directory.

2.1.328 StarBurn_UDF2_DirectoryProcessEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryProcessEx(
    const char * Name,
    const char * PathName,
    unsigned long * GUID,
    void * Parent,
    PSTARBURN_UDF2_PROGRESS_CALLBACK Callback,
    const void * Context,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char * Name	Directory name itself (zero-terminated string).
const char * PathName	Directory path and name (zero-terminated string).
unsigned long * GUID	Pointer to GUID.
void * Parent	Pointer to parent directory.
PSTARBURN_UDF2_PROGRESS_CALLBACK Callback	Pointer to progress callback.
const void * Context	Pointer to context value.
STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to date and time structure.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single UDF directory.

2.1.329 StarBurn_UDF2_DirectoryProcessExEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryProcessExEx(
    const char * Name,
    const char * PathName,
    unsigned long * GUID,
    void * Parent,
    PSTARBURN_UDF2_PROGRESS_CALLBACK Callback,
    const void * Context,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    PSTARBURN_UDF2_COLLISION_CALLBACK_EX CollisionCallback,
    const void* CollisionContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const char * Name	Directory name itself (zero-terminated string).
const char * PathName	Directory path and name (zero-terminated string).
unsigned long * GUID	Pointer to GUID.
void * Parent	Pointer to parent directory.
PSTARBURN_UDF2_PROGRESS_CALLBACK Callback	Pointer to progress callback.
const void * Context	Pointer to context value.
STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to date and time structure.
PSTARBURN_UDF2_COLLISION_CALLBACK_EX CollisionCallback	Pointer to name collision callback
const void* CollisionContext	Pointer to context value.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single UDF directory.

2.1.330 StarBurn_UDF2_DirectoryQuery Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_DirectoryQuery(
    void * Directory,
    STARBURN_UDF2_DIRECTORY_INFO * Info
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Directory	Pointer to UDF directory object.
STARBURN_UDF2_DIRECTORY_INFO * Info	Pointer to UDF directory information structure.

Returns

Nothing

Description

This function does query for UDF directory properties.

2.1.331 StarBurn_UDF2_DirectoryRootCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryRootCreate(
    void ** Directory,
    unsigned long * GUID
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Directory	Pointer to pointer to directory object.
unsigned long * GUID	Pointer to GUID.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates root UDF directory.

2.1.332 StarBurn_UDF2_DirectoryUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryUnicodeCreate(
    void ** Directory,
    const unsigned short * UnicodeName,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Directory	Pointer to pointer to directory object.
const unsigned short * UnicodeName	Pointer to zero-terminated directory name (UNICODE format).
unsigned long * GUID	Pointer to GUID.
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UNICODE UDF directory.

2.1.333 StarBurn_UDF2_DirectoryUnicodeProcess Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryUnicodeProcess(
    const unsigned short * UnicodeName,
    const unsigned short * UnicodePathName,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short * UnicodeName	Directory name itself (zero-terminated string in UNICODE format).
const unsigned short * UnicodePathName	Directory path and name (zero-terminated string in UNICODE format).
unsigned long * GUID	Pointer to GUID.
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single UNICODE UDF directory.

2.1.334 StarBurn_UDF2_DirectoryUnicodeProcessEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryUnicodeProcessEx(
    const unsigned short* UnicodeName,
    const unsigned short* UnicodePathName,
    unsigned long* GUID,
    void* Parent,
    PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK Callback,
    const void* Context,
    STARBURN_UDF2_FILE_DATE_TIME* DateTime
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short* UnicodeName	Directory name itself (zero-terminated string in UNICODE format).
const unsigned short* UnicodePathName	Directory path and name (zero-terminated string in UNICODE format).
unsigned long* GUID	Pointer to GUID.
void* Parent	Pointer to parent directory.
PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK Callback	Pointer to UDF UNICODE progress callback.
const void* Context	Pointer to context value.
STARBURN_UDF2_FILE_DATE_TIME* DateTime	Pointer to date and time structure.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single UNICODE UDF directory.

2.1.335 StarBurn_UDF2_DirectoryUnicodeProcessExEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_DirectoryUnicodeProcessExEx(
    const unsigned short* UnicodeName,
    const unsigned short* UnicodePathName,
    unsigned long* GUID,
    void* Parent,
    PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK Callback,
    const void* Context,
    STARBURN_UDF2_FILE_DATE_TIME* DateTime,
    PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX CollisionCallback,
    const void* CollisionContext
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
const unsigned short* UnicodeName	Directory name itself (zero-terminated string in UNICODE format).
const unsigned short* UnicodePathName	Directory path and name (zero-terminated string in UNICODE format).
unsigned long* GUID	Pointer to GUID.
void* Parent	Pointer to parent directory.
PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK Callback	Pointer to UDF UNICODE progress callback.
const void* Context	Pointer to context value.
STARBURN_UDF2_FILE_DATE_TIME* DateTime	Pointer to date and time structure.
PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX CollisionCallback	Pointer to name collision callback
const void* CollisionContext	Pointer to context value.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function processes single UNICODE UDF directory.

2.1.336 StarBurn_UDF2_FileBootCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_FileBootCreate(
    void ** File,
    const char * Name,
    const char * PathName,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** File`	Pointer to pointer to file object.
`const char * Name`	Pointer to zero-terminated name.
`const char * PathName`	Pointer to zero-terminated path and name.
`unsigned long * GUID`	Pointer to GUID.
`void * Parent`	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF boot file.

2.1.337 StarBurn_UDF2_FileBootUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_FileBootUnicodeCreate(
    void ** File,
    const unsigned short * UnicodeName,
    const unsigned short * UnicodePathName,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** File`	Pointer to pointer to file object.
`const unsigned short * UnicodeName`	Pointer to zero-terminated name (UNICODE format).
`const unsigned short * UnicodePathName`	Pointer to zero-terminated path and name (UNICODE format).
`unsigned long * GUID`	Pointer to GUID.
`void * Parent`	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UNICODE UDF boot file.

2.1.338 StarBurn_UDF2_FileCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_FileCreate(
    void ** File,
    const char * Name,
    const char * PathName,
    unsigned long * GUID,
    void * Parent
```

```
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** File`	Pointer to pointer to file object.
`const char * Name`	Pointer to zero-terminated name.
`const char * PathName`	Pointer to zero-terminated path and name.
`unsigned long * GUID`	Pointer to GUID.
`void * Parent`	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF file.

2.1.339 StarBurn_UDF2_FileDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileDestroy(
    void * File
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * File`	Pointer to file object to destroy.

Returns

Nothing

Description

This function destroys created UDF file object.

2.1.340 StarBurn_UDF2_FileDirectoryDateTimeGet Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileDirectoryDateTimeGet(
    void * FileDirectory,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to file or directory object.
`STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to output file date and time.

Returns

Nothing

Description

This function gets date and time from created UDF file or directory object.

2.1.341 StarBurn_UDF2_FileDirectoryDateTimeGetEx Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileDirectoryDateTimeGetEx(
    void * FileDirectory,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    STARBURN_UDF2_FILE_DATE_TIME_TYPE Type
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to file or directory object.
`STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to input file date and time.
`DateType`	Type of date and time to be set

Returns

Nothing

Description

This function gets date and time from created UDF file or directory object.

It may get creation, last access, or last write date and time.

2.1.342 StarBurn_UDF2_FileDirectoryDateTimeSet Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileDirectoryDateTimeSet(
    void * FileDirectory,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to file or directory object.
`STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to input file date and time.

Returns

Nothing

Description

This function sets date and time on created UDF file or directory object.

2.1.343 StarBurn_UDF2_FileDirectoryDateTimeSetEx Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileDirectoryDateTimeSetEx(
    void * FileDirectory,
    STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    STARBURN_UDF2_FILE_DATE_TIME_TYPE DateType
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to file or directory object.
`STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to input file date and time.
`STARBURN_UDF2_FILE_DATE_TIME_TYPE DateType`	Type of date and time to be set

Returns

Nothing

Description

This function sets date and time on created UDF file or directory object.

It may set creation, last access, or last write date and time.

2.1.344 StarBurn_UDF2_FileDirectoryNameSet Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileDirectoryNameSet(
    void * FileDirectory,
    const char * Name
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to UDF file or directory object.
`const char * Name`	Pointer to new file or directory name.

Returns

Nothing

Description

This function set new name for UDF file or directory object.

2.1.345 StarBurn_UDF2_FileDirectoryUnicodeNameSet Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileDirectoryUnicodeNameSet(
    void * FileDirectory,
    const unsigned short * UnicodeName
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * FileDirectory`	Pointer to UDF file or directory object.
`const unsigned short * UnicodeName`	Pointer to new file or directory UNICODE name.

Returns

Nothing

Description

This function set new UNICODE name for UDF file or directory object.

2.1.346 StarBurn_UDF2_FileMemoryCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_FileMemoryCreate(
    void ** File,
    const char * Name,
    const void * MemoryRegion,
    unsigned long MemoryRegionSizeInUCHARs,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** File`	Pointer to pointer to file object.
`const char * Name`	Pointer to zero-terminated name.
`const void * MemoryRegion`	Pointer to memory region.
`unsigned long MemoryRegionSizeInUCHARs`	Memory region size in unsigned chars.
`unsigned long * GUID`	Pointer to GUID.
`void * Parent`	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF memory file.

2.1.347 StarBurn_UDF2_FileMemoryUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_FileMemoryUnicodeCreate(
    void ** File,
    const unsigned short * UnicodeName,
    const void * MemoryRegion,
    unsigned long MemoryRegionSizeInUCHARs,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** File`	Pointer to pointer to file object.
`const unsigned short * UnicodeName`	Pointer to zero-terminated name (UNICODE format).
`const void * MemoryRegion`	Pointer to memory region.
`unsigned long MemoryRegionSizeInUCHARs`	Memory region size in unsigned chars.
`unsigned long * GUID`	Pointer to GUID.
`void * Parent`	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UNICODE UDF memory file.

2.1.348 StarBurn_UDF2_FileQuery Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileQuery(
    void * File,
    STARBURN_UDF2_FILE_INFO * Info
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * File`	Pointer to UDF file object.
`STARBURN_UDF2_FILE_INFO * Info`	Pointer to UDF file information structure.

Returns

Nothing

Description

This function do query for UDF file properties.

Remarks

Some structure fields are valid only for imported or non-imported nodes. Refer to structure description.

2.1.349 StarBurn_UDF2_FileSetAttributes Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_FileSetAttributes(
    void* File,
    unsigned long Attributes
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void* File`	Pointer to UDF file object.
`unsigned long Attributes`	File attributes bit mask.

Returns

Nothing.

Description

This function sets attributes for UDF file.

2.1.350 StarBurn_UDF2_FileSystemGetSizes Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_FileSystemGetSizes(
    void * Root,
    UDF_VERSION Version,
    unsigned long * VolumeSizeInLBs,
    unsigned long long * VolumeSizeInBytes
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Root`	Pointer to root directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function calculates UDF file system sizes in Bytes and LBS on the fly (without build volume).

Version

UDF version (use only constants defined in versions like NAPALM_UDF_VERSION_1_02)

2.1.351 StarBurn_UDF2_FileUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_FileUnicodeCreate(
    void ** File,
    const unsigned short * UnicodeName,
    const unsigned short * UnicodePathName,
    unsigned long * GUID,
    void * Parent
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** File	Pointer to pointer to file object.
const unsigned short * UnicodeName	Pointer to zero-terminated name (UNICODE format).
const unsigned short * UnicodePathName	Pointer to zero-terminated path and name (UNICODE format).
unsigned long * GUID	Pointer to GUID.
void * Parent	Pointer to parent directory.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UNICODE UDF file.

2.1.352 StarBurn_UDF2_ImpVolumeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_ImpVolumeCreate(
    void ** Volume,
    void * Root,
    unsigned long * GUID,
    PSTARBURN_UDF2_IMPORT_CALLBACK ImportCallback,
    void * CallbackContext,
    unsigned long VolumeSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to UDF import volume object.
void * Root	Pointer to root directory.
unsigned long * GUID	Pointer to GUID.
PSTARBURN_UDF2_IMPORT_CALLBACK ImportCallback	Pointer to import callback.
void * CallbackContext	Pointer to callback context.
unsigned long VolumeSizeInLBs	Volume size in logical blocks.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF import volume.

2.1.353 StarBurn_UDF2_ImpVolumeDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_ImpVolumeDestroy(
    void * Volume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to UDF import volume object to destroy.

Returns

Nothing.

Description

This function destroys created UDF import volume object.

2.1.354 StarBurn_UDF2_ImpVolumeImport Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_ImpVolumeImport(
    void * Volume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to the UDF import volume object.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function imports UDF import volume content.

2.1.355 StarBurn_UDF2_ISO9660FileTreeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_ISO9660FileTreeCreate(
```

```
    void ** ISO9660FileTree,
    void * UdfVolume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** ISO9660FileTree`	Pointer to pointer to the object that toolkit will set to the ISO 9660 File Tree object it will allocate.
`void * UdfVolume`	Pointer to UDF volume object that toolkit allocated before.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function create ISO image from already created UDF volume.

2.1.356 StarBurn_UDF2_ISO9660FileTreeCreateEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_ISO9660FileTreeCreateEx(
    void ** ISO9660FileTree,
    void ** UdfVolumes,
    UINT numVolumes
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** ISO9660FileTree`	Pointer to pointer to the object that toolkit will set to the ISO 9660 File Tree object it will allocate.
`UdfVolume`	Pointer to UDF volume object that toolkit allocated before.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function create ISO image from already created UDF volume.

2.1.357 StarBurn_UDF2_VolumeAnchorGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeAnchorGet(
    void * Volume,
    void * Descriptor,
    unsigned long DescriptorSizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`void * Descriptor`	Pointer to descriptor buffer to read data to.
`unsigned long DescriptorSizeInUCHARs`	Descriptor buffer size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function gets first anchor volume descriptor pointer we'll have to replace on rewritable media.

2.1.358 StarBurn_UDF2_VolumeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeCreate(
    void ** Volume,
    void * Root,
    const char * Name,
    BOOL IsUnicode,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned char OSClass,
    unsigned long InitialVolumeSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** Volume`	Pointer to pointer to volume object.
`void * Root`	Pointer to root directory.
`const char * Name`	Pointer to zero-terminated string with volume name.
`BOOL IsUnicode`	Is UNICODE (should we convert single character nodes to UNICODE).
`BOOL IsGlobalDateTime`	Is global date and time (should we set all files and directories on the volume to global date and time).
`const STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to volume global date and time structure.
`unsigned char OSClass`	OS class to assign to the volume.
`unsigned long InitialVolumeSizeInLBs`	Initial volume size in logical blocks (offset to build volume from).

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF 1.02 volume.

2.1.359 StarBurn_UDF2_VolumeCreateBoot Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeCreateBoot(
    void ** Volume,
    void * Root,
    const char * Name,
    BOOL IsUnicode,
```

```
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    UDF_OS_CLASS OSClass,
    UDF_VERSION Version,
    unsigned long InitialVolumeSizeInLBs,
    const char * SystemIdentifier,
    const char * VolumeIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier,
    const char * ValidationIdentifier,
    unsigned char BootMediaType,
    unsigned short LoadSegment,
    unsigned char SystemType,
    unsigned char SectorCount
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** Volume`	Pointer to pointer to volume object.
`void * Root`	Pointer to root directory.
`const char * Name`	Pointer to zero-terminated string with volume name.
`BOOL IsUnicode`	Is UNICODE (should we convert single character nodes to UNICODE).
`BOOL IsGlobalDateTime`	Is global date and time (should we set all files and directories on the volume to global date and time).
`const STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to volume global date and time structure.
`UDF_OS_CLASS OSClass`	OS class to assign to the volume.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF Bootable volume.

Version

UDF version.

InitialVolumeSizeInLBs - Initial volume size in logical blocks (offset to build volume from).

SystemIdentifier - Pointer to zero-terminated string with volume name system identifier.

VolumeIdentifier - Pointer to zero-terminated string with volume identifier.

VolumeSetIdentifier - Pointer to zero-terminated string with volume set identifier.

PublisherIdentifier - Pointer to zero-terminated string with publisher identifier.

DataPreparerIdentifier - Pointer to zero-terminated string with data preparer identifier.

ApplicationIdentifier - Pointer to zero-terminated string with application identifier.

CopyrightFileIdentifier - Pointer to zero-terminated string with copyright file identifier.

AbstractFileIdentifier - Pointer to zero-terminated string with abstract file identifier.

BibliographicFileIdentifier - Pointer to zero-terminated string with bibliographic file identifier.

ValidationIdentifier - Pointer to zero-terminated string with volume name validation identifier.

BootMediaType - Media type of bootable volume.

LoadSegment - Load segment

SystemType - System type

SectorCount - Number of sectors to load from boot image.

2.1.360 StarBurn_UDF2_VolumeCreateBootElTorito Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeCreateBootElTorito(
    void ** Volume,
    void * Root,
    const char * Name,
    BOOL IsUnicode,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    UDF_OS_CLASS OSClass,
    UDF_VERSION Version,
    unsigned long InitialVolumeSizeInLBs,
    void * BootCatalog,
    const char * SystemIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const char * Name	Pointer to zero-terminated string with volume name.
BOOL IsUnicode	Is UNICODE (should we convert single character nodes to UNICODE).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
UDF_OS_CLASS OSClass	OS class to assign to the volume.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF Bootable volume.

Version

UDF version.

InitialVolumeSizeInLBs - Initial volume size in logical blocks (offset to build volume from).

BootCatalog - Pointer to ElTorito boot catalog

SystemIdentifier - Pointer to zero-terminated string with volume name system identifier.

VolumeSetIdentifier - Pointer to zero-terminated string with volume set identifier.

PublisherIdentifier - Pointer to zero-terminated string with publisher identifier.

DataPreparerIdentifier - Pointer to zero-terminated string with data preparer identifier.

ApplicationIdentifier - Pointer to zero-terminated string with application identifier.

CopyrightFileIdentifier - Pointer to zero-terminated string with copyright file identifier.

AbstractFileIdentifier - Pointer to zero-terminated string with abstract file identifier.

BibliographicFileIdentifier - Pointer to zero-terminated string with bibliographic file identifier.

2.1.361 StarBurn_UDF2_VolumeCreateEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeCreateEx(
    void ** Volume,
    void * Root,
    const char * Name,
    BOOL IsUnicode,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    UDF_OS_CLASS OSClass,
    UDF_VERSION Version,
    unsigned long InitialVolumeSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const char * Name	Pointer to zero-terminated string with volume name.
BOOL IsUnicode	Is UNICODE (should we convert single character nodes to UNICODE).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
UDF_OS_CLASS OSClass	OS class to assign to the volume.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF volume.

Version

UDF version (use only constants defined in versions like NAPALM_UDF_VERSION_1_02)

InitialVolumeSizeInLBs - Initial volume size in logical blocks (offset to build volume from).

2.1.362 StarBurn_UDF2_VolumeCreateUnicodeBootElTorito Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeCreateUnicodeBootElTorito(
    void ** Volume,
    void * Root,
    unsigned short * UnicodeName,
    BOOL IsUnicode,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    UDF_OS_CLASS OSClass,
    UDF_VERSION Version,
    unsigned long InitialVolumeSizeInLBs,
    void * BootCatalog,
    unsigned short * SystemIdentifier,
    unsigned short * VolumeSetIdentifier,
    unsigned short * PublisherIdentifier,
    unsigned short * DataPreparerIdentifier,
    unsigned short * ApplicationIdentifier,
    unsigned short * CopyrightFileIdentifier,
    unsigned short * AbstractFileIdentifier,
    unsigned short * BibliographicFileIdentifier
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
BOOL IsUnicode	Is UNICODE (should we convert single character nodes to UNICODE).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
UDF_OS_CLASS OSClass	OS class to assign to the volume.
Name	Pointer to zero-terminated string with volume name.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF Bootable volume.

Version

UDF version.

InitialVolumeSizeInLBs - Initial volume size in logical blocks (offset to build volume from).

BootCatalog - Pointer to ElTorito boot catalog

SystemIdentifier - Pointer to zero-terminated string with volume name system identifier.

VolumeSetIdentifier - Pointer to zero-terminated string with volume set identifier.

PublisherIdentifier - Pointer to zero-terminated string with publisher identifier.

DataPreparerIdentifier - Pointer to zero-terminated string with data preparer identifier.

ApplicationIdentifier - Pointer to zero-terminated string with application identifier.

CopyrightFileIdentifier - Pointer to zero-terminated string with copyright file identifier.

AbstractFileIdentifier - Pointer to zero-terminated string with abstract file identifier.

BibliographicFileIdentifier - Pointer to zero-terminated string with bibliographic file identifier.

2.1.363 StarBurn_UDF2_VolumeDestroy Function

C++

```
__stdcall STARBURN_IMPEX_API void StarBurn_UDF2_VolumeDestroy(
    void * Volume
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Volume	Pointer to UDF volume object to destroy.

Returns

Nothing

Description

This function destroys created UDF volume object.

2.1.364 StarBurn_UDF2_VolumeInitialSizeInLBsGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeInitialSizeInLBsGet(
    void * Volume,
    long * InitialSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void * Volume	Pointer to volume object.
SizeInUCHARs	Pointer to the variable to receive UDF volume initial size in logical blocks.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function get UDF volume size in unsigned chars.

2.1.365 StarBurn_UDF2_VolumeSeekRead Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeSeekRead(
    void * Volume,
    void * Buffer,
    unsigned long BufferSizeInLogicalBlocks,
    unsigned long OffsetInLogicalBlocks
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`void * Buffer`	Pointer to data buffer to read data to.
`unsigned long BufferSizeInLogicalBlocks`	Buffer size in logical blocks.
`unsigned long OffsetInLogicalBlocks`	Offset within UDF volume in logical blocks.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function seeks to passed offset and reads requested amount of data.

2.1.366 StarBurn_UDF2_VolumeSizeInLBsGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeSizeInLBsGet(
    void * Volume,
    unsigned long * SizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`SizeInUCHARs`	Pointer to the variable to receive UDF volume size in logical blocks.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function get UDF volume size in unsigned chars.

2.1.367 StarBurn_UDF2_VolumeSizeInUCHARsGet Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeSizeInUCHARsGet(
    void * Volume,
    unsigned __int64 * SizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`unsigned __int64 * SizeInUCHARs`	Pointer to the variable to receive UDF volume size in unsigned chars.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function get UDF volume size in unsigned chars.

2.1.368 StarBurn_UDF2_VolumeStore Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeStore(
    void * Volume,
    const char * Name
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void * Volume`	Pointer to volume object.
`const char * Name`	Pointer to zero-terminated string with file name to store volume to.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function stores UDF volume to file.

2.1.369 StarBurn_UDF2_VolumeUnicodeCreate Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeUnicodeCreate(
    void ** Volume,
```

```
    void * Root,
    const unsigned short * UnicodeName,
    BOOL IsUnicode,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    unsigned char OSClass,
    unsigned long InitialVolumeSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
const unsigned short * UnicodeName	Pointer to zero-terminated string with volume name (in UNICODE).
BOOL IsUnicode	Is UNICODE (should we convert single character nodes to UNICODE).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
unsigned char OSClass	OS class to assign to the volume.
unsigned long InitialVolumeSizeInLBs	Initial volume size in logical blocks (offset to build volume from).

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF 1.02 UNICODE volume.

2.1.370 StarBurn_UDF2_VolumeUnicodeCreateBoot Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeUnicodeCreateBoot(
    void ** Volume,
    void * Root,
    unsigned short * UnicodeName,
    BOOL IsUnicode,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
    UDF_OS_CLASS OSClass,
    UDF_VERSION Version,
    unsigned long InitialVolumeSizeInLBs,
    const char * SystemIdentifier,
    const char * VolumeIdentifier,
    const char * VolumeSetIdentifier,
    const char * PublisherIdentifier,
    const char * DataPreparerIdentifier,
    const char * ApplicationIdentifier,
    const char * CopyrightFileIdentifier,
    const char * AbstractFileIdentifier,
    const char * BibliographicFileIdentifier,
    const char * ValidationIdentifier,
    unsigned char BootMediaType,
    unsigned short LoadSegment,
    unsigned char SystemType,
    unsigned char SectorCount
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
void ** Volume	Pointer to pointer to volume object.
void * Root	Pointer to root directory.
unsigned short * UnicodeName	Pointer to zero-terminated string with volume name (in UNICODE).
BOOL IsUnicode	Is UNICODE (should we convert single character nodes to UNICODE).
BOOL IsGlobalDateTime	Is global date and time (should we set all files and directories on the volume to global date and time).
const STARBURN_UDF2_FILE_DATE_TIME * DateTime	Pointer to volume global date and time structure.
UDF_OS_CLASS OSClass	OS class to assign to the volume.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF Unicode Bootable volume.

Version

UDF version.

InitialVolumeSizeInLBs - Initial volume size in logical blocks (offset to build volume from).

SystemIdentifier - Pointer to zero-terminated string with volume name system identifier.

VolumeIdentifier - Pointer to zero-terminated string with volume identifier.

VolumeSetIdentifier - Pointer to zero-terminated string with volume set identifier.

PublisherIdentifier - Pointer to zero-terminated string with publisher identifier.

DataPreparerIdentifier - Pointer to zero-terminated string with data preparer identifier.

ApplicationIdentifier - Pointer to zero-terminated string with application identifier.

CopyrightFileIdentifier - Pointer to zero-terminated string with copyright file identifier.

AbstractFileIdentifier - Pointer to zero-terminated string with abstract file identifier.

BibliographicFileIdentifier - Pointer to zero-terminated string with bibliographic file identifier.

ValidationIdentifier - Pointer to zero-terminated string with volume name validation identifier.

BootMediaType - Media type of bootable volume.

LoadSegment - Load segment

SystemType - System type

SectorCount - Number of sectors to load from boot image.

2.1.371 StarBurn_UDF2_VolumeUnicodeCreateEx Function

C++

```
__stdcall STARBURN_IMPEX_API long StarBurn_UDF2_VolumeUnicodeCreateEx(
    void ** Volume,
    void * Root,
    const unsigned short * UnicodeName,
    BOOL IsUnicode,
    BOOL IsGlobalDateTime,
    const STARBURN_UDF2_FILE_DATE_TIME * DateTime,
```

```
    UDF_OS_CLASS OSClass,
    UDF_VERSION Version,
    unsigned long InitialVolumeSizeInLBs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
`void ** Volume`	Pointer to pointer to volume object.
`void * Root`	Pointer to root directory.
`const unsigned short * UnicodeName`	Pointer to zero-terminated string with volume name (in UNICODE).
`BOOL IsUnicode`	Is UNICODE (should we convert single character nodes to UNICODE).
`BOOL IsGlobalDateTime`	Is global date and time (should we set all files and directories on the volume to global date and time).
`const STARBURN_UDF2_FILE_DATE_TIME * DateTime`	Pointer to volume global date and time structure.
`UDF_OS_CLASS OSClass`	OS class to assign to the volume.

Returns

If success then returns ERROR_SUCCESS, otherwise error code.

Description

This function creates UDF UNICODE volume.

Version

UDF version.

InitialVolumeSizeInLBs - Initial volume size in logical blocks (offset to build volume from).

2.1.372 StarBurn_UDFBridge_CreateEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDFBridge_CreateEx(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__UDFRoot,
    IN STARBURN_UDF_BRIDGE_TYPE p__UDF_BRIDGE_TYPE,
    IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PCHAR p__PCHAR__VolumeLabel,
    IN PCHAR p__PCHAR__PublisherPreparerName,
    IN PCHAR p__PCHAR__ApplicationName,
    IN LONG p__LONG__Year,
    IN LONG p__LONG__Month,
    IN LONG p__LONG__Day,
    IN LONG p__LONG__Hour,
    IN LONG p__LONG__Minute,
    IN LONG p__LONG__Second,
    IN LONG p__LONG__MilliSecond
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_UDFBridge_CreateEx.

2.1.373 StarBurn_UDFBridge_CreateExUnicode Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UDFBridge_CreateExUnicode(
    IN PUDF_TREE_ITEM p__PUDF_TREE_ITEM__UDFRoot,
    IN STARBURN_UDF_BRIDGE_TYPE p__UDF_BRIDGE_TYPE,
    IN PUDF_CONTROL_BLOCK p__PUDF_CONTROL_BLOCK,
    OUT PCHAR p__PCHAR__ExceptionText,
    IN ULONG p__ULONG__ExceptionTextSizeInUCHARs,
    OUT PULONG p__PULONG__SystemError,
    IN PWCHAR p__PWCHAR__VolumeLabel,
    IN PWCHAR p__PWCHAR__PublisherPreparerName,
    IN PWCHAR p__PWCHAR__ApplicationName,
    IN LONG p__LONG__Year,
    IN LONG p__LONG__Month,
    IN LONG p__LONG__Day,
    IN LONG p__LONG__Hour,
    IN LONG p__LONG__Minute,
    IN LONG p__LONG__Second,
    IN LONG p__LONG__MilliSecond
);
```

File

StarBurn.h (see page 662)

Description

This is function StarBurn_UDFBridge_CreateExUnicode.

2.1.374 StarBurn_UpStart Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UpStart(
    IN VOID
);
```

File

StarBurn.h (see page 662)

Returns

Execution status. EN_SUCCESS if initialization process completed successfully.

Description

This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used. License file StarBurn.key is assumed to be located near main application executable. Since StarBurn SDK V12 it's possible to use this call again. It will just enable using embedded evaluational StarBurn SDK key (file shipped with StarBurn SDK and called StarBurnKey.h).

Remarks

See samples for details how and when call StarBurn_UpStart() function.

See Also

See samples for details how and when call StarBurn_UpStart() function.

Example

This example checks for the toolkit version and initializes it.

```
// Somewhere in the data region
ULONG l__ULONG__Version;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER = EN_SUCCESS;

// Get toolkit version
l__ULONG__Version =
StarBurn_GetVersion();

// Check for correct number
if ( l__ULONG__Version < SUPPORTED_VERSION_NUMBER )
{
// Handle error here...
}

// Try to initialize toolkit
l__EXCEPTION_NUMBER =
StarBurn_UpStart();

// Check for success
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

2.1.375 StarBurn_UpStartEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UpStartEx(
    IN PVOID p__PVOID__RegistrationKey,
    IN ULONG p__ULONG__RegistrationKeySizeInUCHARs
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__RegistrationKey	Pointer to registration key.
IN ULONG p__ULONG__RegistrationKeySizeInUCHARs	Registration key size in UCHARs.

Returns

Execution status. EN_SUCCESS if initialization process completed successfully.

Description

This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used. The difference between this call and call to StarBurn_UpStart (see page 417)(...) is that this particular call assumes StarBurn key is being embedded to the application binary and StarBurn_UpStart (see page 417)(...) always looks for StarBurn.key near application executable. Use this API call to initialize your application using registry-stored debug parameters, use StarBurn_UpStart (see page 417)(...) API call to initialize your app with embedded evaluational key and use advanced StarBurn_UpStartExEx (see page 419)(...) API call to pass user-defined extended debug parameters (unique to all burning applications using the same StarBurn SDK on the same machine).

Remarks

See samples for details how and when call StarBurn_UpStartEx() and StarBurn_UpStart (see page 417)() functions.

See Also

See samples for details how and when call StarBurn_UpStart (see page 417)() and StarBurn_UpStartEx() functions.

Example

This example checks for the toolkit version and initializes it.

```
// Somewhere in the data region
ULONG l__ULONG__Version;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER = EN_SUCCESS;
UCHAR l__UCHAR__RegistrationKey[ ] = { ... }; // Registration key goes here

// Get toolkit version
l__ULONG__Version =
StarBurn_GetVersion();

// Check for correct number
if ( l__ULONG__Version < SUPPORTED_VERSION_NUMBER )
{
// Handle error here...
}

// Try to initialize toolkit
l__EXCEPTION_NUMBER =
StarBurn_UpStartEx(
    ( PVOID )( &l__UCHAR__RegistrationKey ),
    sizeof( l__UCHAR__RegistrationKey )
    );

// Check for success
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

2.1.376 StarBurn_UpStartExEx Function

C++

```
__stdcall STARBURN_IMPEX_API EXCEPTION_NUMBER StarBurn_UpStartExEx(
    IN PVOID p__PVOID__RegistrationKey,
    IN ULONG p__ULONG__RegistrationKeySizeInUCHARs,
    IN ULONG p__ULONG__DebugFacility,
    IN PCHAR p__PCHAR__DebugFileName,
    IN HANDLE p__HANDLE__DebugCustomHandle
);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
IN PVOID p__PVOID__RegistrationKey	Pointer to registration key.
IN ULONG p__ULONG__RegistrationKeySizeInUCHARs	Registration key size in UCHARs.
IN ULONG p__ULONG__DebugFacility	Debug facility (None, debug output, log file, system console or custom). Please use next constant: STARBURN_DEBUG_FACILITY_NONE (see page 635), STARBURN_DEBUG_FACILITY_DEBUG_OUTPUT (see page 635), STARBURN_DEBUG_FACILITY_LOG_FILE (see page 635), STARBURN_DEBUG_FACILITY_SYSTEM_CONSOLE (see page 636), STARBURN_DEBUG_FACILITY_CUSTOM (see page 634)
IN PCHAR p__PCHAR__DebugFileName	Debug log file name (used with log file debug facility).
IN HANDLE p__HANDLE__DebugCustomHandle	Debug file handle (used with custom debug facility).

Returns

Execution status. EN_SUCCESS if initialization process completed successfully.

Description

This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Unlike StarBurn_UpStartEx (see page 418)(...) API call this one allows to use custom debug output control. See StarBurn_UpStartEx (see page 418)(...) API call for more information about this topic.

Remarks

See samples for details how and when call StarBurn_UpStartEx (see page 418)() and StarBurn_UpStart (see page 417)() functions.

See Also

See samples for details how and when call StarBurn_UpStart (see page 417)() and StarBurn_UpStartEx (see page 418)() functions.

Example

This example checks for the toolkit version and initializes it.

```
// Somewhere in the data region
ULONG l__ULONG__Version;
EXCEPTION_NUMBER l__EXCEPTION_NUMBER = EN_SUCCESS;
UCHAR l__UCHAR__RegistrationKey[ ] = { ... }; // Registration key goes here
HANDLE l__HANDLE__DebugLogFile = INVALID_HANDLE_VALUE;

// Get toolkit version
l__ULONG__Version =
StarBurn_GetVersion();

// Check for correct number
if ( l__ULONG__Version < SUPPORTED_VERSION_NUMBER )
{
// Handle error here...
}

// We assume l__HANDLE__DebugLogFile was initialized before

// Try to initialize toolkit
l__EXCEPTION_NUMBER =
StarBurn_UpStartExEx(
    ( PVOID )( &l__UCHAR__RegistrationKey ),
    sizeof( l__UCHAR__RegistrationKey ),
    STARBURN_DEBUG_FACILITY_CUSTOM,
    NULL,
    l__HANDLE__DebugLogFile
    );

// Check for success
if ( l__EXCEPTION_NUMBER != EN_SUCCESS )
{
// Handle error here...
}
```

2.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	_CALLBACK_NUMBER (see page 424)	Enum that represents callback number
	_DISC_TYPE (see page 432)	Enum that represents inserted disc type
	_ERASE_TYPE (see page 434)	Enum that represents erase type
	_EXCEPTION_NUMBER (see page 435)	Enum that represents exception number

	_FILE_TIME (see page 438)	Enum that represents file time that will be included in the ISO9660/Joliet image
	_FILE_TREE (see page 438)	Enum that represents file tree
	_ISO9660_KIDTYPE (see page 441)	Enum that represent type of ISO9660 tree node
	_NAME_COLLISION_SOLUTION (see page 442)	Enum that represents the solutions of name collisions
	_READ_MODE (see page 444)	Enum that represents raw read modes
	_STARBURN_CD_MODE (see page 449)	Enum that represents CD modes type
	_STARBURN_ELTORITO_MEDIA (see page 452)	Media emulation types
	_STARBURN_ELTORITO_PLATFORM (see page 452)	ElTorito platform types
	_STARBURN_STARWAVE_CALLBACK_REASON (see page 454)	StarWave callback reasons we'll be using
	_STARBURN_STARWAVE_COMPRESSION (see page 454)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	_STARBURN_STARWAVE2_COMPRESS_TYPE (see page 456)	Compress settings
	_STARBURN_STARWAVE2_CONVERSION_MODE (see page 456)	Structure that represents conversion mode
	_STARBURN_STARWAVE2_QUALITY_MODE (see page 458)	Enum that represents MP3 quality for conversion
	_STARBURN_UDF_BRIDGE_TYPE (see page 461)	Restore original structure packing
	_STARBURN_UDF2_FILE_DATE_TIME_TYPE (see page 462)	Enum that represents StarBurn date/time type for files in UDF
	_STARPORT_DEVICE_TYPE (see page 464)	Enum that represents StarPort device type
	_STARWAVE2_COMPRESSION (see page 465)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	_UDF_OS_CLASS (see page 472)	UDF OS class
	_UDF_VERSION (see page 474)	Enum that represents UDF version
	_WRITE_MODE (see page 476)	Enum that represents write modes
	CALLBACK_NUMBER (see page 476)	Enum that represents callback number
	DISC_TYPE (see page 484)	Enum that represents inserted disc type
	ERASE_TYPE (see page 486)	Enum that represents erase type
	EXCEPTION_NUMBER (see page 487)	Enum that represents exception number
	FILE_TIME (see page 490)	Enum that represents file time that will be included in the ISO9660/Joliet image
	FILE_TREE (see page 490)	Enum that represents file tree
	ISO9660_KIDTYPE (see page 493)	Enum that represent type of ISO9660 tree node
	NAME_COLLISION_SOLUTION (see page 494)	Enum that represents the solutions of name collisions
	PCALLBACK_NUMBER (see page 495)	Enum that represents callback number
	PDISC_TYPE (see page 502)	Enum that represents inserted disc type
	PERASE_TYPE (see page 504)	Enum that represents erase type
	PEXCEPTION_NUMBER (see page 505)	Enum that represents exception number
	PFILE_TIME (see page 508)	Enum that represents file time that will be included in the ISO9660/Joliet image
	PFILE_TREE (see page 508)	Enum that represents file tree
	PISO9660_KIDTYPE (see page 511)	Enum that represent type of ISO9660 tree node
	PNAME_COLLISION_SOLUTION (see page 512)	Enum that represents the solutions of name collisions
	PPSTARBURN_STARWAVE_CALLBACK_REASON (see page 515)	StarWave callback reasons we'll be using
	PPSTARBURN_STARWAVE_COMPRESSION (see page 515)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PPSTARPORT_DEVICE_TYPE (see page 518)	Enum that represents StarPort device type
	PPSTARWAVE2_COMPRESSION (see page 518)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PREAD_MODE (see page 521)	Enum that represents raw read modes
	PSTARBURN_CD_MODE (see page 526)	Enum that represents CD modes type
	PSTARBURN_STARWAVE_CALLBACK_REASON (see page 529)	StarWave callback reasons we'll be using
	PSTARBURN_STARWAVE_COMPRESSION (see page 530)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PSTARBURN_STARWAVE2_CONVERSION_MODE (see page 531)	Structure that represents conversion mode

	PSTARBURN_STARWAVE2_QUALITY_MODE (see page 533)	Enum that represents MP3 quality for conversion
	PSTARPORT_DEVICE_TYPE (see page 537)	Enum that represents StarPort device type
	PSTARWAVE2_COMPRESSION (see page 538)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PWRITE_MODE (see page 548)	Enum that represents write modes
	READ_MODE (see page 549)	Enum that represents raw read modes
	STARBURN_CD_MODE (see page 553)	Enum that represents CD modes type
	STARBURN_ELTORITO_MEDIA (see page 557)	Media emulation types
	STARBURN_ELTORITO_PLATFORM (see page 557)	ElTorito platform types
	STARBURN_STARWAVE_CALLBACK_REASON (see page 558)	StarWave callback reasons we'll be using
	STARBURN_STARWAVE_COMPRESSION (see page 559)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	STARBURN_STARWAVE2_COMPRESS_TYPE (see page 560)	Compress settings
	STARBURN_STARWAVE2_CONVERSION_MODE (see page 561)	Structure that represents conversion mode
	STARBURN_STARWAVE2_QUALITY_MODE (see page 562)	Enum that represents MP3 quality for conversion
	STARBURN_UDF_BRIDGE_TYPE (see page 565)	Restore original structure packing
	STARBURN_UDF2_FILE_DATE_TIME_TYPE (see page 567)	Enum that represents StarBurn date/time type for files in UDF
	STARPORT_DEVICE_TYPE (see page 569)	Enum that represents StarPort device type
	STARWAVE2_COMPRESSION (see page 570)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	UDF_OS_CLASS (see page 577)	UDF OS class
	UDF_VERSION (see page 579)	Enum that represents UDF version
	WRITE_MODE (see page 581)	Enum that represents write modes

Structures

	Name	Description
	_CDB_FAILURE_INFORMATION (see page 428)	Structure that represents CDB failure information
	_DAO_DISC_LAYOUT (see page 428)	Structure that represents DAO disc layout
	_DAO_DISC_LAYOUT_ENTRY (see page 429)	Structure that represents DAO disc layout entry (track)
	_DISC_FILESYSTEM (see page 430)	Structure with disc file system flags
	_DISC_LAYOUT (see page 430)	Structure that represents disc layout
	_DISC_LAYOUT_ENTRY (see page 431)	Structure that represents disc layout entry (track)
	_DVD_VIDEO_CONTROL_BLOCK (see page 434)	Structure that represents DVD-Video control block and pointer to DVD-Video control block
	_FULL_TOC_ENTRY_RAW (see page 439)	Structure that represents full TOC entry
	_ISO9660_DATE_TIME (see page 440)	Structure that represents ISO9660 date and time
	_NAME_COLLISION_INFO (see page 441)	Structure that represents information about name collision
	_PQ_SUBCHANNEL (see page 443)	Structure that represents formatted PQ sub-channel
	_SCSI_DEVICE_ADDRESS (see page 445)	Structure that represents SCSI device address
	_STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS (see page 445)	Structure that represents advanced supported media formats
	_STARBURN_BDRE_FORMAT_PROFILE (see page 448)	Structure that represents format profile for BD-RE media
	_STARBURN_DISC_ATIP_INFORMATION (see page 449)	Structure that represents Disc ATIP information
	_STARBURN_DISC_INFORMATION (see page 451)	Structure that represents Disc information
	_STARBURN_ISO9660_DIRECTORY_INFO (see page 453)	This is record _STARBURN_ISO9660_DIRECTORY_INFO.
	_STARBURN_ISO9660_FILE_INFO (see page 453)	ISO9660 file information
	_STARBURN_STARWAVE2_INIT_PARAMS (see page 457)	Structure that represents initialization parameters for audio conversion
	_STARBURN_TRACK_INFORMATION (see page 458)	Structure that represents Track information
	_STARBURN_TRACK_INFORMATION_EX (see page 459)	Structure that represents Track information extended
	_STARBURN_UDF2_DIRECTORY_INFO (see page 461)	UDF directory information
	_STARBURN_UDF2_FILE_DATE_TIME (see page 461)	Structure that represents UDF2 file date and time
	_STARBURN_UDF2_FILE_INFO (see page 463)	UDF file information
	_STARPORT_DEVICE_LIST (see page 463)	Structure that represents StarPort device list
	_STARPORT_DEVICE_LIST_ENTRY (see page 464)	Structure that represents StarPort device list entry

	_STARWAVE2_COMPRESSION_PROFILE (see page 466)	
	_TOC_ENTRY (see page 466)	Structure that represents TOC entry
	_TOC_INFORMATION (see page 468)	Structure that represents TOC information
	_UDF_CONTROL_BLOCK (see page 469)	Structure that represents UDF control block
	_UDF_FILE_EXTENT (see page 469)	Structure that describes a single file extent
	_UDF_FILE_HANDLE (see page 470)	Structure that represents UDF file handle
	_UDF_FILE_LOOKUP_ENTRY (see page 470)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function
	_UDF_LOOKUP_DIR_ENTRY (see page 471)	Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	_UDF_LOOKUP_FILE_ENTRY (see page 471)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	_UDF_TREE_ITEM (see page 472)	Structure that represents UDF tree item
	_WAVE_FILE_HEADER (see page 474)	Structure that represents WAVE file header
	_WAVE_FORMAT_CHUNK (see page 475)	Structure that represents WAVE format chunk
	CDB_FAILURE_INFORMATION (see page 480)	Structure that represents CDB failure information
	DAO_DISC_LAYOUT (see page 480)	Structure that represents DAO disc layout
	DAO_DISC_LAYOUT_ENTRY (see page 481)	Structure that represents DAO disc layout entry (track)
	DISC_FILESYSTEM (see page 482)	Structure with disc file system flags
	DISC_LAYOUT (see page 482)	Structure that represents disc layout
	DISC_LAYOUT_ENTRY (see page 483)	Structure that represents disc layout entry (track)
	DVD_VIDEO_CONTROL_BLOCK (see page 486)	Structure that represents DVD-Video control block and pointer to DVD-Video control block
	FULL_TOC_ENTRY_RAW (see page 491)	Structure that represents full TOC entry
	ISO9660_DATE_TIME (see page 492)	Structure that represents ISO9660 date and time
	NAME_COLLISION_INFO (see page 493)	Structure that represents information about name collision
	PCDB_FAILURE_INFORMATION (see page 498)	Structure that represents CDB failure information
	PDAO_DISC_LAYOUT (see page 499)	Structure that represents DAO disc layout
	PDAO_DISC_LAYOUT_ENTRY (see page 499)	Structure that represents DAO disc layout entry (track)
	PDISC_FILESYSTEM (see page 500)	Structure with disc file system flags
	PDISC_LAYOUT (see page 500)	Structure that represents disc layout
	PDISC_LAYOUT_ENTRY (see page 501)	Structure that represents disc layout entry (track)
	PDVD_VIDEO_CONTROL_BLOCK (see page 504)	Structure that represents DVD-Video control block and pointer to DVD-Video control block
	PFULL_TOC_ENTRY_RAW (see page 509)	Structure that represents full TOC entry
	PISO9660_DATE_TIME (see page 510)	Structure that represents ISO9660 date and time
	PNAME_COLLISION_INFO (see page 511)	Structure that represents information about name collision
	PPQ_SUBCHANNEL (see page 513)	Structure that represents formatted PQ sub-channel
	PPSTARBURN_BDRE_FORMAT_PROFILE (see page 514)	Structure that represents format profile for BD-RE media
	PPSTARPORT_DEVICE_LIST (see page 517)	Structure that represents StarPort device list
	PPSTARPORT_DEVICE_LIST_ENTRY (see page 517)	Structure that represents StarPort device list entry
	PPSTARWAVE2_COMPRESSION_PROFILE (see page 519)	
	PQ_SUBCHANNEL (see page 520)	Structure that represents formatted PQ sub-channel
	PSCSI_DEVICE_ADDRESS (see page 522)	Structure that represents SCSI device address
	PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS (see page 523)	Structure that represents advanced supported media formats
	PSTARBURN_BDRE_FORMAT_PROFILE (see page 525)	Structure that represents format profile for BD-RE media
	PSTARBURN_DISC_ATIP_INFORMATION (see page 527)	Structure that represents Disc ATIP information
	PSTARBURN_DISC_INFORMATION (see page 528)	Structure that represents Disc information
	PSTARBURN_STARWAVE2_INIT_PARAMS (see page 532)	Structure that represents initialization parameters for audio conversion
	PSTARBURN_TRACK_INFORMATION (see page 533)	Structure that represents Track information
	PSTARBURN_TRACK_INFORMATION_EX (see page 534)	Structure that represents Track information extended
	PSTARPORT_DEVICE_LIST (see page 536)	Structure that represents StarPort device list
	PSTARPORT_DEVICE_LIST_ENTRY (see page 536)	Structure that represents StarPort device list entry
	PSTARWAVE2_COMPRESSION_PROFILE (see page 539)	
	PTOC_ENTRY (see page 539)	Structure that represents TOC entry
	PTOC_INFORMATION (see page 541)	Structure that represents TOC information
	PUDF_CONTROL_BLOCK (see page 542)	Structure that represents UDF control block

	PUDF_FILE_EXTENT (see page 542)	Structure that describes a single file extent
	PUDF_FILE_HANDLE (see page 543)	Structure that represents UDF file handle
	PUDF_FILE_LOOKUP_ENTRY (see page 543)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function
	PUDF_LOOKUP_DIR_ENTRY (see page 544)	Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	PUDF_LOOKUP_FILE_ENTRY (see page 544)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	PUDF_TREE_ITEM (see page 545)	Structure that represents UDF tree item
	PWAVE_FILE_HEADER (see page 546)	Structure that represents WAVE file header
	PWAVE_FORMAT_CHUNK (see page 547)	Structure that represents WAVE format chunk
	SCSI_DEVICE_ADDRESS (see page 549)	Structure that represents SCSI device address
	STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS (see page 550)	Structure that represents advanced supported media formats
	STARBURN_BDRE_FORMAT_PROFILE (see page 553)	Structure that represents format profile for BD-RE media
	STARBURN_DISC_ATIP_INFORMATION (see page 554)	Structure that represents Disc ATIP information
	STARBURN_DISC_INFORMATION (see page 555)	Structure that represents Disc information
	STARBURN_ISO9660_DIRECTORY_INFO (see page 558)	This is type STARBURN_ISO9660_DIRECTORY_INFO.
	STARBURN_ISO9660_FILE_INFO (see page 558)	ISO9660 file information
	STARBURN_STARWAVE2_INIT_PARAMS (see page 562)	Structure that represents initialization parameters for audio conversion
	STARBURN_TRACK_INFORMATION (see page 563)	Structure that represents Track information
	STARBURN_TRACK_INFORMATION_EX (see page 564)	Structure that represents Track information extended
	STARBURN_UDF2_DIRECTORY_INFO (see page 566)	UDF directory information
	STARBURN_UDF2_FILE_DATE_TIME (see page 566)	Structure that represents UDF2 file date and time
	STARBURN_UDF2_FILE_INFO (see page 567)	UDF file information
	STARPORT_DEVICE_LIST (see page 568)	Structure that represents StarPort device list
	STARPORT_DEVICE_LIST_ENTRY (see page 568)	Structure that represents StarPort device list entry
	STARWAVE2_COMPRESSION_PROFILE (see page 571)	
	TOC_ENTRY (see page 571)	Structure that represents TOC entry
	TOC_INFORMATION (see page 573)	Structure that represents TOC information
	UDF_CONTROL_BLOCK (see page 574)	Structure that represents UDF control block
	UDF_FILE_EXTENT (see page 574)	Structure that describes a single file extent
	UDF_FILE_HANDLE (see page 575)	Structure that represents UDF file handle
	UDF_FILE_LOOKUP_ENTRY (see page 575)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function
	UDF_LOOKUP_DIR_ENTRY (see page 576)	Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	UDF_LOOKUP_FILE_ENTRY (see page 576)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	UDF_TREE_ITEM (see page 577)	Structure that represents UDF tree item
	WAVE_FILE_HEADER (see page 579)	Structure that represents WAVE file header
	WAVE_FORMAT_CHUNK (see page 580)	Structure that represents WAVE format chunk

2.2.1 _CALLBACK_NUMBER Enumeration

C++

```
enum _CALLBACK_NUMBER {
  CN_FILE_TREE_PROGRESS_ADD = 0,
  CN_FILE_TREE_PROGRESS_REMOVE,
  CN_FILE_TREE_PROGRESS_IGNORE,
  CN_FILE_TREE_PROGRESS_NAME_COLLISION,
  CN_TARGET_FILE_ANALYZE_BEGIN,
  CN_TARGET_FILE_ANALYZE_END,
  CN_WAIT_CACHE_FULL_BEGIN,
  CN_WAIT_CACHE_FULL_END,
```

```
    CN_SYNCHRONIZE_CACHE_BEGIN,
    CN_SYNCHRONIZE_CACHE_END,
    CN_FIND_DEVICE,
    CN_CDVD_READ_PROGRESS,
    CN_CDVD_WRITE_PROGRESS,
    CN_CDVD_BUFFER_STATUS,
    CN_CDVD_TRACK_BEGIN,
    CN_CDVD_TRACK_END,
    CN_CDVD_SPLIT_BEGIN,
    CN_CDVD_SPLIT_END,
    CN_CDVD_READ_BAD_BLOCK_HIT,
    CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT,
    CN_CDVD_READ_RETRY,
    CN_DVDPLUSRW_FORMAT_BEGIN,
    CN_DVDPLUSRW_FORMAT_END,
    CN_DVDRAM_FORMAT_BEGIN,
    CN_DVDRAM_FORMAT_END,
    CN_BUFFER_UNDERRUN,
    CN_DVD_MEDIA_PADDING_SIZE,
    CN_DVD_MEDIA_PADDING_BEGIN,
    CN_DVD_MEDIA_PADDING_END,
    CN_CDVD_READ_CANCEL_QUERY,
    CN_DVDRW_QUICK_FORMAT_BEGIN,
    CN_DVDRW_QUICK_FORMAT_END,
    CN_DVD_TEST_WRITE_DISABLED,
    CN_CDVD_DPM_BEGIN,
    CN_CDVD_DPM_END,
    CN_CDVD_DPM_PROGRESS,
    CN_CDVD_VERIFY_PROGRESS,
    CN_SAO_TRACK_WRITE_BEGIN,
    CN_SAO_TRACK_WRITE_END,
    CN_DVD_MEDIA_PADDING_WRITE_PROGRESS,
    CN_CDVD_WRITE_BEGIN,
    CN_CDVD_WRITE_END,
    CN_CDVD_VERIFY_BEGIN,
    CN_CDVD_VERIFY_END,
    CN_BDRE_FORMAT_BEGIN,
    CN_BDRE_FORMAT_END,
    CN_UDF_FILE_LOOKUP,
    CN_UDF_LOOKUP_FILE,
    CN_UDF_LOOKUP_DIR,
    CN_CDVD_BLANKAREA_PROGRESS
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
CN_FILE_TREE_PROGRESS_ADD = 0	Item was add to the file tree
CN_FILE_TREE_PROGRESS_REMOVE	Item was removed from the file tree
CN_FILE_TREE_PROGRESS_IGNORE	Item was ignored during processing file tree
CN_FILE_TREE_PROGRESS_NAME_COLLISION	There are two nodes which have the same name
CN_TARGET_FILE_ANALYZE_BEGIN	File internal structure (size, type etc) analyze started
CN_TARGET_FILE_ANALYZE_END	File structure analyze completed
CN_WAIT_CACHE_FULL_BEGIN	Toolkit started to wait for cache to become full
CN_WAIT_CACHE_FULL_END	Toolkit finished to wait for cache fullness
CN_SYNCHRONIZE_CACHE_BEGIN	Cache flushing to the media started
CN_SYNCHRONIZE_CACHE_END	Cache flushing completed
CN_FIND_DEVICE	Find device operation completed
CN_CDVD_READ_PROGRESS	CD/DVD/Blu-Ray/HD-DVD read operation progress
CN_CDVD_WRITE_PROGRESS	CD/DVD/Blu-Ray/HD-DVD write operation progress
CN_CDVD_BUFFER_STATUS	CD/DVD/Blu-Ray/HD-DVD buffer status information queried
CN_CDVD_TRACK_BEGIN	CD/DVD/Blu-Ray/HD-DVD track processing started
CN_CDVD_TRACK_END	CD/DVD/Blu-Ray/HD-DVD track processing completed
CN_CDVD_SPLIT_BEGIN	CD/DVD/Blu-Ray/HD-DVD split section processing started
CN_CDVD_SPLIT_END	CD/DVD/Blu-Ray/HD-DVD split section processing completed

CN_CDVD_READ_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a bad (unrecoverable) block
CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a ECC/EDC bad (recoverable) block
CN_CDVD_READ_RETRY	CD/DVD/Blu-Ray/HD-DVD read operation was retried
CN_DVDPLUSRW_FORMAT_BEGIN	DVD+RW format operation started
CN_DVDPLUSRW_FORMAT_END	DVD+RW format operation completed
CN_DVDRAM_FORMAT_BEGIN	DVD-RAM format operation started
CN_DVDRAM_FORMAT_END	DVD-RAM format operation completed
CN_BUFFER_UNDERRUN	Buffer underrun condition happened
CN_DVD_MEDIA_PADDING_SIZE	DVD media would be padded to 1GB size, additional info passed
CN_DVD_MEDIA_PADDING_BEGIN	DVD media padding burn process started
CN_DVD_MEDIA_PADDING_END	DVD media padding burn process completed
CN_CDVD_READ_CANCEL_QUERY	CD/DVD/Blu-Ray/HD-DVD media processing plug-in queries cancel status
CN_DVDRW_QUICK_FORMAT_BEGIN	DVD-RW quick format operation started
CN_DVDRW_QUICK_FORMAT_END	DVD-RW quick format operation completed
CN_DVD_TEST_WRITE_DISABLED	Test write is disabled (for DVD+R/RW, DVD-RAM or multisession DVD-RW)
CN_CDVD_DPM_BEGIN	CD/DVD/Blu-Ray/HD-DVD DPM processing started
CN_CDVD_DPM_END	CD/DVD/Blu-Ray/HD-DVD DPM processing completed
CN_CDVD_DPM_PROGRESS	CD/DVD/Blu-Ray/HD-DVD DPM processing progress
CN_CDVD_VERIFY_PROGRESS	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_SAO_TRACK_WRITE_BEGIN	SAO track write started
CN_SAO_TRACK_WRITE_END	SAO track write completed
CN_DVD_MEDIA_PADDING_WRITE_PROGRESS	DVD media padding burn progress indication
CN_CDVD_WRITE_BEGIN	CD/DVD/Blu-Ray/HD-DVD write operation started
CN_CDVD_WRITE_END	CD/DVD/Blu-Ray/HD-DVD write operation completed
CN_CDVD_VERIFY_BEGIN	CD/DVD/Blu-Ray/HD-DVD verify operation started
CN_CDVD_VERIFY_END	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_BDRE_FORMAT_BEGIN	BD-RE format operation started
CN_BDRE_FORMAT_END	BD-RE format operation completed
CN_UDF_FILE_LOOKUP	UDF file found during fast lookup
CN_UDF_LOOKUP_FILE	UDF file found during fast lookup
CN_UDF_LOOKUP_DIR	UDF directory found during fast lookup
CN_CDVD_BLANKAREA_PROGRESS	CD/DVD/Blu-Ray/HD-DVD blank area operation progress

Description

Enum that represents callback number

Member	Definition
CN_FILE_TREE_PROGRESS_ADD	Item was add to the file tree
CN_FILE_TREE_PROGRESS_REMOVE	Item was removed from the file tree
CN_FILE_TREE_PROGRESS_IGNORE	Item was ignored during processing file tree
CN_FILE_TREE_PROGRESS_NAME...	There are two nodes which have the same name
CN_TARGET_FILE_ANALYZE_BEGIN	File internal structure (size, type etc) analyze started
CN_TARGET_FILE_ANALYZE_END	File structure analyze completed
CN_WAIT_CACHE_FULL_BEGIN	Toolkit started to wait for cache to become full
CN_WAIT_CACHE_FULL_END	Toolkit finished to wait for cache fullness
CN_SYNCHRONIZE_CACHE_BEGIN	Cache flushing to the media started
CN_SYNCHRONIZE_CACHE_END	Cache flushing to the media completed
CN_FIND_DEVICE	Find device operation completed
CD_CDVD_READ_PROGRESS	CD/DVD/Blu-Ray/HD-DVD read operation progress
CN_CDVD_WRITE_PROGRESS	CD/DVD/Blu-Ray/HD-DVD write operation progress
CN_CDVD_BUFFER_STATUS	CD/DVD/Blu-Ray/HD-DVD buffer status information queried

CN_CDVD_TRACK_BEGIN	CD/DVD/Blu-Ray/HD-DVD track processing started
CN_CDVD_TRACK_END	CD/DVD/Blu-Ray/HD-DVD track processing completed
CN_CDVD_SPLIT_BEGIN	CD/DVD/Blu-Ray/HD-DVD split section processing started
CN_CDVD_SPLIT_END	CD/DVD/Blu-Ray/HD-DVD split section processing completed
CN_CDVD_READ_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a bad (unrecoverable) block
CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a ECC/EDC bad (recoverable) block
CN_CDVD_READ_RETRY	CD/DVD/Blu-Ray/HD-DVD read operation was retried
CN_DVDPLUSRW_FORMAT_BEGIN	DVD+RW format operation started
CN_DVDPLUSRW_FORMAT_EDN	DVD+RW format operation completed
CN_DVDRAM_FORMAT_BEGIN	DVD-RAM format operation started
CN_DVDRAM_FORMAT_EDN	DVD-RAM format operation completed
CN_BUFFER_UNDERRUN	Buffer underrun condition happened
CN_DVD_MEDIA_PADDING_SIZE	DVD media would be padded to 1GB size, additional info passed
CN_DVD_MEDIA_PADDING_BEGIN	DVD media padding burn process started
CN_DVD_MEDIA_PADDING_END	DVD media padding burn process completed
CN_CDVD_READ_CANCEL_QUERY	CD/DVD/Blu-Ray/HD-DVD media processing plug-in queries cancel status
CN_DVDRW_QUICK_FORMAT_BEGIN	DVD-RW quick format operation started
CN_DVDRW_QUICK_FORMAT_END	DVD-RW quick format operation completed
CN_DVD_TEST_WRITE_DISABLED	Test write is disabled (for DVD+R/RW, DVD-RAM or multisession DVD-RW)
CN_CDVD_DPM_BEGIN	CD/DVD/Blu-Ray/HD-DVD DPM processing started
CN_CDVD_DPM_END	CD/DVD/Blu-Ray/HD-DVD DPM processing completed
CD_CDVD_DPM_PROGRESS	CD/DVD/Blu-Ray/HD-DVD DPM operation progress
CN_CDVD_VERIFY_PROGRESS	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_SAO_TRACK_WRITE_BEGIN	SAO track write started
CN_SAO_TRACK_WRITE_END	SAO track write completed
CN_DVD_MEDIA_PADDING_WRITE_PRO...	DVD media padding burn progress indication
CN_CDVD_WRITE_BEGIN	CD/DVD/Blu-Ray/HD-DVD write operation started
CN_CDVD_WRITE_END	CD/DVD/Blu-Ray/HD-DVD write operation completed
CN_CDVD_VERIFY_BEGIN	CD/DVD/Blu-Ray/HD-DVD verify operation started
CN_CDVD_VERIFY_END	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_BDRE_FORMAT_BEGIN	BD-RE format operation started
CN_BDRE_FORMAT_END	BD-RE format operation completed
CN_UDF_FILE_LOOKUP	UDF file found during fast lookup
CN_UDF_LOOKUP_FILE	UDF file found during fast lookup
CN_UDF_LOOKUP_DIR	UDF directory found during fast lookup

2.2.2 _CDB_FAILURE_INFORMATION Structure

C++

```
struct _CDB_FAILURE_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__CDBSizeInUCHARs;
  UCHAR m__UCHAR__CDB[ 16 ];
  UCHAR m__UCHAR__SenseSizeInUCHARs;
  UCHAR m__UCHAR__Sense[ 32 ];
  UCHAR m__UCHAR__TransportStatus;
  UCHAR m__UCHAR__TargetStatus;
  UCHAR m__UCHAR__HostAdapterStatus;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__CDBSizeInUCHARs;`	CDB size in UCHARs
`UCHAR m__UCHAR__CDB[ 16 ];`	CDB dump
`UCHAR m__UCHAR__SenseSizeInUCHARs;`	SCSI sense size in UCHARs
`UCHAR m__UCHAR__Sense[ 32 ];`	SCSI sense dump
`UCHAR m__UCHAR__TransportStatus;`	SCSI transport status
`UCHAR m__UCHAR__TargetStatus;`	SCSI target status
`UCHAR m__UCHAR__HostAdapterStatus;`	SCSI host adapter status

Description

Structure that represents CDB failure information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__CDBSizeInUCHARs	CDB size in UCHARs
m__UCHAR__CDB[16]	CDB dump
m__UCHAR__SenseSizeInUCHARs	SCSI sense size in UCHARs
m__UCHAR__Sense[32]	SCSI sense dump
m__UCHAR__TransportStatus	SCSI transport status
m__UCHAR__TargetStatus	SCSI target status
m__UCHAR__HostAdapterStatus	SCSI host adapter status

2.2.3 _DAO_DISC_LAYOUT Structure

C++

```
struct _DAO_DISC_LAYOUT {
  LONG m__LONG__NumberOfEntries;
  DAO_DISC_LAYOUT_ENTRY m__DAO_DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of entries
`DAO_DISC_LAYOUT_ENTRY m__DAO_DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];`	DAO disc layout entries

Description

Structure that represents DAO disc layout

Member	Definition
m__LONG__NumberOfEntries	Number of entries
m__DAO_DISC_LAYOUT_ENTRY	DAO disc layout entries

2.2.4 _DAO_DISC_LAYOUT_ENTRY Structure

C++

```
struct _DAO_DISC_LAYOUT_ENTRY {
  LONG m__LONG__TrackSizeInLBs;
  LONG m__LONG__TrackStartingLBA;
  BOOLEAN m__BOOLEAN__IsUnicode;
  union {
    CHAR m__CHAR__TrackName[ (MAX_PATH * sizeof(WCHAR) ) ];
    WCHAR m__WCHAR__TrackName[ (MAX_PATH) ];
  };
  PVOID m__PVOID__File;
  BOOLEAN m__BOOLEAN__IsDataTrack;
  BOOLEAN m__BOOLEAN__IsRawTrack;
  BOOLEAN m__BOOLEAN__IsAudioExTrack;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__TrackSizeInLBs;`	Track size in LBs (logical blocks)
`LONG m__LONG__TrackStartingLBA;`	Track starting LBA (logical block address)
`BOOLEAN m__BOOLEAN__IsUnicode;`	Is name in Unicode format or not (see below)
`CHAR m__CHAR__TrackName[ (MAX_PATH * sizeof(WCHAR) ) ];`	Track name (absolute path & name) in ANSI format (see above)
`WCHAR m__WCHAR__TrackName[ (MAX_PATH) ];`	Track name (absolute path & name) in Unicode format (see above)
`PVOID m__PVOID__File;`	Pointer to internally created disk file object (NULL initially)
`BOOLEAN m__BOOLEAN__IsDataTrack;`	Is this data track (audio otherwise)
`BOOLEAN m__BOOLEAN__IsRawTrack;`	Is this raw track (MDF image)
`BOOLEAN m__BOOLEAN__IsAudioExTrack;`	Is this extended audio track (2448 UCHARs/logical block)

Description

Structure that represents DAO disc layout entry (track)

Member	Definition
m__LONG__TrackSizeInLBs	Track size in LBs (logical blocks)

m__LONG__TrackStartingLBA	Track starting LBA (logical block address)
m__BOOLEAN__IsUnicode	Is name in Unicode format or not (see below)
m__CHAR__TrackName	Track name (absolute path & name) in ANSI or Unicode format (see above)
m__PVOID__File	Pointer to internally created disk file object (NULL initially)
m__BOOLEAN__IsDataTrack	Is this data track (audio otherwise)
m__BOOLEAN__IsRawTrack	Is this raw track (MDF image)
m__BOOLEAN__IsAudioExTrack	Is this extended audio track (2448 UCHARs/logical block)

2.2.5 _DISC_FILESYSTEM Structure

C++

```
struct _DISC_FILESYSTEM {
  BOOLEAN m__BOOLEAN__UDFPresent;
  BOOLEAN m__BOOLEAN__JolietPresent;
  BOOLEAN m__BOOLEAN__ISO9660Present;
  BOOLEAN m__BOOLEAN__ElToritoBootRecordPresent;
};
```

File

StarBurn.h (see page 662)

Description

Structure with disc file system flags

Member Definition
--------- ----------
m__BOOLEAN__UDFPresent UDF file system is present
m__BOOLEAN__ISO9660Present ISO9660 file system is present
m__BOOLEAN__JolietPresent Joliet extension for ISO9660 file system is present
m__BOOLEAN__ElToritoBootRecordPresent El Torito boot record is present

2.2.6 _DISC_LAYOUT Structure

C++

```
struct _DISC_LAYOUT {
  LONG m__LONG__NumberOfEntries;
  DISC_LAYOUT_ENTRY m__DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];
  LONG m__LONG__NumberOfRawRawPWTracks;
  LONG m__LONG__RawRawPWTrackSizeInLBs[ NUMBER_OF_TRACKS ];
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of entries
`DISC_LAYOUT_ENTRY m__DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];`	Disc layout entries
`LONG m__LONG__NumberOfRawRawPWTracks;`	Number of raw + raw P-W sub-channel tracks
`LONG m__LONG__RawRawPWTrackSizeInLBs[ NUMBER_OF_TRACKS ];`	Raw + raw P-W sub-channel track sizes

Description

Structure that represents disc layout

Member	Definition
m__LONG__NumberOfEntries	Number of entries
m__DISC_LAYOUT_ENTRY	Disc layout entries
m__LONG__NumberOfRawRawPWTracks	Number of raw + raw P-W sub-channel tracks
m__LONG__RawRawPWTrackSize	Raw + raw P-W sub-channel track sizes

2.2.7 _DISC_LAYOUT_ENTRY Structure

C++

```
struct _DISC_LAYOUT_ENTRY {
  BOOLEAN m__BOOLEAN__IsAudio;
  BOOLEAN m__BOOLEAN__IsVideo;
  BOOLEAN m__BOOLEAN__IsRawRawPW;
  LONG m__LONG__TrackSizeInLBs;
  LONG m__LONG__PreGapSizeInLBs;
  LONG m__LONG__PostGapSizeInLBs;
  union {
    CHAR m__CHAR__TrackName[ MAX_PATH*sizeof(WCHAR) ];
    WCHAR m__WCHAR__TrackName[ MAX_PATH ];
  };
  PVOID m__PVOID__File;
  PVOID m__PVOID__Tree;
  PVOID m__PVOID__BackSideTree;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsAudio;`	Is this audio track
`BOOLEAN m__BOOLEAN__IsVideo;`	Is this video track
`BOOLEAN m__BOOLEAN__IsRawRawPW;`	Is this raw + raw P-W sub-channel track
`LONG m__LONG__TrackSizeInLBs;`	Track size in LBs (logical blocks)
`LONG m__LONG__PreGapSizeInLBs;`	PreGap size in LBs (logical blocks)
`LONG m__LONG__PostGapSizeInLBs;`	PostGap size in LBs (logical blocks)
`CHAR m__CHAR__TrackName[ MAX_PATH*sizeof(WCHAR) ];`	Track name (ansi)(absolute path & name), can be zero array if next pointer is valid
`WCHAR m__WCHAR__TrackName[ MAX_PATH ];`	Track name (unicode)(absolute path & name), can be zero array if next pointer is valid
`PVOID m__PVOID__File;`	Pointer to internally created disk file object
`PVOID m__PVOID__Tree;`	Pointer to object created with StarBurn_ISO9660JolietFileTree_Create (see page 288)(), can be NULL
`PVOID m__PVOID__BackSideTree;`	Pointer to undecorated ISO9660 file tree object

Description

Structure that represents disc layout entry (track)

Member	Definition
m__BOOLEAN__IsAudio	Is this audio track
m__BOOLEAN__IsVideo	Is this video track
m__BOOLEAN__IsRawRawPW	Is this raw + raw P-W sub-channel track
m__LONG__TrackSizeInLBs	Track size in LBs (logical blocks)
m__LONG__PreGapSizeInLBs	PreGap size in LBs (logical blocks)
m__LONG__PostGapSizeInLBs	PostGap size in LBs (logical blocks)
m__CHAR__TrackName	Track name (absolute path & name), can be zero array if next pointer is valid
m__PVOID__File	Pointer to internally created disk file object
m__PVOID__Tree	Pointer to object created with StarBurn_ISO9660JolietFileTree_Create (see page 288)(), can be NULL
m__PVOID__BackSideTree	Pointer to undecorated ISO9660 file tree object

2.2.8 _DISC_TYPE Enumeration

C++

```
enum _DISC_TYPE {
  DISC_TYPE_UNKNOWN = 0,
  DISC_TYPE_CDROM = 1,
  DISC_TYPE_CDR = 2,
  DISC_TYPE_CDRW = 3,
  DISC_TYPE_DVDROM = 4,
  DISC_TYPE_DVDR = 5,
  DISC_TYPE_DVDRAM = 6,
  DISC_TYPE_DVDRWRO = 7,
  DISC_TYPE_DVDRWSR = 8,
  DISC_TYPE_DVDPLUSRW = 9,
  DISC_TYPE_DDCDROM = 10,
  DISC_TYPE_DDCDR = 11,
  DISC_TYPE_DDCDRW = 12,
  DISC_TYPE_DVDPLUSR = 13,
  DISC_TYPE_NO_MEDIA = 14,
  DISC_TYPE_DVDPLUSR_DL = 15,
  DISC_TYPE_DVDR_DL = 16,
  DISC_TYPE_BDROM = 17,
  DISC_TYPE_BDR = 18,
  DISC_TYPE_BDRE = 19,
  DISC_TYPE_HDDVDROM = 20,
  DISC_TYPE_HDDVDR = 21,
  DISC_TYPE_HDDVDRW = 22
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
DISC_TYPE_UNKNOWN = 0	Unknown disc type
DISC_TYPE_CDROM = 1	CD-ROM
DISC_TYPE_CDR = 2	CD-R

DISC_TYPE_CDRW = 3	CD-RW
DISC_TYPE_DVDROM = 4	DVD-ROM
DISC_TYPE_DVDR = 5	DVD-R
DISC_TYPE_DVDRAM = 6	DVD-RAM
DISC_TYPE_DVDRWRO = 7	DVD-RW RO (Restricted Overwrite)
DISC_TYPE_DVDRWSR = 8	DVD-RW SR (Sequential Recording)
DISC_TYPE_DVDPLUSRW = 9	DVD+RW
DISC_TYPE_DDCDROM = 10	DD (Double Density) CD-ROM
DISC_TYPE_DDCDR = 11	DD (Double Density) CD-R
DISC_TYPE_DDCDRW = 12	DD (Double Density) CD-RW
DISC_TYPE_DVDPLUSR = 13	DVD+R
DISC_TYPE_NO_MEDIA = 14	No media is inserted to the disc drive
DISC_TYPE_DVDPLUSR_DL = 15	DVD+R DL (Double Layer)
DISC_TYPE_DVDR_DL = 16	DVD-R DL (Dual Layer)
DISC_TYPE_BDROM = 17	BD-ROM (Blu-Ray ROM)
DISC_TYPE_BDR = 18	BD-R (Blu-Ray Disc Recordable)
DISC_TYPE_BDRE = 19	BD-RE (Blu-Ray Disc Recordable Erasable)
DISC_TYPE_HDDVDROM = 20	HD-DVD ROM
DISC_TYPE_HDDVDR = 21	HD-DVD Recordable
DISC_TYPE_HDDVDRW = 22	HD-DVD ReWritable

Description

Enum that represents inserted disc type

Member	Definition
DISC_TYPE_UNKNOWN	Unknown disc type
DISC_TYPE_CDROM,	CD-ROM
DISC_TYPE_CDR,	CD-R
DISC_TYPE_CDRW,	CD-RW
DISC_TYPE_DVDROM,	DVD-ROM
DISC_TYPE_DVDR,	DVD-R
DISC_TYPE_DVDRAM,	DVD-RAM
DISC_TYPE_DVDRWRO,	DVD-RW RO (Restricted Overwrite)
DISC_TYPE_DVDRWSR,	DVD-RW SR (Sequential Recording)
DISC_TYPE_DVDPLUSRW,	DVD+RW
DISC_TYPE_DDCDROM,	DD (Double Density) CD-ROM
DISC_TYPE_DDCDR,	DD (Double Density) CD-R
DISC_TYPE_DDCRW	DD (Double Density) CD-RW
DISC_TYPE_DVDPLUSR	DVD+R
DISC_TYPE_NO_MEDIA	No media is inserted to the disc drive
DISC_TYPE_DVDPLUSR_DL	DVD+R DL (Double Layer)
DISC_TYPE_DVDR_DL	DVD-R DL (Dual Layer)
DISC_TYPE_BDROM	BD-ROM (Blu-Ray ROM)
DISC_TYPE_BDR	BD-R (Blu-Ray Disc Recordable)
DISC_TYPE_BDRE	BD-RE (Blu-Ray Disc Recordable Erasable)
DISC_TYPE_HDDVDROM	HD-DVD ROM
DISC_TYPE_HDDVDR	HD-DVD Recordable
DISC_TYPE_HDDVDRW	HD-DVD ReWritable

2.2.9 _DVD_VIDEO_CONTROL_BLOCK Structure

C++

```
struct _DVD_VIDEO_CONTROL_BLOCK {
  UDF_CONTROL_BLOCK m__UDF_CONTROL_BLOCK;
  UDF_TREE_ITEM m__UDF_TREE_ITEM__Directory[ 4 ];
  UDF_TREE_ITEM m__UDF_TREE_ITEM__File[ 1192 ];
  LONG m__LONG__NumberOfFiles;
  LONG m__LONG__NumberOfTitles;
  LONG m__LONG__VtsIndex[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ][ 2 ];
  BOOLEAN m__BOOLEAN__IsMainMenuVob;
  BOOLEAN m__BOOLEAN__IsTitleMenuVob[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ];
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UDF_CONTROL_BLOCK m__UDF_CONTROL_BLOCK;`	Main control block for ISO9660/UDF bridge file system
`UDF_TREE_ITEM m__UDF_TREE_ITEM__Directory[ 4 ];`	1st element of the array is not used (reserved), 2nd goes for ROOT, 3d for AUDIO_TS and 4th for VIDEO_TS directory 1 (reserved) + 1 (ROOT) + 1 (AUDIO_TS) + 1 (VIDEO_TS) = 4
`UDF_TREE_ITEM m__UDF_TREE_ITEM__File[ 1192 ];`	1st element of the array is not used (reserved), then goes 3 entries for VIDEO_TS.IFO, VIDEO_TS.VOB and VIDEO_TS.BUP. Then goes up to 99 titles each can contain up to 10 chapters and IFO and BUP files 1 (reserved) + 3 (VIDEO_TS.IFO, VIDEO_TS.VOB and VIDEO_TS.BUP) + 99 * 12 (VTS_xx_0.IFO, VTS_xx_0.VOB .. VTS_xx_9.VOB + VTS_xx_0.BUP) = 1 + 3 + 1188 = 1192
`LONG m__LONG__NumberOfFiles;`	Number of files really used in m__UDF_TREE_ITEM__File[ ] array
`LONG m__LONG__NumberOfTitles;`	Number of titles in DVD-Video compilation
`LONG m__LONG__VtsIndex[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ][ 2 ];`	Indexes for VTS_xx_0.IFO and VTS_xx_0.BUP for every title
`BOOLEAN m__BOOLEAN__IsMainMenuVob;`	Is VIDEO_TS.VOB file present (does whole movie has a main menu)
`BOOLEAN m__BOOLEAN__IsTitleMenuVob[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ];`	Is VTS_xx_0.VOB file present (does title set has a menu)

Description

Structure that represents DVD-Video control block and pointer to DVD-Video control block

2.2.10 _ERASE_TYPE Enumeration

C++

```
enum _ERASE_TYPE {
  ERASE_TYPE_BLANK_DISC_FULL = 0,
  ERASE_TYPE_BLANK_DISC_FAST,
  ERASE_TYPE_BLANK_TRACK,
  ERASE_TYPE_UNRESERVE_TRACK,
  ERASE_TYPE_BLANK_TRACK_TAIL,
  ERASE_TYPE_UNCLOSE_LAST_SESSION,
  ERASE_TYPE_BLANK_SESSION
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
ERASE_TYPE_BLANK_DISC_FULL = 0	Completely erase the rewritable disc
ERASE_TYPE_BLANK_DISC_FAST	Erase only TOC, PMA and first track lead in on the rewritable disc
ERASE_TYPE_BLANK_TRACK	Erase track only
ERASE_TYPE_UNRESERVE_TRACK	Unreserve track that was reserved before
ERASE_TYPE_BLANK_TRACK_TAIL	Erase track tail
ERASE_TYPE_UNCLOSE_LAST_SESSION	Unclose last closed session
ERASE_TYPE_BLANK_SESSION	Erase last session

Description

Enum that represents erase type

Member	Definition
ERASE_TYPE_BLANK_DISC_FULL	Completely erase the rewritable disc
ERASE_TYPE_BLANK_DISC_FAST	Erase only TOC, PMA and first track lead in on the rewritable disc
ERASE_TYPE_BLANK_TRACK	Erase track only
ERASE_TYPE_UNRESERVE_TRACK	Unreserve track that was reserved before
ERASE_TYPE_BLANK_TRACK_TAIL	Erase track tail
ERASE_TYPE_UNCLOSE_LAST_SESSION	Unclose last closed session
ERASE_TYPE_BLANK_SESSION	Erase last session

2.2.11 _EXCEPTION_NUMBER Enumeration

C++

```
enum _EXCEPTION_NUMBER {
  EN_SUCCESS = 0,
  EN_FAILURE,
  EN_INVALID_INPUT_PARAMETER,
  EN_INVALID_STATE,
  EN_MEMORY_ALLOCATION_FAILED,
  EN_SYSTEM_CALL_FAILED,
  EN_SCSI_TRANSPORT_FAILED,
  EN_SCSI_DEVICE_BUSY,
  EN_SCSI_CDB_FAILED,
  EN_SCSI_DEVICE_INVALID_TYPE,
  EN_INVALID_RESPONSE,
  EN_BUFFER_UNDERRUN,
  EN_INVALID_EXCEPTION,
  EN_ACCESS_TO_FEATURE_DENIED,
  EN_USER_EXCEPTION,
  EN_PATH_TOO_LONG,
  EN_UNDER_CONSTRUCTION,
  EN_NOT_FOUND,
  EN_FILE_TOO_BIG,
  EN_NOT_IMPLEMENTED,
  EN_RANGE,
  EN_REGISTRATION_FAILED,
  EN_UNSUPPORTED_AUDIO,
  EN_BUFFER_TOO_SMALL,
  EN_SYSTEM_CALL_FAILED_EX,
  EN_ERROR_RECOVERY_FAILED,
  EN_UNRECOGNIZED_MEDIA,
  EN_GENERAL_SEEK_ERROR,
  EN_GENERAL_READ_ERROR,
```

```
    EN_ILLEGAL_OPERATION_FOR_TRACK,
    EN_UNSUPPORTED_READ_MODE,
    EN_REQUEST_TOO_LARGE,
    EN_FULL_ERASE_REQUIRED,
    EN_VERIFY_FAILED,
    EN_DPM_FAILED,
    EN_DEVICE_SHARING_VIOLATION,
    EN_CALL_IS_OBSOLETE,
    EN_WRONG_OS_VERSION,
    EN_INVALID_FIELD_IN_CDB,
    EN_CACHE_IS_TOO_SMALL
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
EN_SUCCESS = 0	Nothing really happened, operation completed successfully
EN_FAILURE	Undefined error happened, something goes really wrong
EN_INVALID_INPUT_PARAMETER	User input parameter is not valid
EN_INVALID_STATE	The state of the object is not valid for current operation
EN_MEMORY_ALLOCATION_FAILED	Memory allocation failed
EN_SYSTEM_CALL_FAILED	System call failed, check SystemError pointer to system error value
EN_SCSI_TRANSPORT_FAILED	SCSI transport internal error
EN_SCSI_DEVICE_BUSY	SCSI device is busy for a while
EN_SCSI_CDB_FAILED	SCSI CDB delivery failed, check CDB_FAILURE_INFORMATION (see page 480) for details
EN_SCSI_DEVICE_INVALID_TYPE	SCSI device of this type is not supported
EN_INVALID_RESPONSE	Something really unsupported returned
EN_BUFFER_UNDERRUN	Buffer underrun happened
EN_INVALID_EXCEPTION	Exception was not allocated
EN_ACCESS_TO_FEATURE_DENIED	Access to feature denied b/s of the wrong version
EN_USER_EXCEPTION	This is not a real exception just the result of user interaction
EN_PATH_TOO_LONG	File path is too long
EN_UNDER_CONSTRUCTION	This feature is still under construction
EN_NOT_FOUND	Operation could not be performed b/s either device or requested parameter not found
EN_FILE_TOO_BIG	File is too big for the requested operation
EN_NOT_IMPLEMENTED	Feature is not implemented yet
EN_RANGE	Passed range is not valid
EN_REGISTRATION_FAILED	Registration procedure completed with errors
EN_UNSUPPORTED_AUDIO	Unsupported audio format used as either input or output
EN_BUFFER_TOO_SMALL	Buffer size supplied by caller is not sufficient
EN_SYSTEM_CALL_FAILED_EX	Middle-layer error happened, check SystemError pointer for specific error value
EN_ERROR_RECOVERY_FAILED	Error recovery failed
EN_UNRECOGNIZED_MEDIA	Current media type is not recognized
EN_GENERAL_SEEK_ERROR	General seek error on CD/DVD/Blu-Ray/HD-DVD media
EN_GENERAL_READ_ERROR	General read error on CD/DVD/Blu-Ray/HD-DVD media
EN_ILLEGAL_OPERATION_FOR_TRACK	Illegal operation for track
EN_UNSUPPORTED_READ_MODE	Currently selected read mode is not supported by device
EN_REQUEST_TOO_LARGE	Request size is too large to be handled
EN_FULL_ERASE_REQUIRED	Full erase required before recording to inserted media
EN_VERIFY_FAILED	Verify operation found different memory buffers
EN_DPM_FAILED	DPM associated call failed
EN_DEVICE_SHARING_VIOLATION	Device failed to open b/s of sharing violation
EN_CALL_IS_OBSOLETE	Function is obsolete
EN_WRONG_OS_VERSION	OS version is not supported
EN_INVALID_FIELD_IN_CDB	Invalid field in CDB
EN_CACHE_IS_TOO_SMALL	Allocated cache size is too small to handle "write" operations

Description

Enum that represents exception number

Member	Definition
EN_SUCCESS	Nothing really happened, operation completed successfully
EN_FAILURE	Undefined error happened, something goes really wrong
EN_INVALID_INPUT_PARAMETER	User input parameter is not valid
EN_INVALID_STATE	The state of the object is not valid for current operation
EN_MEMORY_ALLOCATION_FAILED	Memory allocation failed
EN_SYSTEM_CALL_FAILED	System call failed, check SystemError pointer to system error value
EN_SCSI_TRANSPORT_FAILED	SCSI transport internal error
EN_SCSI_DEVICE_BUSY	SCSI device is busy for a while
EN_SCSI_CDB_FAILED	SCSI CDB delivery failed, check CDB_FAILURE_INFORMATION (see page 480) for details
EN_SCSI_DEVICE_INVALID_TYPE	SCSI device of this type is not supported
EN_INVALID_RESPONSE	Something really unsupported returned
EN_BUFFER_UNDERRUN	Buffer underrun happened
EN_INVALID_EXCEPTION	Exception was not allocated
EN_ACCESS_TO_FEATURE_DENIED	Access to feature denied b/s of the wrong version
EN_USER_EXCEPTION	This is not a real exception just the result of user interaction
EN_PATH_TOO_LONG	File path is too long
EN_UNDER_CONSTRUCTION	This feature is still under construction
EN_NOT_FOUND	Operation could not be performed b/s either device or requested parameter not found
EN_FILE_TOO_BIG	File is too big for requested operation
EN_NOT_IMPLEMENTED	Feature is not implemented yet
EN_RANGE	Passed range is not valid
EN_REGISTRATION_FAILED	Registration procedure completed with errors
EN_UNSUPPORTED_AUDIO	Unsupported audio format used as either input or output
EN_BUFFER_TOO_SMALL	Buffer size supplied by caller is not sufficient
EN_SYSTEM_CALL_FAILED_EX	Middle-layer error happened, check SystemError pointer for specific error value
EN_ERROR_RECOVERY_FAILED	Error recovery failed
EN_UNRECOGNIZED_MEDIA	Current media type is not recognized
EN_GENERAL_SEEK_ERROR	General seek error on CD/DVD/Blu-Ray/HD-DVD media
EN_GENERAL_READ_ERROR	General read error on CD/DVD/Blu-Ray/HD-DVD media
EN_ILLEGAL_OPERATION_FOR_TRACK	Illegal operation for track
EN_UNSUPPORTED_READ_MODE	Currently selected read mode is not supported by device
EN_REQUEST_TOO_LARGE	Request size is too large to be handled
EN_FULL_ERASE_REQUIRED	Full erase required before recording to inserted media
EN_VERIFY_FAILED	Verify operation found different memory buffers
EN_DPM_FAILED	DPM associated call failed
EN_DEVICE_SHARING_VIOLATION	Device failed to open b/s of sharing violation

EN_CALL_IS_OBSOLETE	Function is obsolete
EN_WRONG_OS_VERSION	OS version is not supported
EN_INVALID_FIELD_IN_CDB	Invalid field in CDB
EN_CACHE_IS_TOO_SMALL	Allocated cache size is too small to handle "write" operations

2.2.12 _FILE_TIME Enumeration

C++

```
enum _FILE_TIME {
  FILE_TIME_CREATION = 0,
  FILE_TIME_LAST_ACCESS,
  FILE_TIME_LAST_WRITE
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
FILE_TIME_CREATION = 0	Original creation file time will be included into the file system image
FILE_TIME_LAST_ACCESS	Last access time will be included into the file system image
FILE_TIME_LAST_WRITE	Last write tile will be included into the file system image

Description

Enum that represents file time that will be included in the ISO9660/Joliet image

Member	Definition
FILE_TIME_CREATION	Original creation file time will be included into the file system image
FILE_TIME_LAST_ACCESS	Last access time will be included into the file system image
FILE_TIME_LAST_WRITE	Last write tile will be included into the file system image

2.2.13 _FILE_TREE Enumeration

C++

```
enum _FILE_TREE {
  FILE_TREE_ISO9660 = 0,
  FILE_TREE_JOLIET
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
FILE_TREE_ISO9660 = 0	ISO9660 file tree
FILE_TREE_JOLIET	ISO9660 file tree + Joliet extensions

Description

Enum that represents file tree

Member	Definition
FILE_TREE_ISO9660	ISO9660 file tree
FILE_TREE_JOLIET	ISO9660 file tree + Joliet extensions

2.2.14 _FULL_TOC_ENTRY_RAW Structure

C++

```
struct _FULL_TOC_ENTRY_RAW {
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__CONTROL : 4;
  UCHAR m__UCHAR__ADR : 4;
  UCHAR m__UCHAR__TNO;
  UCHAR m__UCHAR__POINT;
  UCHAR m__UCHAR__Min;
  UCHAR m__UCHAR__Sec;
  UCHAR m__UCHAR__Frame;
  UCHAR m__UCHAR__Zero;
  UCHAR m__UCHAR__PMIN;
  UCHAR m__UCHAR__PSEC;
  UCHAR m__UCHAR__PFRAME;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UCHAR m__UCHAR__SessionNumber;`	Session number
`UCHAR m__UCHAR__CONTROL : 4;`	Control
`UCHAR m__UCHAR__ADR : 4;`	Address
`UCHAR m__UCHAR__TNO;`	Track number
`UCHAR m__UCHAR__POINT;`	Point
`UCHAR m__UCHAR__Min;`	Minute
`UCHAR m__UCHAR__Sec;`	Second
`UCHAR m__UCHAR__Frame;`	Frame
`UCHAR m__UCHAR__Zero;`	Zero (always 0)
`UCHAR m__UCHAR__PMIN;`	Starting minute (from MSF)
`UCHAR m__UCHAR__PSEC;`	Starting second (from MSF)
`UCHAR m__UCHAR__PFRAME;`	Starting frame (from MSF)

Description

Structure that represents full TOC entry

Member	Definition
m__UCHAR__SessionNumber	Session number
m__UCHAR__CONTROL : 4	Control
m__UCHAR__ADR : 4	Address
m__UCHAR__TNO	Track number
m__UCHAR__POINT	Point
m__UCHAR__Min	Minute
m__UCHAR__Sec	Second

m__UCHAR__Frame	Frame
m__UCHAR__Zero	Zero (always 0)
m__UCHAR__PMIN	Starting minute (from MSF)
m__UCHAR__PSEC	Starting second (from MSF)
m__UCHAR__PFRAME	Starting frame (from MSF)

2.2.15 _ISO9660_DATE_TIME Structure

C++

```
struct _ISO9660_DATE_TIME {
  UCHAR m__UCHAR__Year;
  UCHAR m__UCHAR__Month;
  UCHAR m__UCHAR__Day;
  UCHAR m__UCHAR__Hour;
  UCHAR m__UCHAR__Minute;
  UCHAR m__UCHAR__Second;
  UCHAR m__UCHAR__GMTOffset;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UCHAR m__UCHAR__Year;`	Year
`UCHAR m__UCHAR__Month;`	Month
`UCHAR m__UCHAR__Day;`	Day
`UCHAR m__UCHAR__Hour;`	Hour
`UCHAR m__UCHAR__Minute;`	Minute
`UCHAR m__UCHAR__Second;`	Second
`UCHAR m__UCHAR__GMTOffset;`	Offset from GMT in hours

Description

Structure that represents ISO9660 date and time

Member	Definition
m__UCHAR__Year	Year
m__UCHAR__Month	Month
M__UCHAR__Day	Day
m__UCHAR__Hour	Hour
m__UCHAR__Minute	Minute
m__UCHAR__Second	Second
m__UCHAR__GMTOffset	Offset from GMT in hours

2.2.16 _ISO9660_KIDTYPE Enumeration

C++

```
enum _ISO9660_KIDTYPE {
  KIDTYPE_UNKNOWN,
  KIDTYPE_IMPORTED,
  KIDTYPE_FROMDISK,
  KIDTYPE_VIRTUAL
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
KIDTYPE_UNKNOWN	Unknown kid type
KIDTYPE_IMPORTED	Node was imported from previous track
KIDTYPE_FROMDISK	Node was added from the disk
KIDTYPE_VIRTUAL	Node was created as memory file or virtual directory

Description

Enum that represent type of ISO9660 tree node

Member	Definition
KIDTYPE_UNKNOWN	Unknown kid type
KIDTYPE_IMPORTED	Node was imported from previous track
KIDTYPE_FROMDISK	Node was added from the disk
KIDTYPE_VIRTUAL	Node was created as memory file or virtual directory

2.2.17 _NAME_COLLISION_INFO Structure

C++

```
struct _NAME_COLLISION_INFO {
  BOOLEAN m__BOOLEAN__IsJolietName;
  UCHAR m__UCHAR__Name[ MAX_PATH * 2 ];
  PCHAR m__PCHAR__OldAbsoluteName;
  ULONG m__ULONG__OldFileAttributes;
  PCHAR m__PCHAR__NewAbsoluteName;
  ULONG m__ULONG__NewFileAttributes;
  NAME_COLLISION_SOLUTION m__NAME_COLLISION_SOLUTION;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsJolietName;	Indicates the format of the m__UCHAR__Name
UCHAR m__UCHAR__Name[MAX_PATH * 2];	The node name (CHAR for ISO9660 or USHORT for Joliet)
PCHAR m__PCHAR__OldAbsoluteName;	Absolute file name of the old file node
ULONG m__ULONG__OldFileAttributes;	Old file attributes

`PCHAR m__PCHAR__NewAbsoluteName;`	Absolute file name of the new file node
`ULONG m__ULONG__NewFileAttributes;`	New file attributes
`NAME_COLLISION_SOLUTION m__NAME_COLLISION_SOLUTION;`	The solution of the collision (what action has been performed)

Description

Structure that represents information about name collision

Member	Definition
m__BOOLEAN__IsJolietName	Indicates the format of the m__UCHAR__Name
m__UCHAR__Name	The node name (CHAR for ISO9660 or USHORT for Joliet)
m__PCHAR__OldAbsoluteName	Absolute file name of the old file node
m__ULONG__OldFileAttributes	Old file attributes
m__PCHAR__NewAbsoluteName	Absolute file name of the new file node
m__ULONG__NewFileAttributes	New file attributes
m__NAME_COLLISION_SOLUTION	The solution of the collision (what action has been performed)

2.2.18 _NAME_COLLISION_SOLUTION Enumeration

C++

```
enum _NAME_COLLISION_SOLUTION {
  NAME_COLLISION_SKIP_NEW = 0,
  NAME_COLLISION_REPLACE_OLD = 1,
  NAME_COLLISION_RENAME_NEW = 2,
  NAME_COLLISION_MERGE_FOLDERS = 3,
  NAME_COLLISION_FORCE_DWORD = 0xFFFFFFFF
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`NAME_COLLISION_SKIP_NEW = 0`	Remove new node and leave old node
`NAME_COLLISION_REPLACE_OLD = 1`	Remove old node and insert new node
`NAME_COLLISION_RENAME_NEW = 2`	Leave old node and insert new node with the new name
`NAME_COLLISION_MERGE_FOLDERS = 3`	Add files from new folder into an old folder
`NAME_COLLISION_FORCE_DWORD = 0xFFFFFFFF`	Force type size to 4 bytes

Description

Enum that represents the solutions of name collisions

Member	Definition
NAME_COLLISION_SKIP_NEW	Remove new node and leave old node
NAME_COLLISION_REPLACE_OLD	Remove old node and insert new node
NAME_COLLISION_RENAME_NEW	Leave old node and insert new node with the new name
NAME_COLLISION_MERGE_FOLDERS	Add files from new folder into an old folder

2.2.19 _PQ_SUBCHANNEL Structure

C++

```
struct _PQ_SUBCHANNEL {
  UCHAR m__UCHAR__ADR : 4;
  UCHAR m__UCHAR__CONTROL : 4;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__IndexNumber;
  UCHAR m__UCHAR__Min;
  UCHAR m__UCHAR__Sec;
  UCHAR m__UCHAR__Frame;
  UCHAR m__UCHAR__ZERO;
  UCHAR m__UCHAR__AMIN;
  UCHAR m__UCHAR__ASEC;
  UCHAR m__UCHAR__AFRAME;
  UCHAR m__UCHAR__CRC1_Reserved1;
  UCHAR m__UCHAR__CRC2_Reserved2;
  UCHAR m__UCHAR__Reserved3[ 3 ];
  UCHAR m__UCHAR__Reserved4 : 7;
  UCHAR m__UCHAR__PSubChannel : 1;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UCHAR m__UCHAR__ADR : 4;`	Track ADR
`UCHAR m__UCHAR__CONTROL : 4;`	Track CONTROL
`UCHAR m__UCHAR__TrackNumber;`	Track Number
`UCHAR m__UCHAR__IndexNumber;`	Index Number
`UCHAR m__UCHAR__Min;`	Min (from the beginning of the track)
`UCHAR m__UCHAR__Sec;`	Sec (from the beginning of the track)
`UCHAR m__UCHAR__Frame;`	Frame (from the beginning of the track)
`UCHAR m__UCHAR__ZERO;`	MBZ
`UCHAR m__UCHAR__AMIN;`	Min (from the beginning of the disc)
`UCHAR m__UCHAR__ASEC;`	Sec (from the beginning of the disc)
`UCHAR m__UCHAR__AFRAME;`	Frame (from the beginning of the disc)
`UCHAR m__UCHAR__CRC1_Reserved1;`	CRC1 or MBZ
`UCHAR m__UCHAR__CRC2_Reserved2;`	CRC2 or MBZ
`UCHAR m__UCHAR__Reserved3[ 3 ];`	MBZ
`UCHAR m__UCHAR__Reserved4 : 7;`	MBZ
`UCHAR m__UCHAR__PSubChannel : 1;`	P sub-channel bit

Description

Structure that represents formatted PQ sub-channel

Member	Definition
m__UCHAR__ADR	Track ADR
m__UCHAR__CONTROL	Track CONTROL
m__UCHAR__TrackNumber	Track Number
m__UCHAR__IndexNumber	Index Number
m__UCHAR__Min	Min (from the beginning of the track)
m__UCHAR__Sec	Sec (from the beginning of the track)

m__UCHAR__Frame	Frame (from the beginning of the track)
m__UCHAR__ZERO	MBZ
m__UCHAR__AMIN	Min (from the beginning of the disc)
m__UCHAR__ASEC	Sec (from the beginning of the disc)
m__UCHAR__AFRAME	Frame (from the beginning of the disc)
m__UCHAR__CRC1_Reserved1	CRC1 or MBZ
m__UCHAR__CRC2_Reserved2	CRC2 or MBZ
m__UCHAR__Reserved3[3]	MBZ
m__UCHAR__Reserved4	MBZ
m__UCHAR__PSubChannel	P sub-channel bit

2.2.20 _READ_MODE Enumeration

C++

```
enum _READ_MODE {
  READ_MODE_COOKED = 0,
  READ_MODE_RAW,
  READ_MODE_RAW_PQ,
  READ_MODE_RAW_RAW_PW,
  READ_MODE_PQ,
  READ_MODE_RAW_PW
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
READ_MODE_COOKED = 0	Cooked data
READ_MODE_RAW	Raw data
READ_MODE_RAW_PQ	Raw data + PQ sub-channel
READ_MODE_RAW_RAW_PW	Raw data + raw P-W sub-channel
READ_MODE_PQ	PQ sub-channel only (no main channel data)
READ_MODE_RAW_PW	Raw P-W sub-channel only (no main channel data)

Description

Enum that represents raw read modes

Member	Definition
READ_MODE_COOKED	Cooked data
READ_MODE_RAW	Raw data
READ_MODE_RAW_PQ	Raw data + PQ sub-channel
READ_MODE_RAW_RAW_PW	Raw data + raw P-W sub-channel
READ_MODE_PQ	PQ sub-channel only (no main channel data)
READ_MODE_RAW_PW	Raw P-W sub-channel only (no main channel data)

2.2.21 _SCSI_DEVICE_ADDRESS Structure

C++

```
struct _SCSI_DEVICE_ADDRESS {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__PortID;
  UCHAR m__UCHAR__BusID;
  UCHAR m__UCHAR__TargetID;
  UCHAR m__UCHAR__LUN;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsValid;	Is this data valid (structure was really filled)
UCHAR m__UCHAR__PortID;	Port Id - Logical SCSI adapter ID if ASPI is used
UCHAR m__UCHAR__BusID;	Bus ID - 0 if ASPI is used
UCHAR m__UCHAR__TargetID;	Target Id
UCHAR m__UCHAR__LUN;	LUN (Logical Unit Number)

Description

Structure that represents SCSI device address

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__PortID	Port ID - Logical SCSI adapter ID if ASPI is used
m__UCHAR__BusID	Bus ID - 0 if ASPI is used
m__UCHAR__TargetID	Target ID
m__UCHAR__LUN	LUN (Logical Unit Number)

2.2.22 _STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS Structure

C++

```
struct _STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS {
  ULONG m__ULONG__SizeInUCHARs;
  BOOLEAN m__BOOLEAN__IsCDROMRead;
  BOOLEAN m__BOOLEAN__IsCDRRead;
  BOOLEAN m__BOOLEAN__IsCDRWrite;
  BOOLEAN m__BOOLEAN__IsCDRWRead;
  BOOLEAN m__BOOLEAN__IsCDRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDROMRead;
  BOOLEAN m__BOOLEAN__IsDVDRRead;
  BOOLEAN m__BOOLEAN__IsDVDRWrite;
  BOOLEAN m__BOOLEAN__IsDVDRDLRead;
```

```
  BOOLEAN m__BOOLEAN__IsDVDRDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDRWRead;
  BOOLEAN m__BOOLEAN__IsDVDRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDRWDLRead;
  BOOLEAN m__BOOLEAN__IsDVDRWDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDRAMRead;
  BOOLEAN m__BOOLEAN__IsDVDRAMWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite;
  BOOLEAN m__BOOLEAN__IsBDROMRead;
  BOOLEAN m__BOOLEAN__IsBDRRead;
  BOOLEAN m__BOOLEAN__IsBDRWrite;
  BOOLEAN m__BOOLEAN__IsBDRDLRead;
  BOOLEAN m__BOOLEAN__IsBDRDLWrite;
  BOOLEAN m__BOOLEAN__IsBDRERead;
  BOOLEAN m__BOOLEAN__IsBDREWrite;
  BOOLEAN m__BOOLEAN__IsBDREDLRead;
  BOOLEAN m__BOOLEAN__IsBDREDLWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDROMRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRDLRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRWRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
ULONG m__ULONG__SizeInUCHARs;	This structure size in UCHARs
BOOLEAN m__BOOLEAN__IsCDROMRead;	Is CD-ROM read capable
BOOLEAN m__BOOLEAN__IsCDRRead;	Is CD-R read capable
BOOLEAN m__BOOLEAN__IsCDRWrite;	Is CD-R write capable
BOOLEAN m__BOOLEAN__IsCDRWRead;	Is CD-RW read capable
BOOLEAN m__BOOLEAN__IsCDRWWrite;	Is CD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDROMRead;	Is DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsDVDRRead;	Is DVD-R read capable
BOOLEAN m__BOOLEAN__IsDVDRWrite;	Is DVD-R write capable
BOOLEAN m__BOOLEAN__IsDVDRDLRead;	Is DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsDVDRDLWrite;	Is DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsDVDRWRead;	Is DVD-RW read capable
BOOLEAN m__BOOLEAN__IsDVDRWWrite;	Is DVD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDRWDLRead;	Is DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDRWDLWrite;	Is DVD-RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDRAMRead;	Is DVD-RAM read capable
BOOLEAN m__BOOLEAN__IsDVDRAMWrite;	Is DVD-RAM write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead;	Is DVD+RW read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite;	Is DVD+RW write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead;	Is DVD+RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite;	Is DVD+RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRRead;	Is DVD+R read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite;	Is DVD+R write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead;	Is DVD+R DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite;	Is DVD+R DL write capable

BOOLEAN m__BOOLEAN__IsBDROMRead;	Is BD-ROM read capable
BOOLEAN m__BOOLEAN__IsBDRRead;	Is BD-R read capable
BOOLEAN m__BOOLEAN__IsBDRWrite;	Is BD-R write capable
BOOLEAN m__BOOLEAN__IsBDRDLRead;	Is BD-R DL read capable
BOOLEAN m__BOOLEAN__IsBDRDLWrite;	Is BD-R DL write capable
BOOLEAN m__BOOLEAN__IsBDRERead;	Is BD-RE read capable
BOOLEAN m__BOOLEAN__IsBDREWrite;	Is BD-RE write capable
BOOLEAN m__BOOLEAN__IsBDREDLRead;	Is BD-RE DL read capable
BOOLEAN m__BOOLEAN__IsBDREDLWrite;	Is BD-RE DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDROMRead;	Is HD-DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsHDDVDRRead;	Is HD-DVD-R read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWrite;	Is HD-DVD-R write capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLRead;	Is HD-DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite;	Is HD-DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWRead;	Is HD-DVD-RW read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWWrite;	Is HD-DVD-RW write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead;	Is HD-DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite;	Is HD-DVD-RW DL write capable

Description

Structure that represents advanced supported media formats

Member Definition
--------- ----------
ULONG m__ULONG__SizeInUCHARs This structure size in UCHARs
BOOLEAN m__BOOLEAN__IsCDROMRead Is CD-ROM read capable
BOOLEAN m__BOOLEAN__IsCDRRead Is CD-R read capable
BOOLEAN m__BOOLEAN__IsCDRWrite Is CD-R write capable
BOOLEAN m__BOOLEAN__IsCDRWRead Is CD-RW read capable
BOOLEAN m__BOOLEAN__IsCDRWWrite Is CD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDROMRead Is DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsDVDRRead Is DVD-R read capable
BOOLEAN m__BOOLEAN__IsDVDRWrite Is DVD-R write capable
BOOLEAN m__BOOLEAN__IsDVDRDLRead Is DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsDVDRDLWrite Is DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsDVDRWRead Is DVD-RW read capable
BOOLEAN m__BOOLEAN__IsDVDRWWrite Is DVD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDRWDLRead Is DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDRWDLWrite Is DVD-RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDRAMRead Is DVD-RAM read capable
BOOLEAN m__BOOLEAN__IsDVDRAMWrite Is DVD-RAM write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead Is DVD+RW read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite Is DVD+RW write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead Is DVD+RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite Is DVD+RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRRead Is DVD+R read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite Is DVD+R write capable

BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead Is DVD+R DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite Is DVD+R DL write capable
BOOLEAN m__BOOLEAN__IsBDROMRead Is BD-ROM read capable
BOOLEAN m__BOOLEAN__IsBDRRead Is BD-R read capable
BOOLEAN m__BOOLEAN__IsBDRWrite Is BD-R write capable
BOOLEAN m__BOOLEAN__IsBDRDLRead Is BD-R DL read capable
BOOLEAN m__BOOLEAN__IsBDRDLWrite Is BD-R DL write capable
BOOLEAN m__BOOLEAN__IsBDRERead Is BD-RE read capable
BOOLEAN m__BOOLEAN__IsBDREWrite Is BD-RE write capable
BOOLEAN m__BOOLEAN__IsBDREDLRead Is BD-RE DL read capable
BOOLEAN m__BOOLEAN__IsBDREDLWrite Is BD-RE DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDROMMRead Is HD-DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsHDDVDRRead Is HD-DVD-R read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWrite Is HD-DVD-R write capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLRead Is HD-DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite Is HD-DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWRead Is HD-DVD-RW read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWWrite Is HD-DVD-RW write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead Is HD-DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite Is HD-DVD-RW DL write capable

2.2.23 _STARBURN_BDRE_FORMAT_PROFILE Structure

C++

```
struct _STARBURN_BDRE_FORMAT_PROFILE {
  LONG m__LONG__FormatProfileNumber;
  ULONG m__ULONG__NumberOfBlocksOrUserAreaDataSize;
  UCHAR m__UCHAR__CertificationType : 2;
  UCHAR m__UCHAR__FormatType : 6;
  ULONG m__ULONG__SpareAreaSizeInClustersOrBlockLength;
};
```

File

StarBurn.h (see page 662)

Description

Structure that represents format profile for BD-RE media

Member Definition
--------- ----------
LONG m__LONG__FormatProfileNumber Format profile number
UCHAR m__ULONG__NumberOfBlocksOrUserAreaDataSize Number of logical block(s) or user area data size
UCHAR m__UCHAR__CertificationType Certification type
UCHAR m__UCHAR__FormatType Format type

UCHAR m__ULONG__SpareAreaSizeInClustersOrBlockLength Spare area size in cluster(s) or block length

2.2.24 _STARBURN_CD_MODE Enumeration

C++

```
enum _STARBURN_CD_MODE {
  CD_MODE_UNKNOWN = 0,
  CD_MODE_AUDIO,
  CD_MODE_VIDEO,
  CD_MODE_DATA,
  CD_MODE_RESERVED
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
CD_MODE_UNKNOWN = 0	Unknown CD mode
CD_MODE_AUDIO	Audio CD mode
CD_MODE_VIDEO	Video CD mode
CD_MODE_DATA	Data CD mode
CD_MODE_RESERVED	Reserved

Description

Enum that represents CD modes type

Member	Definition
CD_MODE_UNKNOWN	Unknown CD mode
CD_MODE_AUDIO	Audio CD mode
CD_MODE_VIDEO	Video CD mode
CD_MODE_DATA	Data CD mode
CD_MODE_RESERVED	Reserved

2.2.25 _STARBURN_DISC_ATIP_INFORMATION Structure

C++

```
struct _STARBURN_DISC_ATIP_INFORMATION {
  USHORT m__USHORT__DataLength;
  UCHAR m__UCHAR__Reserved1[2];
  UCHAR m__UCHAR__RefrenceSpeed : 3;
  UCHAR m__UCHAR__Reserved2 : 1;
  UCHAR m__UCHAR__IndicativeTargetWritingPower : 4;
  UCHAR m__UCHAR__Reserved3 : 6;
  UCHAR m__UCHAR__URU : 1;
  UCHAR m__UCHAR__Zero : 1;
  UCHAR m__UCHAR__IsA3Valid : 1;
  UCHAR m__UCHAR__IsA2Valid : 1;
  UCHAR m__UCHAR__IsA1Valid : 1;
  UCHAR m__UCHAR__DiscSubType : 3;
  UCHAR m__UCHAR__DiscType : 1;
```

```
    UCHAR m__UCHAR__One : 1;
    UCHAR m__UCHAR__Reserved4;
    UCHAR m__UCHAR__LeadInStartTime[3];
    UCHAR m__UCHAR__Reserved5;
    UCHAR m__UCHAR__LastPossibleLeadOutStartTime[3];
    UCHAR m__UCHAR__Reserved6;
    UCHAR m__UCHAR__A1Values[3];
    UCHAR m__UCHAR__Reserved7;
    UCHAR m__UCHAR__A2Values[3];
    UCHAR m__UCHAR__Reserved8;
    UCHAR m__UCHAR__A3Values[3];
    UCHAR m__UCHAR__Reserved9;
    UCHAR m__UCHAR__S4Values[3];
    UCHAR m__UCHAR__Reserved10;
  };
```

File

StarBurn.h (see page 662)

Description

Structure that represents Disc ATIP information

Member Definition
--- ---
m__USHORT__DataLength
m__UCHAR__Reserved1
m__UCHAR__RefrenceSpeed
m__UCHAR__Reserved2
m__UCHAR__IndicativeTargetWritingPower
m__UCHAR__Reserved3
m__UCHAR__URU
m__UCHAR__Zero
m__UCHAR__IsA3Valid
m__UCHAR__IsA2Valid
m__UCHAR__IsA1Valid
m__UCHAR__DiscSubType
m__UCHAR__DiscType
m__UCHAR__One
m__UCHAR__Reserved4
m__UCHAR__LeadInStartTime
m__UCHAR__Reserved5
m__UCHAR__LastPossibleLeadOutStartTime
m__UCHAR__Reserved6
m__UCHAR__A1Values
m__UCHAR__Reserved7
m__UCHAR__A2Values
m__UCHAR__Reserved8
m__UCHAR__A3Values
m__UCHAR__Reserved9

m__UCHAR__S4Values
m__UCHAR__Reserved10

2.2.26 _STARBURN_DISC_INFORMATION Structure

C++

```
struct _STARBURN_DISC_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__DiscStatus;
  UCHAR m__UCHAR__LastSessionStatus;
  BOOLEAN m__BOOLEAN__IsErasable;
  UCHAR m__UCHAR__FirstTrackNumber;
  UCHAR m__UCHAR__NumberOfSessions;
  UCHAR m__UCHAR__LastSessionFirstTrack;
  UCHAR m__UCHAR__LastSessionLastTrack;
  BOOLEAN m__BOOLEAN__IsGEN;
  BOOLEAN m__BOOLEAN__IsDBCValid;
  BOOLEAN m__BOOLEAN__IsDIDValid;
  UCHAR m__UCHAR__DiscType;
  UCHAR m__UCHAR__DiscIdentification[ 4 ];
  UCHAR m__UCHAR__LastSessionLeadIn[ 4 ];
  UCHAR m__UCHAR__LastPossibleStartTime[ 4 ];
  UCHAR m__UCHAR__DiscBarCode[ 8 ];
  UCHAR m__UCHAR__NumberOfOPCEntries;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__DiscStatus;`	Disc status (See DISC_STATUS_XXX constants)
`UCHAR m__UCHAR__LastSessionStatus;`	Last session status (See LAST_SESSION_XXX constants)
`BOOLEAN m__BOOLEAN__IsErasable;`	Is disc erasable (CD-RW or DVD-RW or DVD+RW or DVD-RAM)
`UCHAR m__UCHAR__FirstTrackNumber;`	First track number on the disc
`UCHAR m__UCHAR__NumberOfSessions;`	Number of sessions on the disc
`UCHAR m__UCHAR__LastSessionFirstTrack;`	First track number of last complete session on the disc
`UCHAR m__UCHAR__LastSessionLastTrack;`	Last track number of last complete session on the disc
`BOOLEAN m__BOOLEAN__IsGEN;`	Unrestricted use, when set to one - every application can write to the disc
`BOOLEAN m__BOOLEAN__IsDBCValid;`	Is Disc Bar Code field valid
`BOOLEAN m__BOOLEAN__IsDIDValid;`	Is Disc Identification field Valid
`UCHAR m__UCHAR__DiscType;`	Disc type (See DISC_TYPE_XXX constants)
`UCHAR m__UCHAR__DiscIdentification[ 4 ];`	Disc identification number recorded in the PMA, see m__UCHAR__IsDIDValid
`UCHAR m__UCHAR__LastSessionLeadIn[ 4 ];`	MSF of last session lead in start time
`UCHAR m__UCHAR__LastPossibleStartTime[ 4 ];`	Last possible start time for lead out on this disc
`UCHAR m__UCHAR__DiscBarCode[ 8 ];`	Disc bar code stored here if the device is capable of reading disc bar codes
`UCHAR m__UCHAR__NumberOfOPCEntries;`	Number of optimum power calibration entries at the end of this structure

Description

Structure that represents Disc information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__DiscStatus	Disc status (See DISC_STATUS_XXX constants)

m__UCHAR__LastSessionStatus	Last session status (See LAST_SESSION_XXX constants)
m__BOOLEAN__IsErasable	Is disc erasable (CD-RW or DVD-RW or DVD+RW or DVD-RAM)
m__UCHAR__FirstTrackNumber	First track number on the disc
m__UCHAR__NumberOfSessions	Number of sessions on the disc
m__UCHAR__LastSessionFirstTrack	First track number of last complete session on the disc
m__UCHAR__LastSessionLastTrack	Last track number of last complete session on the disc
m__BOOLEAN__IsGEN	Unrestricted use, when set to one - every application can write to the disc
m__BOOLEAN__IsDBCValid	Is Disc Bar Code field valid
m__BOOLEAN__IsDIDValid	Is Disc Identification field Valid
m__UCHAR__DiscType	Disc type (See DISC_TYPE_XXX constants)
m__UCHAR__DiscIdentification	Disc identification number recorded in the PMA, see m__UCHAR__IsDIDValid
m__UCHAR__LastSessionLeadId	MSF of last session lead in start time
m__UCHAR__LastPossibleStartTime	Last possible start time for lead out on this disc
m__UCHAR__DiscBarCode	Disc bar code stored here if the device is capable of reading disc bar codes
m__UCHAR__NumberOfOPCEntries	Number of optimum power calibration entries at the end of this structure

2.2.27 _STARBURN_ELTORITO_MEDIA Enumeration

C++

```
enum _STARBURN_ELTORITO_MEDIA {
  ELTORITO_MEDIA_CUSTOM = 0x00,
  ELTORITO_MEDIA_FLOPPY120 = 0x01,
  ELTORITO_MEDIA_FLOPPY144 = 0x02,
  ELTORITO_MEDIA_FLOPPY288 = 0x03,
  ELTORITO_MEDIA_HARDDISK = 0x04
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
ELTORITO_MEDIA_CUSTOM = 0x00	Custom bootable media emulation
ELTORITO_MEDIA_FLOPPY120 = 0x01	1.2 MB floppy bootable media emulation
ELTORITO_MEDIA_FLOPPY144 = 0x02	1.44 MB floppy bootable media emulation
ELTORITO_MEDIA_FLOPPY288 = 0x03	2.88 MB floppy bootable media emulation
ELTORITO_MEDIA_HARDDISK = 0x04	Hard disk bootable media emulation

Description

Media emulation types

2.2.28 _STARBURN_ELTORITO_PLATFORM Enumeration

C++

```
enum _STARBURN_ELTORITO_PLATFORM {
  ELTORITO_PLATFORM_80X86 = 0x00,
  ELTORITO_PLATFORM_POWERPC = 0x01,
  ELTORITO_PLATFORM_MAC = 0x02,
```

```
  ELTORITO_PLATFORM_EFI = 0xEF
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
ELTORITO_PLATFORM_80X86 = 0x00	x86 platform identifier
ELTORITO_PLATFORM_POWERPC = 0x01	PowerPC platform identifier
ELTORITO_PLATFORM_MAC = 0x02	MAC platform identifier
ELTORITO_PLATFORM_EFI = 0xEF	EFI platform identifier

Description

ElTorito platform types

2.2.29 _STARBURN_ISO9660_DIRECTORY_INFO Structure

C++

```
struct _STARBURN_ISO9660_DIRECTORY_INFO {
  unsigned char ISOType;
  unsigned long NumberOfKids;
  unsigned long TotalNumberOfFiles;
  unsigned long TotalNumberOfDirs;
  unsigned __int64 TotalChildrenSizeInLBs;
};
```

File

StarBurn.h (see page 662)

Description

This is record _STARBURN_ISO9660_DIRECTORY_INFO.

2.2.30 _STARBURN_ISO9660_FILE_INFO Structure

C++

```
struct _STARBURN_ISO9660_FILE_INFO {
  char Name[ ( STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS + 1 ) ];
  unsigned short UnicodeName[ ( STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS + 1 ) ];
  unsigned short UnicodePathName[ ( STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS + 1 ) ];
  BOOL IsImported;
  unsigned __int64 ContentSizeInUCHARs;
  unsigned long SizeInLogicalBlocks;
  unsigned long GUID;
  unsigned long Attributes;
  unsigned long BeginLBA;
  unsigned long EndLBA;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
unsigned long BeginLBA;	File location on disc or in the image it is valid only for imported nodes i.e. when IsImported is TRUE

Description

ISO9660 file information

2.2.31 _STARBURN_STARWAVE_CALLBACK_REASON Enumeration

C++

```
enum _STARBURN_STARWAVE_CALLBACK_REASON {
  STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0,
  STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

Description

StarWave callback reasons we'll be using

Member	Definition
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

2.2.32 _STARBURN_STARWAVE_COMPRESSION Enumeration

C++

```
enum _STARBURN_STARWAVE_COMPRESSION {
  STARBURN_STARWAVE_COMPRESSION_NONE = 0,
  STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

Member	Definition
STARBURN_STARWAVE_COMPRESSION_NONE	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSS...	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98	WMA compressed stream, VBR at Quality 98
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K	WMA compressed stream, CBR at 32 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K	WMA compressed stream, CBR at 320 Kbps

2.2.33 _STARBURN_STARWAVE2_COMPRESS_TYPE Enumeration

C++

```
enum _STARBURN_STARWAVE2_COMPRESS_TYPE {
  COMPRESS_TYPE_INVALID_VALUE = 0,
  COMPRESS_TYPE_MP3_CBR,
  COMPRESS_TYPE_MP3_ABR,
  COMPRESS_TYPE_MP3_VBR,
  COMPRESS_TYPE_OGG_CBR,
  COMPRESS_TYPE_OGG_ABR,
  COMPRESS_TYPE_OGG_VBR,
  COMPRESS_TYPE_OGG_APPROXIMATE,
  COMPRESS_TYPE_WMA_LOSSLESS_VBR,
  COMPRESS_TYPE_WMA_CBR,
  COMPRESS_TYPE_WMA_VBR,
  COMPRESS_TYPE_UNCOMPRESSED_WAV_44100_STEREO,
  COMPRESS_TYPE_CUSTOM,
  COMPRESS_TYPE_FORCE_INT = 0xFFFFFFFF
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
COMPRESS_TYPE_INVALID_VALUE = 0	Bad type
COMPRESS_TYPE_MP3_CBR	MP3 Types
COMPRESS_TYPE_OGG_CBR	OGG Types
COMPRESS_TYPE_WMA_LOSSLESS_VBR	WMA Types
COMPRESS_TYPE_UNCOMPRESSED_WAV_44100_STEREO	WAV Types
COMPRESS_TYPE_CUSTOM	... Custom Types
COMPRESS_TYPE_FORCE_INT = 0xFFFFFFFF	Force type size to 4 bytes

Description

Compress settings

2.2.34 _STARBURN_STARWAVE2_CONVERSION_MODE Enumeration

C++

```
enum _STARBURN_STARWAVE2_CONVERSION_MODE {
  STARBURN_STARWAVE2_OGG_MODE0,
  STARBURN_STARWAVE2_OGG_MODE1,
  STARBURN_STARWAVE2_OGG_MODE2,
  STARBURN_STARWAVE2_OGG_MODE_MAX = STARBURN_STARWAVE2_OGG_MODE2,
  STARBURN_STARWAVE2_MP3_MODE0,
  STARBURN_STARWAVE2_MP3_MODE1,
  STARBURN_STARWAVE2_MP3_MODE2,
  STARBURN_STARWAVE2_MP3_MODE_MAX = STARBURN_STARWAVE2_MP3_MODE2
};
```

File

StarBurn.h (see page 662)

Description

Structure that represents conversion mode

Member	Definition
OGG_MODE0	VBR, variable bit rate
OGG_MODE1	ABR, average bit rate
OGG_MODE2	CBR, constant bit rate
OGG_MODE_MAX	Now equal to OGG_MODE2
MP3_MODE0	CBR, constant bit rate
MP3_MODE1	ABR, average bit rate
MP3_MODE2	VBR, variable bit rate
MP3_MODE_MAX	Now equal to MP3_MODE2

2.2.35 _STARBURN_STARWAVE2_INIT_PARAMS Structure

C++

```
struct _STARBURN_STARWAVE2_INIT_PARAMS {
  STARBURN_STARWAVE2_CONVERSION_MODE m__STARBURN_STARWAVE2_CONVERSION_MODE;
  FLOAT m__FLOAT__flQuality;
  LONG m__LONG__nQuality;
  LONG m__LONG__maxBitrate;
  LONG m__LONG__nominalBitrate;
  LONG m_LONG__minBitrate;
  STARBURN_STARWAVE2_QUALITY_MODE m__STARBURN_STARWAVE2_QUALITY_MODE;
};
```

File

StarBurn.h (see page 662)

Description

Structure that represents initialization parameters for audio conversion

Member	Definition
m__STARBURN_STARWAVE2_CONVERSION_MODE	Conversion mode
m__FLOAT__flQuality	Quality level from 0. (lo) to 1. (hi) (use for OGG_MODE0)
m__LONG__nQuality	Quality level from 1 to 9 (use for MP3_MODE2)
m__LONG__maxBitrate	Maximum bit rate (use for OGG_MODE1/OGG_MODE2)
m__LONG__nominalBitrate	Nominal bit rate (use for OGG_MODE1/OGG_MODE2/MP3_MODE0/MP3_MODE1)
m__LONG__minBitrate	Minimum bit rate (use for OGG_MODE1/OGG_MODE2)
m__StarBurn_StarWave2_QUALITY_MODE	Quality mode (use for /MP3_MODE0/MP3_MODE1/MP3_MODE2)

2.2.36 _STARBURN_STARWAVE2_QUALITY_MODE Enumeration

C++

```
enum _STARBURN_STARWAVE2_QUALITY_MODE {
  STARBURN_STARWAVE2_QM_FAST = 0,
  STARBURN_STARWAVE2_QM_STANDARD,
  STARBURN_STARWAVE2_QM_HIGH
};
```

File

StarBurn.h (see page 662)

Description

Enum that represents MP3 quality for conversion

Member	Definition
StarBurn_StarWave2_QM_FAST	Fast mode
StarBurn_StarWave2_QM_STANDARD	Standard mode
StarBurn_StarWave2_QM_HIGH	High quality mode

2.2.37 _STARBURN_TRACK_INFORMATION Structure

C++

```
struct _STARBURN_TRACK_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__TrackMode;
  BOOLEAN m__BOOLEAN__IsCopy;
  BOOLEAN m__BOOLEAN__IsDamage;
  UCHAR m__UCHAR__DataMode;
  BOOLEAN m__BOOLEAN__IsFixedPacket;
  BOOLEAN m__BOOLEAN__IsPacket;
  BOOLEAN m__BOOLEAN__IsBlank;
  BOOLEAN m__BOOLEAN__IsReserved;
  BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;
  LONG m__LONG__TrackStartAddress;
  LONG m__LONG__NextWritableAddress;
  LONG m__LONG__FreeLBs;
  LONG m__LONG__FixedPacketSizeInLBs;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsValid;	Is this data valid (structure was really filled)
UCHAR m__UCHAR__TrackNumber;	Track number of this track
UCHAR m__UCHAR__SessionNumber;	Session number that contains this track
UCHAR m__UCHAR__TrackMode;	Track mode

BOOLEAN m__BOOLEAN__IsCopy;	Is this track second or higher generation copy
BOOLEAN m__BOOLEAN__IsDamage;	Is track damaged
UCHAR m__UCHAR__DataMode;	Data mode on the track
BOOLEAN m__BOOLEAN__IsFixedPacket;	Is fixed packet used on this track
BOOLEAN m__BOOLEAN__IsPacket;	Is packet recording used on the track
BOOLEAN m__BOOLEAN__IsBlank;	Is track blank
BOOLEAN m__BOOLEAN__IsReserved;	Is track reserved
BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;	Is NWA (Next Writable Address) valid
LONG m__LONG__TrackStartAddress;	Starting address (LBA or MSF) for this track
LONG m__LONG__NextWritableAddress;	NWA (Next Writable Address)
LONG m__LONG__FreeLBs;	Free logical blocks on this track
LONG m__LONG__FixedPacketSizeInLBs;	Fixed packet size in LBs

Description

Structure that represents Track information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number of this track
m__UCHAR__SessionNumber	Session number that contains this track
m__UCHAR__TrackMode	Track mode
m__BOOLEAN__IsCopy	Is this track second or higher generation copy
m__BOOLEAN__IsDamage	Is track damaged
m__UCHAR__DataMode	Data mode on the track
m__BOOLEAN__IsFixedPacket	Is fixed packet used on this track
m__BOOLEAN__IsPacket	Is packet recording used on the track
m__BOOLEAN__IsBlank	Is track blank
m__BOOLEAN__IsReserved	Is track reserved
m__BOOLEAN__IsNextWritable...	Is NWA (Next Writable Address) valid
m__LONG__TrackStartAddress	Starting address (LBA or MSF) for this track
m__LONG__NextWritableAddress	NWA (Next Writable Address)
m__LONG__FreeLBs	Free logical blocks on this track
m__LONG__FixedPacketSize...	Fixed packet size in UCHARs

2.2.38 _STARBURN_TRACK_INFORMATION_EX Structure

C++

```
struct _STARBURN_TRACK_INFORMATION_EX {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__TrackMode;
  BOOLEAN m__BOOLEAN__IsCopy;
  BOOLEAN m__BOOLEAN__IsDamage;
  UCHAR m__UCHAR__DataMode;
  BOOLEAN m__BOOLEAN__IsFixedPacket;
  BOOLEAN m__BOOLEAN__IsPacket;
  BOOLEAN m__BOOLEAN__IsBlank;
  BOOLEAN m__BOOLEAN__IsReserved;
```

```
  BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;
  LONG m__LONG__TrackStartAddress;
  LONG m__LONG__LastRecordedAddress;
  LONG m__LONG__NextWritableAddress;
  LONG m__LONG__FreeLBs;
  LONG m__LONG__FixedPacketSizeInLBs;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__TrackNumber;`	Track number of this track
`UCHAR m__UCHAR__SessionNumber;`	Session number that contains this track
`UCHAR m__UCHAR__TrackMode;`	Track mode
`BOOLEAN m__BOOLEAN__IsCopy;`	Is this track second or higher generation copy
`BOOLEAN m__BOOLEAN__IsDamage;`	Is track damaged
`UCHAR m__UCHAR__DataMode;`	Data mode on the track
`BOOLEAN m__BOOLEAN__IsFixedPacket;`	Is fixed packet used on this track
`BOOLEAN m__BOOLEAN__IsPacket;`	Is packet recording used on the track
`BOOLEAN m__BOOLEAN__IsBlank;`	Is track blank
`BOOLEAN m__BOOLEAN__IsReserved;`	Is track reserved
`BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;`	Is NWA (Next Writable Address) valid
`LONG m__LONG__TrackStartAddress;`	Starting address (LBA or MSF) for this track
`LONG m__LONG__LastRecordedAddress;`	Last Recorded Address (LBA or MSF) for this track
`LONG m__LONG__NextWritableAddress;`	NWA (Next Writable Address)
`LONG m__LONG__FreeLBs;`	Free logical blocks on this track
`LONG m__LONG__FixedPacketSizeInLBs;`	Fixed packet size in LBs

Description

Structure that represents Track information extended

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number of this track
m__UCHAR__SessionNumber	Session number that contains this track
m__UCHAR__TrackMode	Track mode
m__BOOLEAN__IsCopy	Is this track second or higher generation copy
m__BOOLEAN__IsDamage	Is track damaged
m__UCHAR__DataMode	Data mode on the track
m__BOOLEAN__IsFixedPacket	Is fixed packet used on this track
m__BOOLEAN__IsPacket	Is packet recording used on the track
m__BOOLEAN__IsBlank	Is track blank
m__BOOLEAN__IsReserved	Is track reserved
m__BOOLEAN__IsNextWritable...	Is NWA (Next Writable Address) valid
m__LONG__TrackStartAddress	Starting address (LBA or MSF) for this track
m__LONG__LastRecordedAddress	Last Recorded Address (LBA or MSF) for this track
m__LONG__NextWritableAddress	NWA (Next Writable Address)
m__LONG__FreeLBs	Free logical blocks on this track
m__LONG__FixedPacketSize...	Fixed packet size in UCHARs

2.2.39 _STARBURN_UDF_BRIDGE_TYPE Enumeration

C++

```
enum _STARBURN_UDF_BRIDGE_TYPE {
  UDF_BRIDGE_TYPE_NONE = 0,
  UDF_BRIDGE_TYPE_DVDVIDEO,
  UDF_BRIDGE_TYPE_ISO9660,
  UDF_BRIDGE_TYPE_ISO9660_JOLIET,
  UDF_BRIDGE_TYPE_FORCE_DWORD = 0xFFFFFFFF
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
UDF_BRIDGE_TYPE_NONE = 0	Pure UDF
UDF_BRIDGE_TYPE_DVDVIDEO	UDF - DVDVIDEO/ISO9660 Bridge
UDF_BRIDGE_TYPE_ISO9660	UDF - ISO9660 Bridge
UDF_BRIDGE_TYPE_ISO9660_JOLIET	UDF - ISO9660/Joliet Bridge
UDF_BRIDGE_TYPE_FORCE_DWORD = 0xFFFFFFFF	Force size to 4 Bytes

Description

Restore original structure packing

2.2.40 _STARBURN_UDF2_DIRECTORY_INFO Structure

C++

```
struct _STARBURN_UDF2_DIRECTORY_INFO {
  unsigned long NumberOfKids;
  unsigned long NumberOfKidsDirectories;
};
```

File

StarBurn.h (see page 662)

Description

UDF directory information

2.2.41 _STARBURN_UDF2_FILE_DATE_TIME Structure

C++

```
struct _STARBURN_UDF2_FILE_DATE_TIME {
  unsigned short Year;
  unsigned char Month;
  unsigned char Day;
  unsigned char Hour;
  unsigned char Minute;
  unsigned char Second;
  unsigned char Millisecond;
  short Offset : 12;
  short IsLocal : 1;
  short Reserved : 3;
```

```
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`short Offset : 12;`	UTC offset in minutes (applicable only if Type is set to 1)
`short IsLocal : 1;`	Flag, that indicates is it local time or UTC
`short Reserved : 3;`	Should be set to 0

Description

Structure that represents UDF2 file date and time

Member	Definition
Year	Year
Month	Month
Day	Day
Hour	Hour
Minute	Minute
Second	Second
Millisecnd	Millisecond
Offset	UTC offset in minutes

2.2.42 _STARBURN_UDF2_FILE_DATE_TIME_TYPE Enumeration

C++

```
enum _STARBURN_UDF2_FILE_DATE_TIME_TYPE {
  UDF_FILE_DATE_TIME_TYPE_CREATION = 0,
  UDF_FILE_DATE_TIME_TYPE_LAST_ACCESS,
  UDF_FILE_DATE_TIME_TYPE_LAST_WRITE
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UDF_FILE_DATE_TIME_TYPE_CREATION = 0`	Node originally created
`UDF_FILE_DATE_TIME_TYPE_LAST_ACCESS`	Node last "touched
`UDF_FILE_DATE_TIME_TYPE_LAST_WRITE`	Node last written to

Description

Enum that represents StarBurn date/time type for files in UDF

Member Definition
--------- ----------

UDF_FILE_DATE_TIME_TYPE_CREATION File originally created
UDF_FILE_DATE_TIME_TYPE_LAST_ACCESS File last accessed
UDF_FILE_DATE_TIME_TYPE_LAST_WRITE File last written to

2.2.43 _STARBURN_UDF2_FILE_INFO Structure

C++

```
struct _STARBURN_UDF2_FILE_INFO {
  CHAR Name[ ( STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS + 1 ) ];
  WCHAR UnicodeName[ ( STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS + 1 ) ];
  CHAR PathName[ ( STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS + 1 ) ];
  WCHAR UnicodePathName[ ( STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS + 1 ) ];
  BOOL IsUnicode;
  BOOL IsImported;
  unsigned __int64 ImportedContentSizeInUCHARs;
  unsigned long GUID;
  unsigned long Attributes;
  unsigned long SizeInLogicalBlocks;
  unsigned __int64 SizeInUCHARs;
  unsigned long BeginLBA;
  unsigned long EndLBA;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`unsigned __int64 ImportedContentSizeInUCHARs;`	NAPALM_FILE_DATE_TIME DateTime;
`unsigned long BeginLBA;`	File location on disc or in the image it is valid only for imported nodes i.e. when IsImported is TRUE

Description

UDF file information

2.2.44 _STARPORT_DEVICE_LIST Structure

C++

```
struct _STARPORT_DEVICE_LIST {
  LONG m__LONG__NumberOfEntries;
  STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of StarPort device list entries
`STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];`	StarPort device list entries

Description

Structure that represents StarPort device list

Member	Definition
m__LONG__NumberOfEntries	Number of StarPort device list entries
m__STARPORT_DEVICE_LIST_ENTRY	StarPort device list entries

2.2.45 _STARPORT_DEVICE_LIST_ENTRY Structure

C++

```
struct _STARPORT_DEVICE_LIST_ENTRY {
  LONG m__LONG__Index;
  LONG m__LONG__TargetId;
  CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__Index;`	StarPort device index
`LONG m__LONG__TargetId;`	StarPort device TargetId
`CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];`	StarPort device name

Description

Structure that represents StarPort device list entry

Member	Definition
m__LONG__Index	StarPort device index
m__LONG__TargetId	StarPort device TargetId
m__CHAR__Name	StarPort device name

2.2.46 _STARPORT_DEVICE_TYPE Enumeration

C++

```
enum _STARPORT_DEVICE_TYPE {
  STARPORT_DEVICE_TYPE_UNKNOWN = 0,
  STARPORT_DEVICE_TYPE_RAM,
  STARPORT_DEVICE_TYPE_HDD,
  STARPORT_DEVICE_TYPE_DVD,
  STARPORT_DEVICE_TYPE_AOE,
  STARPORT_DEVICE_TYPE_ISCSI
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`STARPORT_DEVICE_TYPE_UNKNOWN = 0`	Unknown device type

STARPORT_DEVICE_TYPE_RAM	RAM disk
STARPORT_DEVICE_TYPE_HDD	Virtual hard disk
STARPORT_DEVICE_TYPE_DVD	Virtual DVD
STARPORT_DEVICE_TYPE_AOE	AoE (ATA-over-Ethernet)
STARPORT_DEVICE_TYPE_ISCSI	iSCSI (SCSI-over-IP)

Description

Enum that represents StarPort device type

Member	Definition
STARPORT_DEVICE_TYPE_UNKNOWN	Unknown device type
STARPORT_DEVICE_TYPE_RAM	RAM disk
STARPORT_DEVICE_TYPE_HDD	Virtual hard disk
STARPORT_DEVICE_TYPE_DVD	Virtual DVD
STARPORT_DEVICE_TYPE_AOE	AoE (ATA-over-Ethernet)
STARPORT_DEVICE_TYPE_ISCSI	iSCSI (SCSI-over-IP)

2.2.47 _STARWAVE2_COMPRESSION Enumeration

C++

```
enum _STARWAVE2_COMPRESSION {
  STARWAVE2_COMPRESSION_NONE = 0,
  STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10,
  STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25,
  STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50,
  STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75,
  STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90,
  STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98,
  STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000,
  STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000,
  STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000,
  STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000,
  STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000,
  STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000,
  STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000,
  STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000,
  STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000,
  STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARWAVE2_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98
STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps

STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates

All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

2.2.48 _STARWAVE2_COMPRESSION_PROFILE Structure

C++

```
struct _STARWAVE2_COMPRESSION_PROFILE {
  UINT m__UINT__CompressionType;
  PCHAR m__PCHAR__CustomFactoryID;
  union {
    int m__int__CBRBitRate;
    int m__int__VBRQuality;
  };
  int m__int__ABRBitRate;
  int m__int__MaxBitRate;
  union {
    int m__int__MinBitRate;
    int m__int__MaxFrameWindow;
  };
  LPVOID m__LPVOID__CustomData;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
UINT m__UINT__CompressionType;	(STARBURN_STARWAVE2_COMPRESS_TYPE (see page 560))
int m__int__CBRBitRate;	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320
int m__int__VBRQuality;	(MP3 compression 0..9) (WMA compression 1..100)
int m__int__ABRBitRate;	(MP3 && OGG compression)
int m__int__MaxBitRate;	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320, for CBR mode this setting is ignored set this field to 0 if you don't want to use it
int m__int__MinBitRate;	(MP3 && OGG) 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320 set this field to 0 if you don't want to use it
int m__int__MaxFrameWindow;	WMA, Frame size in milliseconds

2.2.49 _TOC_ENTRY Structure

C++

```
struct _TOC_ENTRY {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  LONG m__LONG__StartingLBA;
```

```
  UCHAR m__UCHAR__StartingMSF[ 3 ];
  LONG m__LONG__EndingLBA;
  UCHAR m__UCHAR__EndingMSF[ 3 ];
  BOOLEAN m__BOOLEAN__IsMCNAvailable;
  BOOLEAN m__BOOLEAN__IsISRCAvailable;
  BOOLEAN m__BOOLEAN__IsFourChannelsAudio;
  BOOLEAN m__BOOLEAN__IsPreEmphasisAudio;
  BOOLEAN m__BOOLEAN__IsData;
  BOOLEAN m__BOOLEAN__IsAudio;
  BOOLEAN m__BOOLEAN__IsDigitalCopyProhibited;
  UCHAR m__UCHAR__TrackMode;
  UCHAR m__UCHAR__MODE2Form;
  ULONG m__ULONG__LBSizeInUCHARs;
  LONG m__LONG__Index00;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__TrackNumber;`	Track number
`UCHAR m__UCHAR__SessionNumber;`	Session number that this track belongs to
`LONG m__LONG__StartingLBA;`	Starting LBA
`UCHAR m__UCHAR__StartingMSF[ 3 ];`	Starting MSF
`LONG m__LONG__EndingLBA;`	Ending LBA
`UCHAR m__UCHAR__EndingMSF[ 3 ];`	Ending MSF
`BOOLEAN m__BOOLEAN__IsMCNAvailable;`	Is Media Catalog Number available
`BOOLEAN m__BOOLEAN__IsISRCAvailable;`	Is International Standard Recording Code available
`BOOLEAN m__BOOLEAN__IsFourChannelsAudio;`	Is this four channels audio track
`BOOLEAN m__BOOLEAN__IsPreEmphasisAudio;`	Is this pre-emphasis audio track
`BOOLEAN m__BOOLEAN__IsData;`	Is this data track
`BOOLEAN m__BOOLEAN__IsAudio;`	Is this audio track
`BOOLEAN m__BOOLEAN__IsDigitalCopyProhibited;`	Is digital copy prohibited for this track
`UCHAR m__UCHAR__TrackMode;`	Track mode (0 for audio track, 1 for MODE1 and 2 for MODE2)
`UCHAR m__UCHAR__MODE2Form;`	MODE2 Form (0 for Formless, 1 for Form1 and 2 for Form2)
`ULONG m__ULONG__LBSizeInUCHARs;`	LB (logical block) size in UCHARs on this track
`LONG m__LONG__Index00;`	Index00 LBA (if present)

Description

Structure that represents TOC entry

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number
m__UCHAR__SessionNumber	Session number that this track belongs to
m__LONG__StartingLBA	Starting LBA
m__UCHAR__StartingMSF	Starting MSF
m__LONG__EndingLBA	Ending LBA
m__UCHAR__EndingMSF	Ending MSF
m__BOOLEAN__IsMCNAvailable	Is Media Catalog Number available
m__BOOLEAN__IsISRCAvailable	Is International Standard Recording Code available
m__BOOLEAN__IsFourChannelsAudio	Is this four channels audio track
m__BOOLEAN__IsPreEmphasisAudio	Is this pre-emphasis audio track

m__BOOLEAN__IsData	Is this data track
m__BOOLEAN__IsAudio	Is this audio track
m__BOOLEAN__IsDigitalCopy...	Is digital copy prohibited for this track
m__UCHAR__TrackMode	Track mode (0 for audio track, 1 for MODE1 and 2 for MODE2)
m__UCHAR__MODE2Form	MODE2 Form (0 for Formless, 1 for Form1 and 2 for Form2)
m__ULONG__LBSizeInUCHARs	LB (logical block) size in UCHARs on this track
m__LONG__Index00	Index00 LBA (if present)

2.2.50 _TOC_INFORMATION Structure

C++

```
struct _TOC_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  BOOLEAN m__BOOLEAN__IsDVD;
  USHORT m__USHORT__ProtectedDVDRegions;
  UCHAR m__UCHAR__BusKeyForDiscKey[ 5 ];
  UCHAR m__UCHAR__NumberOfSessions;
  UCHAR m__UCHAR__NumberOfTracks;
  UCHAR m__UCHAR__NumberOfUnsortedEntries;
  TOC_ENTRY m__TOC_ENTRY[ NUMBER_OF_TRACKS ];
  FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW[ NUMBER_OF_TRACKS ];
  FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW__Unsorted[ NUMBER_OF_RAW_TRACKS ];
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`BOOLEAN m__BOOLEAN__IsDVD;`	Is this DVD media or not
`USHORT m__USHORT__ProtectedDVDRegions;`	Number of protected DVD regions
`UCHAR m__UCHAR__BusKeyForDiscKey[ 5 ];`	Bus key for disc key (CSS)
`UCHAR m__UCHAR__NumberOfSessions;`	Number of sessions
`UCHAR m__UCHAR__NumberOfTracks;`	Number of tracks
`UCHAR m__UCHAR__NumberOfUnsortedEntries;`	Number of entries in unsorted raw TOC
`TOC_ENTRY m__TOC_ENTRY[ NUMBER_OF_TRACKS ];`	TOC entries (decoded and extended)
`FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW[ NUMBER_OF_TRACKS ];`	Full TOC raw entries sorted, each entry corresponds to m__TOC_ENTRY with the same TNO
`FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW__Unsorted[ NUMBER_OF_RAW_TRACKS ];`	Full TOC raw entries unsorted

Description

Structure that represents TOC information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__BOOLEAN__IsDVD	Is this DVD media or not
m__USHORT__ProtectedDVDRegions	Number of protected DVD regions
m__UCHAR__BusKeyForDiscKey	Bus key for disc key (CSS)
m__UCHAR__NumberOfSessions	Number of sessions

m__UCHAR__NumberOfTracks	Number of tracks
m__UCHAR__NumberOfUnsortedEntries	Number of entries in unsorted raw TOC
m__TOC_ENTRY	TOC entries (decoded and extended)
m__FULL_TOC_ENTRY_RAW	Full TOC raw entries sorted, each entry corresponds to m__TOC_ENTRY with the same TNO
m__FULL_TOC_ENTRY_RAW__Unsorted	Full TOC raw entries unsorted

2.2.51 _UDF_CONTROL_BLOCK Structure

C++

```
struct _UDF_CONTROL_BLOCK {
  void * m__PVOID__Head;
  void * m__PVOID__SystemStructures;
  void * m__PVOID__Tail;
  void * m__PVOID__Body;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`void * m__PVOID__Head;`	Pointer to the head of the linked list
`void * m__PVOID__SystemStructures;`	Pointer to the allocated and formatted UDF system structures
`void * m__PVOID__Tail;`	Pointer to the tail of the linked list
`void * m__PVOID__Body;`	Pointer to block body itself

Description

Structure that represents UDF control block

Member	Definition
m__PVOID__Head	Pointer to the head of the linked list
m__PVOID__SystemStructures	Pointer to the allocated and formatted UDF system structures
m__PVOID__Tail	Pointer to the tail of the linked list
m__PVOID__Body	Pointer to block body itself

2.2.52 _UDF_FILE_EXTENT Structure

C++

```
struct _UDF_FILE_EXTENT {
  ULONG m__ULONG__BeginLBA;
  ULONG m__ULONG__EndLBA;
};
```

File

StarBurn.h (see page 662)

Description

Structure that describes a single file extent

Member	Definition
m__ULONG__BeginLBA	First LBA of the extent
m__ULONG__EndLBA	Last LBA of the extent

2.2.53 _UDF_FILE_HANDLE Structure

C++

```
struct _UDF_FILE_HANDLE {
  HANDLE m__HANDLE;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`HANDLE m__HANDLE;`	Win32 file handle

Description

Structure that represents UDF file handle

Member	Definition
m__HANDLE	Win32 file handle

2.2.54 _UDF_FILE_LOOKUP_ENTRY Structure

C++

```
struct _UDF_FILE_LOOKUP_ENTRY {
  ULONG m__ULONG__Size;
  PWCHAR m__PCHAR__FileName;
  UINT m__UINT__ExtentCount;
  PUDF_FILE_EXTENT m__Extents;
};
```

File

StarBurn.h (see page 662)

Description

Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function

Member	Definition
m__ULONG__Size	Entry size
m__PWCHAR__FileName	File name

m__UINT__ExtentCount	Number of file extents
m__Extents	Array of file extents

2.2.55 _UDF_LOOKUP_DIR_ENTRY Structure

C++

```
struct _UDF_LOOKUP_DIR_ENTRY {
  PWCHAR m__PWCHAR__DirName;
  PVOID m__PVOID__ParentContext;
};
```

File

StarBurn.h (see page 662)

Description

Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function

Member	Definition
m__PWCHAR__DirName	Directory name
m__PVOID__ParentContext	The user context assigned to this directory

2.2.56 _UDF_LOOKUP_FILE_ENTRY Structure

C++

```
struct _UDF_LOOKUP_FILE_ENTRY {
  ULONG m__ULONG__EntrySize;
  PWCHAR m__PWCHAR__FileName;
  ULONGLONG m__QWORD__FileSize;
  UINT m__UINT__ExtentCount;
  PUDF_FILE_EXTENT m__Extents;
};
```

File

StarBurn.h (see page 662)

Description

Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function

Member	Definition
m__ULONG__EntrySize	Entry size
m__PWCHAR__FileName	File name
m__QWORD__FileSize	File size in bytes
m__UINT__ExtentCount	Number of file extents
m__Extents	Array of file extents

2.2.57 _UDF_OS_CLASS Enumeration

C++

```
enum _UDF_OS_CLASS {
  UDF_OS_CLASS_UNDEFINED = 0x00,
  UDF_OS_CLASS_DOS = 0x01,
  UDF_OS_CLASS_OS_2 = 0x02,
  UDF_OS_CLASS_MACINTOSH_OS = 0x03,
  UDF_OS_CLASS_UNIX = 0x04,
  UDF_OS_CLASS_WINDOWS_9X = 0x05,
  UDF_OS_CLASS_WINDOWS_NT = 0x06,
  UDF_OS_CLASS_OS_400 = 0x07,
  UDF_OS_CLASS_BE_OS = 0x08,
  UDF_OS_CLASS_WINDOWS_CE = 0x09
};
```

File

StarBurn.h (see page 662)

Description

UDF OS class

2.2.58 _UDF_TREE_ITEM Structure

C++

```
struct _UDF_TREE_ITEM {
  unsigned long m__ULONG__FileEntryRBA;
  unsigned long m__ULONG__FileIdentifierRBA;
  unsigned long m__ULONG__FileIdentifierParentOrContentRBA;
  unsigned long m__ULONG__LastTouchedRBA;
  unsigned long m__ULONG__GUID;
  unsigned char m__UCHAR__IsDirectory;
  unsigned char m__UCHAR__IsCached;
  unsigned short m__USHORT__NumberOfKidsAsParents;
  char m__CHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];
  UDF_FILE_HANDLE m__UDF_FILE_HANDLE;
  unsigned char * m__PUCHAR__File;
  unsigned __int64 m__ULONGLONG__SizeInUCHARs;
  unsigned long m__ULONG__SizeInLogicalBlocks;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Next;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Prev;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Kids;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Parent;
  unsigned char m__UCHAR__FileEntryDescriptor[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];
  unsigned char * m__PUCHAR__FileIdentifierDescriptor;
  unsigned long m__ULONG__FileIdentifierDescriptorSizeInUCHARs;
  unsigned char m__UCHAR__FileContent[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];
  void * m__PVOID__Context;
  ISO9660_DATE_TIME m__ISO9660_DATE_TIME;
  WCHAR m__WCHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`unsigned long m__ULONG__FileEntryRBA;`	File entry relative block address
`unsigned long m__ULONG__FileIdentifierRBA;`	File identifier relative block address

`unsigned long m__ULONG__FileIdentifierParentOrContentRBA;`	File identifier parent or content relative block address
`unsigned long m__ULONG__LastTouchedRBA;`	Last touched relative block address (last occupied)
`unsigned long m__ULONG__GUID;`	Globally unique identifier
`unsigned char m__UCHAR__IsDirectory;`	Is this directory (0x01) or file (0x00)
`unsigned char m__UCHAR__IsCached;`	Is this entry content cached (located in memory) or not cached (located on the disk)
`unsigned short m__USHORT__NumberOfKidsAsParents;`	Number of kids that have their own kids
`char m__CHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];`	Name of this node
`UDF_FILE_HANDLE m__UDF_FILE_HANDLE;`	UDF file handle of this node
`unsigned char * m__PUCHAR__File;`	Pointer for file content (for cached files)
`unsigned __int64 m__ULONGLONG__SizeInUCHARs;`	Node content size in UCHARs
`unsigned long m__ULONG__SizeInLogicalBlocks;`	Node content size in logical blocks
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Next;`	Pointer to the next UDF tree item in the linked list
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Prev;`	Pointer to the previous UDF tree item in the linked list
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Kids;`	Pointer to the kids linked list
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Parent;`	Pointer to the parent of the current UDF tree item
`unsigned char m__UCHAR__FileEntryDescriptor[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];`	Array of UCHARs holding UDF file entry descriptor for current UDF tree item
`unsigned char * m__PUCHAR__FileIdentifierDescriptor;`	Pointer to allocated file identifier
`unsigned long m__ULONG__FileIdentifierDescriptorSizeInUCHARs;`	Size of allocated FID in bytes
`unsigned char m__UCHAR__FileContent[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];`	Array of UCHARs holding file content (alternative cached data)
`void * m__PVOID__Context;`	Pointer to context value
`ISO9660_DATE_TIME m__ISO9660_DATE_TIME;`	ISO9660 date and time
`WCHAR m__WCHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];`	Name of this node

Description

Structure that represents UDF tree item

Member	Definition
m__ULONG__FileEntryRBA	File entry relative block address
m__ULONG__FileIdentifierRBA	File identifier relative block address
m__ULONG__FileIdentifier...	File identifier parent or content relative block address
m__ULONG__LastTouchedRBA	Last touched relative block address (last occupied)
m__ULONG__GUID	Globally unique identifier
m__UCHAR__IsDirectory	Is this directory (0x01) or file (0x00)
m__UCHAR__IsCached	Is this entry content cached (located in memory) or not cached (located on the disk)
m__USHORT_NumberOfKidsAsParents	Number of kids that have their own kids
m__CHAR__Name	Name of this node
m__UDF_FILE_HANDLE	UDF file handle of this node
m__PUCHAR__File	Pointer to file content (for cached files)
m__ULONGLONG__SizeInUCHARs	Node content size in UCHARs
m__ULONG__SizeInLogicalBlocks	Node content size in logical blocks
m__PUDF_TREE_ITEM__Next	Pointer to the next UDF tree item in the linked list
m__PUDF_TREE_ITEM__Prev	Pointer to the previous UDF tree item in the linked list
m__PUDF_TREE_ITEM__Kids	Pointer to the kids liked list
m__PUDF_TREE_ITEM__Parent	Pointer to the parent of the current UDF tree item
m__UCHAR__FileEntryDescriptor	Array if UCHARs holding UDF file entry descriptor for current UDF tree item
m__UCHAR__FileIdentifier...	Array of UCHARs holding UDF file identifier descriptor for current UDF tree item

m__UCHAR__FileIdentifier...	Array of UCHARs holding UDF file identifier descriptor for parent of the current UDF tree item
m__UCHAR__FileContent	Array of UCHARs holding file content (alternative cached data)
m__PVOID__Context	Pointer to context value
m__ISO9660_DATE_TIME	ISO9660 date and time

2.2.59 _UDF_VERSION Enumeration

C++

```
enum _UDF_VERSION {
  UDF_VERSION_1_02 = 0,
  UDF_VERSION_1_50,
  UDF_VERSION_2_00,
  UDF_VERSION_2_01,
  UDF_VERSION_2_50,
  UDF_VERSION_2_60
};
```

File

StarBurn.h (see page 662)

Description

Enum that represents UDF version

Member	Definition
UDF_VERSION_1_02	UDF 1.02
UDF_VERSION_1_50	UDF 1.5
UDF_VERSION_2_00	UDF 2.0
UDF_VERSION_2_01	UDF 2.01
UDF_VERSION_2_50	UDF 2.5
UDF_VERSION_2_60	UDF 2.6

2.2.60 _WAVE_FILE_HEADER Structure

C++

```
struct _WAVE_FILE_HEADER {
  ULONG m__ULONG__Riff;
  ULONG m__ULONG__Length;
  ULONG m__ULONG__Wave;
  ULONG m__ULONG__FormatTag;
  ULONG m__ULONG__FormatLength;
  WAVE_FORMAT_CHUNK m__WAVE_FORMAT_CHUNK;
  ULONG m__ULONG__DataTag;
  ULONG m__ULONG__DataLength;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`ULONG m__ULONG__Riff;`	Riff signature
`ULONG m__ULONG__Length;`	Length of data section (w/o header) in UCHARs
`ULONG m__ULONG__Wave;`	Wave signature
`ULONG m__ULONG__FormatTag;`	Format tag
`ULONG m__ULONG__FormatLength;`	Format length (size of WAVE_FORMAT_CHUNK (see page 580) in UCHARs)
`WAVE_FORMAT_CHUNK m__WAVE_FORMAT_CHUNK;`	WAVE format chunk (see WAVE_FORMAT_CHUNK (see page 580) for more details)
`ULONG m__ULONG__DataTag;`	Data tag
`ULONG m__ULONG__DataLength;`	Data length in UCHARs

Description

Structure that represents WAVE file header

Member	Definition
m__ULONG__Riff	Riff signature
m__ULONG__Length	Length of data section (w/o header) in UCHARs
m__ULONG__Wave	Wave signature
m__ULONG__FormatTag	Format tag
m__ULONG__FormatLength	Format length (size of WAVE_FORMAT_CHUNK (see page 580) in UCHARs)
m__WAVE_FORMAT_CHUNK	WAVE format chunk (see WAVE_FORMAT_CHUNK (see page 580) for more details)
m__ULONG__DataTag	Data tag
m__ULONG__DataLenght	Data length in UCHARs

2.2.61 _WAVE_FORMAT_CHUNK Structure

C++

```
struct _WAVE_FORMAT_CHUNK {
  USHORT m__USHORT__Format;
  USHORT m__USHORT__Channels;
  ULONG m__ULONG__SamplesPerSecond;
  ULONG m__ULONG__AverageSamplesPerSecond;
  USHORT m__USHORT__Alignment;
  USHORT m__USHORT__BitsPerSample;
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
`USHORT m__USHORT__Format;`	Format
`USHORT m__USHORT__Channels;`	Number of channels
`ULONG m__ULONG__SamplesPerSecond;`	Number of samples per second
`ULONG m__ULONG__AverageSamplesPerSecond;`	Number of average samples per second
`USHORT m__USHORT__Alignment;`	Alignment
`USHORT m__USHORT__BitsPerSample;`	Number of bits in single sample

Description

Structure that represents WAVE format chunk

Member	Definition
m__USHORT__Format	Format
m__USHORT__Channels	Number of channels
m__ULONG__SamplesPerSecond	Number of samples per second
m__ULONG__AverageSamplesPerSecond	Number of average samples per second
m__USHORT__Alignment	Alignment
m__USHORT__BitsPerSample	Number of bits in single sample

2.2.62 _WRITE_MODE Enumeration

C++

```
enum _WRITE_MODE {
  WRITE_MODE_TRACK_AT_ONCE = 0,
  WRITE_MODE_SESSION_AT_ONCE,
  WRITE_MODE_DISC_AT_ONCE_PQ,
  WRITE_MODE_DISC_AT_ONCE_RAW_PW
};
```

File

StarBurn.h (see page 662)

Members

Members	Description
WRITE_MODE_TRACK_AT_ONCE = 0	Track-At-Once
WRITE_MODE_SESSION_AT_ONCE	Session-At-Once
WRITE_MODE_DISC_AT_ONCE_PQ	Disc-At-Once PQ
WRITE_MODE_DISC_AT_ONCE_RAW_PW	Disc-At-Once raw P-W

Description

Enum that represents write modes

Member	Definition
WRITE_MODE_TRACK_AT_ONCE	Track-At-Once
WRITE_MODE_SESSION_AT_ONCE	Session-At-Once
WRITE_MODE_DISC_AT_ONCE_PQ	Disc-At-Once PQ
WRITE_MODE_DISC_AT_ONCE_RAW_PW	Disc-At-Once raw P-W

2.2.63 CALLBACK_NUMBER Enumeration

C++

```
typedef enum _CALLBACK_NUMBER {
  CN_FILE_TREE_PROGRESS_ADD = 0,
```

```
    CN_FILE_TREE_PROGRESS_REMOVE,
    CN_FILE_TREE_PROGRESS_IGNORE,
    CN_FILE_TREE_PROGRESS_NAME_COLLISION,
    CN_TARGET_FILE_ANALYZE_BEGIN,
    CN_TARGET_FILE_ANALYZE_END,
    CN_WAIT_CACHE_FULL_BEGIN,
    CN_WAIT_CACHE_FULL_END,
    CN_SYNCHRONIZE_CACHE_BEGIN,
    CN_SYNCHRONIZE_CACHE_END,
    CN_FIND_DEVICE,
    CN_CDVD_READ_PROGRESS,
    CN_CDVD_WRITE_PROGRESS,
    CN_CDVD_BUFFER_STATUS,
    CN_CDVD_TRACK_BEGIN,
    CN_CDVD_TRACK_END,
    CN_CDVD_SPLIT_BEGIN,
    CN_CDVD_SPLIT_END,
    CN_CDVD_READ_BAD_BLOCK_HIT,
    CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT,
    CN_CDVD_READ_RETRY,
    CN_DVDPLUSRW_FORMAT_BEGIN,
    CN_DVDPLUSRW_FORMAT_END,
    CN_DVDRAM_FORMAT_BEGIN,
    CN_DVDRAM_FORMAT_END,
    CN_BUFFER_UNDERRUN,
    CN_DVD_MEDIA_PADDING_SIZE,
    CN_DVD_MEDIA_PADDING_BEGIN,
    CN_DVD_MEDIA_PADDING_END,
    CN_CDVD_READ_CANCEL_QUERY,
    CN_DVDRW_QUICK_FORMAT_BEGIN,
    CN_DVDRW_QUICK_FORMAT_END,
    CN_DVD_TEST_WRITE_DISABLED,
    CN_CDVD_DPM_BEGIN,
    CN_CDVD_DPM_END,
    CN_CDVD_DPM_PROGRESS,
    CN_CDVD_VERIFY_PROGRESS,
    CN_SAO_TRACK_WRITE_BEGIN,
    CN_SAO_TRACK_WRITE_END,
    CN_DVD_MEDIA_PADDING_WRITE_PROGRESS,
    CN_CDVD_WRITE_BEGIN,
    CN_CDVD_WRITE_END,
    CN_CDVD_VERIFY_BEGIN,
    CN_CDVD_VERIFY_END,
    CN_BDRE_FORMAT_BEGIN,
    CN_BDRE_FORMAT_END,
    CN_UDF_FILE_LOOKUP,
    CN_UDF_LOOKUP_FILE,
    CN_UDF_LOOKUP_DIR,
    CN_CDVD_BLANKAREA_PROGRESS
} * PCALLBACK_NUMBER, CALLBACK_NUMBER;
```

File

StarBurn.h (see page 662)

Members

Members	Description
CN_FILE_TREE_PROGRESS_ADD = 0	Item was add to the file tree
CN_FILE_TREE_PROGRESS_REMOVE	Item was removed from the file tree
CN_FILE_TREE_PROGRESS_IGNORE	Item was ignored during processing file tree
CN_FILE_TREE_PROGRESS_NAME_COLLISION	There are two nodes which have the same name
CN_TARGET_FILE_ANALYZE_BEGIN	File internal structure (size, type etc) analyze started
CN_TARGET_FILE_ANALYZE_END	File structure analyze completed
CN_WAIT_CACHE_FULL_BEGIN	Toolkit started to wait for cache to become full
CN_WAIT_CACHE_FULL_END	Toolkit finished to wait for cache fullness
CN_SYNCHRONIZE_CACHE_BEGIN	Cache flushing to the media started
CN_SYNCHRONIZE_CACHE_END	Cache flushing completed
CN_FIND_DEVICE	Find device operation completed
CN_CDVD_READ_PROGRESS	CD/DVD/Blu-Ray/HD-DVD read operation progress

CN_CDVD_WRITE_PROGRESS	CD/DVD/Blu-Ray/HD-DVD write operation progress
CN_CDVD_BUFFER_STATUS	CD/DVD/Blu-Ray/HD-DVD buffer status information queried
CN_CDVD_TRACK_BEGIN	CD/DVD/Blu-Ray/HD-DVD track processing started
CN_CDVD_TRACK_END	CD/DVD/Blu-Ray/HD-DVD track processing completed
CN_CDVD_SPLIT_BEGIN	CD/DVD/Blu-Ray/HD-DVD split section processing started
CN_CDVD_SPLIT_END	CD/DVD/Blu-Ray/HD-DVD split section processing completed
CN_CDVD_READ_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a bad (unrecoverable) block
CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a ECC/EDC bad (recoverable) block
CN_CDVD_READ_RETRY	CD/DVD/Blu-Ray/HD-DVD read operation was retried
CN_DVDPLUSRW_FORMAT_BEGIN	DVD+RW format operation started
CN_DVDPLUSRW_FORMAT_END	DVD+RW format operation completed
CN_DVDRAM_FORMAT_BEGIN	DVD-RAM format operation started
CN_DVDRAM_FORMAT_END	DVD-RAM format operation completed
CN_BUFFER_UNDERRUN	Buffer underrun condition happened
CN_DVD_MEDIA_PADDING_SIZE	DVD media would be padded to 1GB size, additional info passed
CN_DVD_MEDIA_PADDING_BEGIN	DVD media padding burn process started
CN_DVD_MEDIA_PADDING_END	DVD media padding burn process completed
CN_CDVD_READ_CANCEL_QUERY	CD/DVD/Blu-Ray/HD-DVD media processing plug-in queries cancel status
CN_DVDRW_QUICK_FORMAT_BEGIN	DVD-RW quick format operation started
CN_DVDRW_QUICK_FORMAT_END	DVD-RW quick format operation completed
CN_DVD_TEST_WRITE_DISABLED	Test write is disabled (for DVD+R/RW, DVD-RAM or multisession DVD-RW)
CN_CDVD_DPM_BEGIN	CD/DVD/Blu-Ray/HD-DVD DPM processing started
CN_CDVD_DPM_END	CD/DVD/Blu-Ray/HD-DVD DPM processing completed
CN_CDVD_DPM_PROGRESS	CD/DVD/Blu-Ray/HD-DVD DPM processing progress
CN_CDVD_VERIFY_PROGRESS	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_SAO_TRACK_WRITE_BEGIN	SAO track write started
CN_SAO_TRACK_WRITE_END	SAO track write completed
CN_DVD_MEDIA_PADDING_WRITE_PROGRESS	DVD media padding burn progress indication
CN_CDVD_WRITE_BEGIN	CD/DVD/Blu-Ray/HD-DVD write operation started
CN_CDVD_WRITE_END	CD/DVD/Blu-Ray/HD-DVD write operation completed
CN_CDVD_VERIFY_BEGIN	CD/DVD/Blu-Ray/HD-DVD verify operation started
CN_CDVD_VERIFY_END	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_BDRE_FORMAT_BEGIN	BD-RE format operation started
CN_BDRE_FORMAT_END	BD-RE format operation completed
CN_UDF_FILE_LOOKUP	UDF file found during fast lookup
CN_UDF_LOOKUP_FILE	UDF file found during fast lookup
CN_UDF_LOOKUP_DIR	UDF directory found during fast lookup
CN_CDVD_BLANKAREA_PROGRESS	CD/DVD/Blu-Ray/HD-DVD blank area operation progress

Description

Enum that represents callback number

Member	Definition
CN_FILE_TREE_PROGRESS_ADD	Item was add to the file tree
CN_FILE_TREE_PROGRESS_REMOVE	Item was removed from the file tree
CN_FILE_TREE_PROGRESS_IGNORE	Item was ignored during processing file tree
CN_FILE_TREE_PROGRESS_NAME...	There are two nodes which have the same name
CN_TARGET_FILE_ANALYZE_BEGIN	File internal structure (size, type etc) analyze started
CN_TARGET_FILE_ANALYZE_END	File structure analyze completed
CN_WAIT_CACHE_FULL_BEGIN	Toolkit started to wait for cache to become full
CN_WAIT_CACHE_FULL_END	Toolkit finished to wait for cache fullness
CN_SYNCHRONIZE_CACHE_BEGIN	Cache flushing to the media started
CN_SYNCHRONIZE_CACHE_END	Cache flushing to the media completed

CN_FIND_DEVICE	Find device operation completed
CD_CDVD_READ_PROGRESS	CD/DVD/Blu-Ray/HD-DVD read operation progress
CN_CDVD_WRITE_PROGRESS	CD/DVD/Blu-Ray/HD-DVD write operation progress
CN_CDVD_BUFFER_STATUS	CD/DVD/Blu-Ray/HD-DVD buffer status information queried
CN_CDVD_TRACK_BEGIN	CD/DVD/Blu-Ray/HD-DVD track processing started
CN_CDVD_TRACK_END	CD/DVD/Blu-Ray/HD-DVD track processing completed
CN_CDVD_SPLIT_BEGIN	CD/DVD/Blu-Ray/HD-DVD split section processing started
CN_CDVD_SPLIT_END	CD/DVD/Blu-Ray/HD-DVD split section processing completed
CN_CDVD_READ_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a bad (unrecoverable) block
CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a ECC/EDC bad (recoverable) block
CN_CDVD_READ_RETRY	CD/DVD/Blu-Ray/HD-DVD read operation was retried
CN_DVDPLUSRW_FORMAT_BEGIN	DVD+RW format operation started
CN_DVDPLUSRW_FORMAT_EDN	DVD+RW format operation completed
CN_DVDRAM_FORMAT_BEGIN	DVD-RAM format operation started
CN_DVDRAM_FORMAT_EDN	DVD-RAM format operation completed
CN_BUFFER_UNDERRUN	Buffer underrun condition happened
CN_DVD_MEDIA_PADDING_SIZE	DVD media would be padded to 1GB size, additional info passed
CN_DVD_MEDIA_PADDING_BEGIN	DVD media padding burn process started
CN_DVD_MEDIA_PADDING_END	DVD media padding burn process completed
CN_CDVD_READ_CANCEL_QUERY	CD/DVD/Blu-Ray/HD-DVD media processing plug-in queries cancel status
CN_DVDRW_QUICK_FORMAT_BEGIN	DVD-RW quick format operation started
CN_DVDRW_QUICK_FORMAT_END	DVD-RW quick format operation completed
CN_DVD_TEST_WRITE_DISABLED	Test write is disabled (for DVD+R/RW, DVD-RAM or multisession DVD-RW)
CN_CDVD_DPM_BEGIN	CD/DVD/Blu-Ray/HD-DVD DPM processing started
CN_CDVD_DPM_END	CD/DVD/Blu-Ray/HD-DVD DPM processing completed
CD_CDVD_DPM_PROGRESS	CD/DVD/Blu-Ray/HD-DVD DPM operation progress
CN_CDVD_VERIFY_PROGRESS	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_SAO_TRACK_WRITE_BEGIN	SAO track write started
CN_SAO_TRACK_WRITE_END	SAO track write completed
CN_DVD_MEDIA_PADDING_WRITE_PRO...	DVD media padding burn progress indication
CN_CDVD_WRITE_BEGIN	CD/DVD/Blu-Ray/HD-DVD write operation started
CN_CDVD_WRITE_END	CD/DVD/Blu-Ray/HD-DVD write operation completed
CN_CDVD_VERIFY_BEGIN	CD/DVD/Blu-Ray/HD-DVD verify operation started
CN_CDVD_VERIFY_END	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_BDRE_FORMAT_BEGIN	BD-RE format operation started
CN_BDRE_FORMAT_END	BD-RE format operation completed
CN_UDF_FILE_LOOKUP	UDF file found during fast lookup
CN_UDF_LOOKUP_FILE	UDF file found during fast lookup
CN_UDF_LOOKUP_DIR	UDF directory found during fast lookup

2.2.64 CDB_FAILURE_INFORMATION Structure

C++

```
typedef struct _CDB_FAILURE_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__CDBSizeInUCHARs;
  UCHAR m__UCHAR__CDB[ 16 ];
  UCHAR m__UCHAR__SenseSizeInUCHARs;
  UCHAR m__UCHAR__Sense[ 32 ];
  UCHAR m__UCHAR__TransportStatus;
  UCHAR m__UCHAR__TargetStatus;
  UCHAR m__UCHAR__HostAdapterStatus;
} * PCDB_FAILURE_INFORMATION, CDB_FAILURE_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsValid;	Is this data valid (structure was really filled)
UCHAR m__UCHAR__CDBSizeInUCHARs;	CDB size in UCHARs
UCHAR m__UCHAR__CDB[16];	CDB dump
UCHAR m__UCHAR__SenseSizeInUCHARs;	SCSI sense size in UCHARs
UCHAR m__UCHAR__Sense[32];	SCSI sense dump
UCHAR m__UCHAR__TransportStatus;	SCSI transport status
UCHAR m__UCHAR__TargetStatus;	SCSI target status
UCHAR m__UCHAR__HostAdapterStatus;	SCSI host adapter status

Description

Structure that represents CDB failure information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__CDBSizeInUCHARs	CDB size in UCHARs
m__UCHAR__CDB[16]	CDB dump
m__UCHAR__SenseSizeInUCHARs	SCSI sense size in UCHARs
m__UCHAR__Sense[32]	SCSI sense dump
m__UCHAR__TransportStatus	SCSI transport status
m__UCHAR__TargetStatus	SCSI target status
m__UCHAR__HostAdapterStatus	SCSI host adapter status

2.2.65 DAO_DISC_LAYOUT Structure

C++

```
typedef struct _DAO_DISC_LAYOUT {
  LONG m__LONG__NumberOfEntries;
  DAO_DISC_LAYOUT_ENTRY m__DAO_DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];
} * PDAO_DISC_LAYOUT, DAO_DISC_LAYOUT;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of entries
`DAO_DISC_LAYOUT_ENTRY m__DAO_DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];`	DAO disc layout entries

Description

Structure that represents DAO disc layout

Member	Definition
m__LONG__NumberOfEntries	Number of entries
m__DAO_DISC_LAYOUT_ENTRY	DAO disc layout entries

2.2.66 DAO_DISC_LAYOUT_ENTRY Structure

C++

```
typedef struct _DAO_DISC_LAYOUT_ENTRY {
  LONG m__LONG__TrackSizeInLBs;
  LONG m__LONG__TrackStartingLBA;
  BOOLEAN m__BOOLEAN__IsUnicode;
  union {
    CHAR m__CHAR__TrackName[ (MAX_PATH * sizeof(WCHAR) ) ];
    WCHAR m__WCHAR__TrackName[ (MAX_PATH) ];
  }
  PVOID m__PVOID__File;
  BOOLEAN m__BOOLEAN__IsDataTrack;
  BOOLEAN m__BOOLEAN__IsRawTrack;
  BOOLEAN m__BOOLEAN__IsAudioExTrack;
} * PDAO_DISC_LAYOUT_ENTRY, DAO_DISC_LAYOUT_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__TrackSizeInLBs;`	Track size in LBs (logical blocks)
`LONG m__LONG__TrackStartingLBA;`	Track starting LBA (logical block address)
`BOOLEAN m__BOOLEAN__IsUnicode;`	Is name in Unicode format or not (see below)
`CHAR m__CHAR__TrackName[ (MAX_PATH * sizeof(WCHAR) ) ];`	Track name (absolute path & name) in ANSI format (see above)
`WCHAR m__WCHAR__TrackName[ (MAX_PATH) ];`	Track name (absolute path & name) in Unicode format (see above)
`PVOID m__PVOID__File;`	Pointer to internally created disk file object (NULL initially)
`BOOLEAN m__BOOLEAN__IsDataTrack;`	Is this data track (audio otherwise)
`BOOLEAN m__BOOLEAN__IsRawTrack;`	Is this raw track (MDF image)
`BOOLEAN m__BOOLEAN__IsAudioExTrack;`	Is this extended audio track (2448 UCHARs/logical block)

Description

Structure that represents DAO disc layout entry (track)

Member	Definition
m__LONG__TrackSizeInLBs	Track size in LBs (logical blocks)

m__LONG__TrackStartingLBA	Track starting LBA (logical block address)
m__BOOLEAN__IsUnicode	Is name in Unicode format or not (see below)
m__CHAR__TrackName	Track name (absolute path & name) in ANSI or Unicode format (see above)
m__PVOID__File	Pointer to internally created disk file object (NULL initially)
m__BOOLEAN__IsDataTrack	Is this data track (audio otherwise)
m__BOOLEAN__IsRawTrack	Is this raw track (MDF image)
m__BOOLEAN__IsAudioExTrack	Is this extended audio track (2448 UCHARs/logical block)

2.2.67 DISC_FILESYSTEM Structure

C++

```
typedef struct _DISC_FILESYSTEM {
  BOOLEAN m__BOOLEAN__UDFPresent;
  BOOLEAN m__BOOLEAN__JolietPresent;
  BOOLEAN m__BOOLEAN__ISO9660Present;
  BOOLEAN m__BOOLEAN__ElToritoBootRecordPresent;
} * PDISC_FILESYSTEM, DISC_FILESYSTEM;
```

File

StarBurn.h (see page 662)

Description

Structure with disc file system flags

Member Definition
--------- ----------
m__BOOLEAN__UDFPresent UDF file system is present
m__BOOLEAN__ISO9660Present ISO9660 file system is present
m__BOOLEAN__JolietPresent Joliet extension for ISO9660 file system is present
m__BOOLEAN__ElToritoBootRecordPresent El Torito boot record is present

2.2.68 DISC_LAYOUT Structure

C++

```
typedef struct _DISC_LAYOUT {
  LONG m__LONG__NumberOfEntries;
  DISC_LAYOUT_ENTRY m__DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];
  LONG m__LONG__NumberOfRawRawPWTracks;
  LONG m__LONG__RawRawPWTrackSizeInLBs[ NUMBER_OF_TRACKS ];
} * PDISC_LAYOUT, DISC_LAYOUT;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of entries
`DISC_LAYOUT_ENTRY m__DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];`	Disc layout entries
`LONG m__LONG__NumberOfRawRawPWTracks;`	Number of raw + raw P-W sub-channel tracks
`LONG m__LONG__RawRawPWTrackSizeInLBs[ NUMBER_OF_TRACKS ];`	Raw + raw P-W sub-channel track sizes

Description

Structure that represents disc layout

Member	Definition
m__LONG__NumberOfEntries	Number of entries
m__DISC_LAYOUT_ENTRY	Disc layout entries
m__LONG__NumberOfRawRawPWTracks	Number of raw + raw P-W sub-channel tracks
m__LONG__RawRawPWTrackSize	Raw + raw P-W sub-channel track sizes

2.2.69 DISC_LAYOUT_ENTRY Structure

C++

```
typedef struct _DISC_LAYOUT_ENTRY {
  BOOLEAN m__BOOLEAN__IsAudio;
  BOOLEAN m__BOOLEAN__IsVideo;
  BOOLEAN m__BOOLEAN__IsRawRawPW;
  LONG m__LONG__TrackSizeInLBs;
  LONG m__LONG__PreGapSizeInLBs;
  LONG m__LONG__PostGapSizeInLBs;
  union {
    CHAR m__CHAR__TrackName[ MAX_PATH*sizeof(WCHAR) ];
    WCHAR m__WCHAR__TrackName[ MAX_PATH ];
  }
  PVOID m__PVOID__File;
  PVOID m__PVOID__Tree;
  PVOID m__PVOID__BackSideTree;
} * PDISC_LAYOUT_ENTRY, DISC_LAYOUT_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsAudio;`	Is this audio track
`BOOLEAN m__BOOLEAN__IsVideo;`	Is this video track
`BOOLEAN m__BOOLEAN__IsRawRawPW;`	Is this raw + raw P-W sub-channel track
`LONG m__LONG__TrackSizeInLBs;`	Track size in LBs (logical blocks)
`LONG m__LONG__PreGapSizeInLBs;`	PreGap size in LBs (logical blocks)
`LONG m__LONG__PostGapSizeInLBs;`	PostGap size in LBs (logical blocks)
`CHAR m__CHAR__TrackName[ MAX_PATH*sizeof(WCHAR) ];`	Track name (ansi)(absolute path & name), can be zero array if next pointer is valid
`WCHAR m__WCHAR__TrackName[ MAX_PATH ];`	Track name (unicode)(absolute path & name), can be zero array if next pointer is valid
`PVOID m__PVOID__File;`	Pointer to internally created disk file object
`PVOID m__PVOID__Tree;`	Pointer to object created with StarBurn_ISO9660JolietFileTree_Create (see page 288)(), can be NULL
`PVOID m__PVOID__BackSideTree;`	Pointer to undecorated ISO9660 file tree object

Description

Structure that represents disc layout entry (track)

Member	Definition
m__BOOLEAN__IsAudio	Is this audio track
m__BOOLEAN__IsVideo	Is this video track
m__BOOLEAN__IsRawRawPW	Is this raw + raw P-W sub-channel track
m__LONG__TrackSizeInLBs	Track size in LBs (logical blocks)
m__LONG__PreGapSizeInLBs	PreGap size in LBs (logical blocks)
m__LONG__PostGapSizeInLBs	PostGap size in LBs (logical blocks)
m__CHAR__TrackName	Track name (absolute path & name), can be zero array if next pointer is valid
m__PVOID__File	Pointer to internally created disk file object
m__PVOID__Tree	Pointer to object created with StarBurn_ISO9660JolietFileTree_Create (see page 288)(), can be NULL
m__PVOID__BackSideTree	Pointer to undecorated ISO9660 file tree object

2.2.70 DISC_TYPE Enumeration

C++

```
typedef enum _DISC_TYPE {
  DISC_TYPE_UNKNOWN = 0,
  DISC_TYPE_CDROM = 1,
  DISC_TYPE_CDR = 2,
  DISC_TYPE_CDRW = 3,
  DISC_TYPE_DVDROM = 4,
  DISC_TYPE_DVDR = 5,
  DISC_TYPE_DVDRAM = 6,
  DISC_TYPE_DVDRWRO = 7,
  DISC_TYPE_DVDRWSR = 8,
  DISC_TYPE_DVDPLUSRW = 9,
  DISC_TYPE_DDCDROM = 10,
  DISC_TYPE_DDCDR = 11,
  DISC_TYPE_DDCDRW = 12,
  DISC_TYPE_DVDPLUSR = 13,
  DISC_TYPE_NO_MEDIA = 14,
  DISC_TYPE_DVDPLUSR_DL = 15,
  DISC_TYPE_DVDR_DL = 16,
  DISC_TYPE_BDROM = 17,
  DISC_TYPE_BDR = 18,
  DISC_TYPE_BDRE = 19,
  DISC_TYPE_HDDVDROM = 20,
  DISC_TYPE_HDDVDR = 21,
  DISC_TYPE_HDDVDRW = 22
} * PDISC_TYPE, DISC_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
DISC_TYPE_UNKNOWN = 0	Unknown disc type
DISC_TYPE_CDROM = 1	CD-ROM
DISC_TYPE_CDR = 2	CD-R

DISC_TYPE_CDRW = 3	CD-RW
DISC_TYPE_DVDROM = 4	DVD-ROM
DISC_TYPE_DVDR = 5	DVD-R
DISC_TYPE_DVDRAM = 6	DVD-RAM
DISC_TYPE_DVDRWRO = 7	DVD-RW RO (Restricted Overwrite)
DISC_TYPE_DVDRWSR = 8	DVD-RW SR (Sequential Recording)
DISC_TYPE_DVDPLUSRW = 9	DVD+RW
DISC_TYPE_DDCDROM = 10	DD (Double Density) CD-ROM
DISC_TYPE_DDCDR = 11	DD (Double Density) CD-R
DISC_TYPE_DDCDRW = 12	DD (Double Density) CD-RW
DISC_TYPE_DVDPLUSR = 13	DVD+R
DISC_TYPE_NO_MEDIA = 14	No media is inserted to the disc drive
DISC_TYPE_DVDPLUSR_DL = 15	DVD+R DL (Double Layer)
DISC_TYPE_DVDR_DL = 16	DVD-R DL (Dual Layer)
DISC_TYPE_BDROM = 17	BD-ROM (Blu-Ray ROM)
DISC_TYPE_BDR = 18	BD-R (Blu-Ray Disc Recordable)
DISC_TYPE_BDRE = 19	BD-RE (Blu-Ray Disc Recordable Erasable)
DISC_TYPE_HDDVDROM = 20	HD-DVD ROM
DISC_TYPE_HDDVDR = 21	HD-DVD Recordable
DISC_TYPE_HDDVDRW = 22	HD-DVD ReWritable

Description

Enum that represents inserted disc type

Member	Definition
DISC_TYPE_UNKNOWN	Unknown disc type
DISC_TYPE_CDROM,	CD-ROM
DISC_TYPE_CDR,	CD-R
DISC_TYPE_CDRW,	CD-RW
DISC_TYPE_DVDROM,	DVD-ROM
DISC_TYPE_DVDR,	DVD-R
DISC_TYPE_DVDRAM,	DVD-RAM
DISC_TYPE_DVDRWRO,	DVD-RW RO (Restricted Overwrite)
DISC_TYPE_DVDRWSR,	DVD-RW SR (Sequential Recording)
DISC_TYPE_DVDPLUSRW,	DVD+RW
DISC_TYPE_DDCDROM,	DD (Double Density) CD-ROM
DISC_TYPE_DDCDR,	DD (Double Density) CD-R
DISC_TYPE_DDCRW	DD (Double Density) CD-RW
DISC_TYPE_DVDPLUSR	DVD+R
DISC_TYPE_NO_MEDIA	No media is inserted to the disc drive
DISC_TYPE_DVDPLUSR_DL	DVD+R DL (Double Layer)
DISC_TYPE_DVDR_DL	DVD-R DL (Dual Layer)
DISC_TYPE_BDROM	BD-ROM (Blu-Ray ROM)
DISC_TYPE_BDR	BD-R (Blu-Ray Disc Recordable)
DISC_TYPE_BDRE	BD-RE (Blu-Ray Disc Recordable Erasable)
DISC_TYPE_HDDVDROM	HD-DVD ROM
DISC_TYPE_HDDVDR	HD-DVD Recordable
DISC_TYPE_HDDVDRW	HD-DVD ReWritable

2.2.71 DVD_VIDEO_CONTROL_BLOCK Structure

C++

```
typedef struct _DVD_VIDEO_CONTROL_BLOCK {
  UDF_CONTROL_BLOCK m__UDF_CONTROL_BLOCK;
  UDF_TREE_ITEM m__UDF_TREE_ITEM__Directory[ 4 ];
  UDF_TREE_ITEM m__UDF_TREE_ITEM__File[ 1192 ];
  LONG m__LONG__NumberOfFiles;
  LONG m__LONG__NumberOfTitles;
  LONG m__LONG__VtsIndex[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ][ 2 ];
  BOOLEAN m__BOOLEAN__IsMainMenuVob;
  BOOLEAN m__BOOLEAN__IsTitleMenuVob[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ];
} * PDVD_VIDEO_CONTROL_BLOCK, DVD_VIDEO_CONTROL_BLOCK;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UDF_CONTROL_BLOCK m__UDF_CONTROL_BLOCK;`	Main control block for ISO9660/UDF bridge file system
`UDF_TREE_ITEM m__UDF_TREE_ITEM__Directory[ 4 ];`	1st element of the array is not used (reserved), 2nd goes for ROOT, 3d for AUDIO_TS and 4th for VIDEO_TS directory 1 (reserved) + 1 (ROOT) + 1 (AUDIO_TS) + 1 (VIDEO_TS) = 4
`UDF_TREE_ITEM m__UDF_TREE_ITEM__File[ 1192 ];`	1st element of the array is not used (reserved), then goes 3 entries for VIDEO_TS.IFO, VIDEO_TS.VOB and VIDEO_TS.BUP. Then goes up to 99 titles each can contain up to 10 chapters and IFO and BUP files 1 (reserved) + 3 (VIDEO_TS.IFO, VIDEO_TS.VOB and VIDEO_TS.BUP) + 99 * 12 (VTS_xx_0.IFO, VTS_xx_0.VOB .. VTS_xx_9.VOB + VTS_xx_0.BUP) = 1 + 3 + 1188 = 1192
`LONG m__LONG__NumberOfFiles;`	Number of files really used in m__UDF_TREE_ITEM__File[ ] array
`LONG m__LONG__NumberOfTitles;`	Number of titles in DVD-Video compilation
`LONG m__LONG__VtsIndex[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ][ 2 ];`	Indexes for VTS_xx_0.IFO and VTS_xx_0.BUP for every title
`BOOLEAN m__BOOLEAN__IsMainMenuVob;`	Is VIDEO_TS.VOB file present (does whole movie has a main menu)
`BOOLEAN m__BOOLEAN__IsTitleMenuVob[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ];`	Is VTS_xx_0.VOB file present (does title set has a menu)

Description

Structure that represents DVD-Video control block and pointer to DVD-Video control block

2.2.72 ERASE_TYPE Enumeration

C++

```
typedef enum _ERASE_TYPE {
  ERASE_TYPE_BLANK_DISC_FULL = 0,
  ERASE_TYPE_BLANK_DISC_FAST,
  ERASE_TYPE_BLANK_TRACK,
  ERASE_TYPE_UNRESERVE_TRACK,
  ERASE_TYPE_BLANK_TRACK_TAIL,
  ERASE_TYPE_UNCLOSE_LAST_SESSION,
  ERASE_TYPE_BLANK_SESSION
} * PERASE_TYPE, ERASE_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
ERASE_TYPE_BLANK_DISC_FULL = 0	Completely erase the rewritable disc
ERASE_TYPE_BLANK_DISC_FAST	Erase only TOC, PMA and first track lead in on the rewritable disc
ERASE_TYPE_BLANK_TRACK	Erase track only
ERASE_TYPE_UNRESERVE_TRACK	Unreserve track that was reserved before
ERASE_TYPE_BLANK_TRACK_TAIL	Erase track tail
ERASE_TYPE_UNCLOSE_LAST_SESSION	Unclose last closed session
ERASE_TYPE_BLANK_SESSION	Erase last session

Description

Enum that represents erase type

Member	Definition
ERASE_TYPE_BLANK_DISC_FULL	Completely erase the rewritable disc
ERASE_TYPE_BLANK_DISC_FAST	Erase only TOC, PMA and first track lead in on the rewritable disc
ERASE_TYPE_BLANK_TRACK	Erase track only
ERASE_TYPE_UNRESERVE_TRACK	Unreserve track that was reserved before
ERASE_TYPE_BLANK_TRACK_TAIL	Erase track tail
ERASE_TYPE_UNCLOSE_LAST_SESSION	Unclose last closed session
ERASE_TYPE_BLANK_SESSION	Erase last session

2.2.73 EXCEPTION_NUMBER Enumeration

C++

```
typedef enum _EXCEPTION_NUMBER {
  EN_SUCCESS = 0,
  EN_FAILURE,
  EN_INVALID_INPUT_PARAMETER,
  EN_INVALID_STATE,
  EN_MEMORY_ALLOCATION_FAILED,
  EN_SYSTEM_CALL_FAILED,
  EN_SCSI_TRANSPORT_FAILED,
  EN_SCSI_DEVICE_BUSY,
  EN_SCSI_CDB_FAILED,
  EN_SCSI_DEVICE_INVALID_TYPE,
  EN_INVALID_RESPONSE,
  EN_BUFFER_UNDERRUN,
  EN_INVALID_EXCEPTION,
  EN_ACCESS_TO_FEATURE_DENIED,
  EN_USER_EXCEPTION,
  EN_PATH_TOO_LONG,
  EN_UNDER_CONSTRUCTION,
  EN_NOT_FOUND,
  EN_FILE_TOO_BIG,
  EN_NOT_IMPLEMENTED,
  EN_RANGE,
  EN_REGISTRATION_FAILED,
  EN_UNSUPPORTED_AUDIO,
  EN_BUFFER_TOO_SMALL,
  EN_SYSTEM_CALL_FAILED_EX,
  EN_ERROR_RECOVERY_FAILED,
  EN_UNRECOGNIZED_MEDIA,
  EN_GENERAL_SEEK_ERROR,
  EN_GENERAL_READ_ERROR,
```

```
    EN_ILLEGAL_OPERATION_FOR_TRACK,
    EN_UNSUPPORTED_READ_MODE,
    EN_REQUEST_TOO_LARGE,
    EN_FULL_ERASE_REQUIRED,
    EN_VERIFY_FAILED,
    EN_DPM_FAILED,
    EN_DEVICE_SHARING_VIOLATION,
    EN_CALL_IS_OBSOLETE,
    EN_WRONG_OS_VERSION,
    EN_INVALID_FIELD_IN_CDB,
    EN_CACHE_IS_TOO_SMALL
  } * PEXCEPTION_NUMBER, EXCEPTION_NUMBER;
```

File

StarBurn.h (see page 662)

Members

Members	Description
EN_SUCCESS = 0	Nothing really happened, operation completed successfully
EN_FAILURE	Undefined error happened, something goes really wrong
EN_INVALID_INPUT_PARAMETER	User input parameter is not valid
EN_INVALID_STATE	The state of the object is not valid for current operation
EN_MEMORY_ALLOCATION_FAILED	Memory allocation failed
EN_SYSTEM_CALL_FAILED	System call failed, check SystemError pointer to system error value
EN_SCSI_TRANSPORT_FAILED	SCSI transport internal error
EN_SCSI_DEVICE_BUSY	SCSI device is busy for a while
EN_SCSI_CDB_FAILED	SCSI CDB delivery failed, check CDB_FAILURE_INFORMATION (see page 480) for details
EN_SCSI_DEVICE_INVALID_TYPE	SCSI device of this type is not supported
EN_INVALID_RESPONSE	Something really unsupported returned
EN_BUFFER_UNDERRUN	Buffer underrun happened
EN_INVALID_EXCEPTION	Exception was not allocated
EN_ACCESS_TO_FEATURE_DENIED	Access to feature denied b/s of the wrong version
EN_USER_EXCEPTION	This is not a real exception just the result of user interaction
EN_PATH_TOO_LONG	File path is too long
EN_UNDER_CONSTRUCTION	This feature is still under construction
EN_NOT_FOUND	Operation could not be performed b/s either device or requested parameter not found
EN_FILE_TOO_BIG	File is too big for the requested operation
EN_NOT_IMPLEMENTED	Feature is not implemented yet
EN_RANGE	Passed range is not valid
EN_REGISTRATION_FAILED	Registration procedure completed with errors
EN_UNSUPPORTED_AUDIO	Unsupported audio format used as either input or output
EN_BUFFER_TOO_SMALL	Buffer size supplied by caller is not sufficient
EN_SYSTEM_CALL_FAILED_EX	Middle-layer error happened, check SystemError pointer for specific error value
EN_ERROR_RECOVERY_FAILED	Error recovery failed
EN_UNRECOGNIZED_MEDIA	Current media type is not recognized
EN_GENERAL_SEEK_ERROR	General seek error on CD/DVD/Blu-Ray/HD-DVD media
EN_GENERAL_READ_ERROR	General read error on CD/DVD/Blu-Ray/HD-DVD media
EN_ILLEGAL_OPERATION_FOR_TRACK	Illegal operation for track
EN_UNSUPPORTED_READ_MODE	Currently selected read mode is not supported by device
EN_REQUEST_TOO_LARGE	Request size is too large to be handled
EN_FULL_ERASE_REQUIRED	Full erase required before recording to inserted media
EN_VERIFY_FAILED	Verify operation found different memory buffers
EN_DPM_FAILED	DPM associated call failed
EN_DEVICE_SHARING_VIOLATION	Device failed to open b/s of sharing violation
EN_CALL_IS_OBSOLETE	Function is obsolete
EN_WRONG_OS_VERSION	OS version is not supported
EN_INVALID_FIELD_IN_CDB	Invalid field in CDB
EN_CACHE_IS_TOO_SMALL	Allocated cache size is too small to handle "write" operations

Description

Enum that represents exception number

Member	Definition
EN_SUCCESS	Nothing really happened, operation completed successfully
EN_FAILURE	Undefined error happened, something goes really wrong
EN_INVALID_INPUT_PARAMETER	User input parameter is not valid
EN_INVALID_STATE	The state of the object is not valid for current operation
EN_MEMORY_ALLOCATION_FAILED	Memory allocation failed
EN_SYSTEM_CALL_FAILED	System call failed, check SystemError pointer to system error value
EN_SCSI_TRANSPORT_FAILED	SCSI transport internal error
EN_SCSI_DEVICE_BUSY	SCSI device is busy for a while
EN_SCSI_CDB_FAILED	SCSI CDB delivery failed, check CDB_FAILURE_INFORMATION (see page 480) for details
EN_SCSI_DEVICE_INVALID_TYPE	SCSI device of this type is not supported
EN_INVALID_RESPONSE	Something really unsupported returned
EN_BUFFER_UNDERRUN	Buffer underrun happened
EN_INVALID_EXCEPTION	Exception was not allocated
EN_ACCESS_TO_FEATURE_DENIED	Access to feature denied b/s of the wrong version
EN_USER_EXCEPTION	This is not a real exception just the result of user interaction
EN_PATH_TOO_LONG	File path is too long
EN_UNDER_CONSTRUCTION	This feature is still under construction
EN_NOT_FOUND	Operation could not be performed b/s either device or requested parameter not found
EN_FILE_TOO_BIG	File is too big for requested operation
EN_NOT_IMPLEMENTED	Feature is not implemented yet
EN_RANGE	Passed range is not valid
EN_REGISTRATION_FAILED	Registration procedure completed with errors
EN_UNSUPPORTED_AUDIO	Unsupported audio format used as either input or output
EN_BUFFER_TOO_SMALL	Buffer size supplied by caller is not sufficient
EN_SYSTEM_CALL_FAILED_EX	Middle-layer error happened, check SystemError pointer for specific error value
EN_ERROR_RECOVERY_FAILED	Error recovery failed
EN_UNRECOGNIZED_MEDIA	Current media type is not recognized
EN_GENERAL_SEEK_ERROR	General seek error on CD/DVD/Blu-Ray/HD-DVD media
EN_GENERAL_READ_ERROR	General read error on CD/DVD/Blu-Ray/HD-DVD media
EN_ILLEGAL_OPERATION_FOR_TRACK	Illegal operation for track
EN_UNSUPPORTED_READ_MODE	Currently selected read mode is not supported by device
EN_REQUEST_TOO_LARGE	Request size is too large to be handled
EN_FULL_ERASE_REQUIRED	Full erase required before recording to inserted media
EN_VERIFY_FAILED	Verify operation found different memory buffers
EN_DPM_FAILED	DPM associated call failed
EN_DEVICE_SHARING_VIOLATION	Device failed to open b/s of sharing violation

EN_CALL_IS_OBSOLETE	Function is obsolete
EN_WRONG_OS_VERSION	OS version is not supported
EN_INVALID_FIELD_IN_CDB	Invalid field in CDB
EN_CACHE_IS_TOO_SMALL	Allocated cache size is too small to handle "write" operations

2.2.74 FILE_TIME Enumeration

C++

```
typedef enum _FILE_TIME {
  FILE_TIME_CREATION = 0,
  FILE_TIME_LAST_ACCESS,
  FILE_TIME_LAST_WRITE
} * PFILE_TIME, FILE_TIME;
```

File

StarBurn.h (see page 662)

Members

Members	Description
FILE_TIME_CREATION = 0	Original creation file time will be included into the file system image
FILE_TIME_LAST_ACCESS	Last access time will be included into the file system image
FILE_TIME_LAST_WRITE	Last write tile will be included into the file system image

Description

Enum that represents file time that will be included in the ISO9660/Joliet image

Member	Definition
FILE_TIME_CREATION	Original creation file time will be included into the file system image
FILE_TIME_LAST_ACCESS	Last access time will be included into the file system image
FILE_TIME_LAST_WRITE	Last write tile will be included into the file system image

2.2.75 FILE_TREE Enumeration

C++

```
typedef enum _FILE_TREE {
  FILE_TREE_ISO9660 = 0,
  FILE_TREE_JOLIET
} * PFILE_TREE, FILE_TREE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
FILE_TREE_ISO9660 = 0	ISO9660 file tree
FILE_TREE_JOLIET	ISO9660 file tree + Joliet extensions

Description

Enum that represents file tree

Member	Definition
FILE_TREE_ISO9660	ISO9660 file tree
FILE_TREE_JOLIET	ISO9660 file tree + Joliet extensions

2.2.76 FULL_TOC_ENTRY_RAW Structure

C++

```
typedef struct _FULL_TOC_ENTRY_RAW {
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__CONTROL : 4;
  UCHAR m__UCHAR__ADR : 4;
  UCHAR m__UCHAR__TNO;
  UCHAR m__UCHAR__POINT;
  UCHAR m__UCHAR__Min;
  UCHAR m__UCHAR__Sec;
  UCHAR m__UCHAR__Frame;
  UCHAR m__UCHAR__Zero;
  UCHAR m__UCHAR__PMIN;
  UCHAR m__UCHAR__PSEC;
  UCHAR m__UCHAR__PFRAME;
} * PFULL_TOC_ENTRY_RAW, FULL_TOC_ENTRY_RAW;
```

File

StarBurn.h (see page 662)

Members

Members	Description
UCHAR m__UCHAR__SessionNumber;	Session number
UCHAR m__UCHAR__CONTROL : 4;	Control
UCHAR m__UCHAR__ADR : 4;	Address
UCHAR m__UCHAR__TNO;	Track number
UCHAR m__UCHAR__POINT;	Point
UCHAR m__UCHAR__Min;	Minute
UCHAR m__UCHAR__Sec;	Second
UCHAR m__UCHAR__Frame;	Frame
UCHAR m__UCHAR__Zero;	Zero (always 0)
UCHAR m__UCHAR__PMIN;	Starting minute (from MSF)
UCHAR m__UCHAR__PSEC;	Starting second (from MSF)
UCHAR m__UCHAR__PFRAME;	Starting frame (from MSF)

Description

Structure that represents full TOC entry

Member	Definition
m__UCHAR__SessionNumber	Session number
m__UCHAR__CONTROL : 4	Control
m__UCHAR__ADR : 4	Address
m__UCHAR__TNO	Track number
m__UCHAR__POINT	Point
m__UCHAR__Min	Minute
m__UCHAR__Sec	Second

m__UCHAR__Frame	Frame
m__UCHAR__Zero	Zero (always 0)
m__UCHAR__PMIN	Starting minute (from MSF)
m__UCHAR__PSEC	Starting second (from MSF)
m__UCHAR__PFRAME	Starting frame (from MSF)

2.2.77 ISO9660_DATE_TIME Structure

C++

```
typedef struct _ISO9660_DATE_TIME {
  UCHAR m__UCHAR__Year;
  UCHAR m__UCHAR__Month;
  UCHAR m__UCHAR__Day;
  UCHAR m__UCHAR__Hour;
  UCHAR m__UCHAR__Minute;
  UCHAR m__UCHAR__Second;
  UCHAR m__UCHAR__GMTOffset;
} * PISO9660_DATE_TIME, ISO9660_DATE_TIME;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UCHAR m__UCHAR__Year;`	Year
`UCHAR m__UCHAR__Month;`	Month
`UCHAR m__UCHAR__Day;`	Day
`UCHAR m__UCHAR__Hour;`	Hour
`UCHAR m__UCHAR__Minute;`	Minute
`UCHAR m__UCHAR__Second;`	Second
`UCHAR m__UCHAR__GMTOffset;`	Offset from GMT in hours

Description

Structure that represents ISO9660 date and time

Member	Definition
m__UCHAR__Year	Year
m__UCHAR__Month	Month
M__UCHAR__Day	Day
m__UCHAR__Hour	Hour
m__UCHAR__Minute	Minute
m__UCHAR__Second	Second
m__UCHAR__GMTOffset	Offset from GMT in hours

2.2.78 ISO9660_KIDTYPE Enumeration

C++

```
typedef enum _ISO9660_KIDTYPE {
  KIDTYPE_UNKNOWN,
  KIDTYPE_IMPORTED,
  KIDTYPE_FROMDISK,
  KIDTYPE_VIRTUAL
} * PISO9660_KIDTYPE, ISO9660_KIDTYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
KIDTYPE_UNKNOWN	Unknown kid type
KIDTYPE_IMPORTED	Node was imported from previous track
KIDTYPE_FROMDISK	Node was added from the disk
KIDTYPE_VIRTUAL	Node was created as memory file or virtual directory

Description

Enum that represent type of ISO9660 tree node

Member	Definition
KIDTYPE_UNKNOWN	Unknown kid type
KIDTYPE_IMPORTED	Node was imported from previous track
KIDTYPE_FROMDISK	Node was added from the disk
KIDTYPE_VIRTUAL	Node was created as memory file or virtual directory

2.2.79 NAME_COLLISION_INFO Structure

C++

```
typedef struct _NAME_COLLISION_INFO {
  BOOLEAN m__BOOLEAN__IsJolietName;
  UCHAR m__UCHAR__Name[ MAX_PATH * 2 ];
  PCHAR m__PCHAR__OldAbsoluteName;
  ULONG m__ULONG__OldFileAttributes;
  PCHAR m__PCHAR__NewAbsoluteName;
  ULONG m__ULONG__NewFileAttributes;
  NAME_COLLISION_SOLUTION m__NAME_COLLISION_SOLUTION;
} * PNAME_COLLISION_INFO, NAME_COLLISION_INFO;
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsJolietName;	Indicates the format of the m__UCHAR__Name
UCHAR m__UCHAR__Name[MAX_PATH * 2];	The node name (CHAR for ISO9660 or USHORT for Joliet)
PCHAR m__PCHAR__OldAbsoluteName;	Absolute file name of the old file node
ULONG m__ULONG__OldFileAttributes;	Old file attributes

PCHAR m__PCHAR__NewAbsoluteName;	Absolute file name of the new file node
ULONG m__ULONG__NewFileAttributes;	New file attributes
NAME_COLLISION_SOLUTION m__NAME_COLLISION_SOLUTION;	The solution of the collision (what action has been performed)

Description

Structure that represents information about name collision

Member	Definition
m__BOOLEAN__IsJolietName	Indicates the format of the m__UCHAR__Name
m__UCHAR__Name	The node name (CHAR for ISO9660 or USHORT for Joliet)
m__PCHAR__OldAbsoluteName	Absolute file name of the old file node
m__ULONG__OldFileAttributes	Old file attributes
m__PCHAR__NewAbsoluteName	Absolute file name of the new file node
m__ULONG__NewFileAttributes	New file attributes
m__NAME_COLLISION_SOLUTION	The solution of the collision (what action has been performed)

2.2.80 NAME_COLLISION_SOLUTION Enumeration

C++

```
typedef enum _NAME_COLLISION_SOLUTION {
  NAME_COLLISION_SKIP_NEW = 0,
  NAME_COLLISION_REPLACE_OLD = 1,
  NAME_COLLISION_RENAME_NEW = 2,
  NAME_COLLISION_MERGE_FOLDERS = 3,
  NAME_COLLISION_FORCE_DWORD = 0xFFFFFFFF
} * PNAME_COLLISION_SOLUTION, NAME_COLLISION_SOLUTION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
NAME_COLLISION_SKIP_NEW = 0	Remove new node and leave old node
NAME_COLLISION_REPLACE_OLD = 1	Remove old node and insert new node
NAME_COLLISION_RENAME_NEW = 2	Leave old node and insert new node with the new name
NAME_COLLISION_MERGE_FOLDERS = 3	Add files from new folder into an old folder
NAME_COLLISION_FORCE_DWORD = 0xFFFFFFFF	Force type size to 4 bytes

Description

Enum that represents the solutions of name collisions

Member	Definition
NAME_COLLISION_SKIP_NEW	Remove new node and leave old node
NAME_COLLISION_REPLACE_OLD	Remove old node and insert new node
NAME_COLLISION_RENAME_NEW	Leave old node and insert new node with the new name
NAME_COLLISION_MERGE_FOLDERS	Add files from new folder into an old folder

2.2.81 PCALLBACK_NUMBER Enumeration

C++

```
typedef enum _CALLBACK_NUMBER {
  CN_FILE_TREE_PROGRESS_ADD = 0,
  CN_FILE_TREE_PROGRESS_REMOVE,
  CN_FILE_TREE_PROGRESS_IGNORE,
  CN_FILE_TREE_PROGRESS_NAME_COLLISION,
  CN_TARGET_FILE_ANALYZE_BEGIN,
  CN_TARGET_FILE_ANALYZE_END,
  CN_WAIT_CACHE_FULL_BEGIN,
  CN_WAIT_CACHE_FULL_END,
  CN_SYNCHRONIZE_CACHE_BEGIN,
  CN_SYNCHRONIZE_CACHE_END,
  CN_FIND_DEVICE,
  CN_CDVD_READ_PROGRESS,
  CN_CDVD_WRITE_PROGRESS,
  CN_CDVD_BUFFER_STATUS,
  CN_CDVD_TRACK_BEGIN,
  CN_CDVD_TRACK_END,
  CN_CDVD_SPLIT_BEGIN,
  CN_CDVD_SPLIT_END,
  CN_CDVD_READ_BAD_BLOCK_HIT,
  CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT,
  CN_CDVD_READ_RETRY,
  CN_DVDPLUSRW_FORMAT_BEGIN,
  CN_DVDPLUSRW_FORMAT_END,
  CN_DVDRAM_FORMAT_BEGIN,
  CN_DVDRAM_FORMAT_END,
  CN_BUFFER_UNDERRUN,
  CN_DVD_MEDIA_PADDING_SIZE,
  CN_DVD_MEDIA_PADDING_BEGIN,
  CN_DVD_MEDIA_PADDING_END,
  CN_CDVD_READ_CANCEL_QUERY,
  CN_DVDRW_QUICK_FORMAT_BEGIN,
  CN_DVDRW_QUICK_FORMAT_END,
  CN_DVD_TEST_WRITE_DISABLED,
  CN_CDVD_DPM_BEGIN,
  CN_CDVD_DPM_END,
  CN_CDVD_DPM_PROGRESS,
  CN_CDVD_VERIFY_PROGRESS,
  CN_SAO_TRACK_WRITE_BEGIN,
  CN_SAO_TRACK_WRITE_END,
  CN_DVD_MEDIA_PADDING_WRITE_PROGRESS,
  CN_CDVD_WRITE_BEGIN,
  CN_CDVD_WRITE_END,
  CN_CDVD_VERIFY_BEGIN,
  CN_CDVD_VERIFY_END,
  CN_BDRE_FORMAT_BEGIN,
  CN_BDRE_FORMAT_END,
  CN_UDF_FILE_LOOKUP,
  CN_UDF_LOOKUP_FILE,
  CN_UDF_LOOKUP_DIR,
  CN_CDVD_BLANKAREA_PROGRESS
} * PCALLBACK_NUMBER, CALLBACK_NUMBER;
```

File

StarBurn.h (see page 662)

Members

Members	Description
CN_FILE_TREE_PROGRESS_ADD = 0	Item was add to the file tree
CN_FILE_TREE_PROGRESS_REMOVE	Item was removed from the file tree
CN_FILE_TREE_PROGRESS_IGNORE	Item was ignored during processing file tree

CN_FILE_TREE_PROGRESS_NAME_COLLISION	There are two nodes which have the same name
CN_TARGET_FILE_ANALYZE_BEGIN	File internal structure (size, type etc) analyze started
CN_TARGET_FILE_ANALYZE_END	File structure analyze completed
CN_WAIT_CACHE_FULL_BEGIN	Toolkit started to wait for cache to become full
CN_WAIT_CACHE_FULL_END	Toolkit finished to wait for cache fullness
CN_SYNCHRONIZE_CACHE_BEGIN	Cache flushing to the media started
CN_SYNCHRONIZE_CACHE_END	Cache flushing completed
CN_FIND_DEVICE	Find device operation completed
CN_CDVD_READ_PROGRESS	CD/DVD/Blu-Ray/HD-DVD read operation progress
CN_CDVD_WRITE_PROGRESS	CD/DVD/Blu-Ray/HD-DVD write operation progress
CN_CDVD_BUFFER_STATUS	CD/DVD/Blu-Ray/HD-DVD buffer status information queried
CN_CDVD_TRACK_BEGIN	CD/DVD/Blu-Ray/HD-DVD track processing started
CN_CDVD_TRACK_END	CD/DVD/Blu-Ray/HD-DVD track processing completed
CN_CDVD_SPLIT_BEGIN	CD/DVD/Blu-Ray/HD-DVD split section processing started
CN_CDVD_SPLIT_END	CD/DVD/Blu-Ray/HD-DVD split section processing completed
CN_CDVD_READ_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a bad (unrecoverable) block
CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a ECC/EDC bad (recoverable) block
CN_CDVD_READ_RETRY	CD/DVD/Blu-Ray/HD-DVD read operation was retried
CN_DVDPLUSRW_FORMAT_BEGIN	DVD+RW format operation started
CN_DVDPLUSRW_FORMAT_END	DVD+RW format operation completed
CN_DVDRAM_FORMAT_BEGIN	DVD-RAM format operation started
CN_DVDRAM_FORMAT_END	DVD-RAM format operation completed
CN_BUFFER_UNDERRUN	Buffer underrun condition happened
CN_DVD_MEDIA_PADDING_SIZE	DVD media would be padded to 1GB size, additional info passed
CN_DVD_MEDIA_PADDING_BEGIN	DVD media padding burn process started
CN_DVD_MEDIA_PADDING_END	DVD media padding burn process completed
CN_CDVD_READ_CANCEL_QUERY	CD/DVD/Blu-Ray/HD-DVD media processing plug-in queries cancel status
CN_DVDRW_QUICK_FORMAT_BEGIN	DVD-RW quick format operation started
CN_DVDRW_QUICK_FORMAT_END	DVD-RW quick format operation completed
CN_DVD_TEST_WRITE_DISABLED	Test write is disabled (for DVD+R/RW, DVD-RAM or multisession DVD-RW)
CN_CDVD_DPM_BEGIN	CD/DVD/Blu-Ray/HD-DVD DPM processing started
CN_CDVD_DPM_END	CD/DVD/Blu-Ray/HD-DVD DPM processing completed
CN_CDVD_DPM_PROGRESS	CD/DVD/Blu-Ray/HD-DVD DPM processing progress
CN_CDVD_VERIFY_PROGRESS	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_SAO_TRACK_WRITE_BEGIN	SAO track write started
CN_SAO_TRACK_WRITE_END	SAO track write completed
CN_DVD_MEDIA_PADDING_WRITE_PROGRESS	DVD media padding burn progress indication
CN_CDVD_WRITE_BEGIN	CD/DVD/Blu-Ray/HD-DVD write operation started
CN_CDVD_WRITE_END	CD/DVD/Blu-Ray/HD-DVD write operation completed
CN_CDVD_VERIFY_BEGIN	CD/DVD/Blu-Ray/HD-DVD verify operation started
CN_CDVD_VERIFY_END	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_BDRE_FORMAT_BEGIN	BD-RE format operation started
CN_BDRE_FORMAT_END	BD-RE format operation completed
CN_UDF_FILE_LOOKUP	UDF file found during fast lookup
CN_UDF_LOOKUP_FILE	UDF file found during fast lookup
CN_UDF_LOOKUP_DIR	UDF directory found during fast lookup
CN_CDVD_BLANKAREA_PROGRESS	CD/DVD/Blu-Ray/HD-DVD blank area operation progress

Description

Enum that represents callback number

Member	Definition
CN_FILE_TREE_PROGRESS_ADD	Item was add to the file tree
CN_FILE_TREE_PROGRESS_REMOVE	Item was removed from the file tree
CN_FILE_TREE_PROGRESS_IGNORE	Item was ignored during processing file tree

CN_FILE_TREE_PROGRESS_NAME...	There are two nodes which have the same name
CN_TARGET_FILE_ANALYZE_BEGIN	File internal structure (size, type etc) analyze started
CN_TARGET_FILE_ANALYZE_END	File structure analyze completed
CN_WAIT_CACHE_FULL_BEGIN	Toolkit started to wait for cache to become full
CN_WAIT_CACHE_FULL_END	Toolkit finished to wait for cache fullness
CN_SYNCHRONIZE_CACHE_BEGIN	Cache flushing to the media started
CN_SYNCHRONIZE_CACHE_END	Cache flushing to the media completed
CN_FIND_DEVICE	Find device operation completed
CD_CDVD_READ_PROGRESS	CD/DVD/Blu-Ray/HD-DVD read operation progress
CN_CDVD_WRITE_PROGRESS	CD/DVD/Blu-Ray/HD-DVD write operation progress
CN_CDVD_BUFFER_STATUS	CD/DVD/Blu-Ray/HD-DVD buffer status information queried
CN_CDVD_TRACK_BEGIN	CD/DVD/Blu-Ray/HD-DVD track processing started
CN_CDVD_TRACK_END	CD/DVD/Blu-Ray/HD-DVD track processing completed
CN_CDVD_SPLIT_BEGIN	CD/DVD/Blu-Ray/HD-DVD split section processing started
CN_CDVD_SPLIT_END	CD/DVD/Blu-Ray/HD-DVD split section processing completed
CN_CDVD_READ_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a bad (unrecoverable) block
CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT	CD/DVD/Blu-Ray/HD-DVD read operation had hit a ECC/EDC bad (recoverable) block
CN_CDVD_READ_RETRY	CD/DVD/Blu-Ray/HD-DVD read operation was retried
CN_DVDPLUSRW_FORMAT_BEGIN	DVD+RW format operation started
CN_DVDPLUSRW_FORMAT_EDN	DVD+RW format operation completed
CN_DVDRAM_FORMAT_BEGIN	DVD-RAM format operation started
CN_DVDRAM_FORMAT_EDN	DVD-RAM format operation completed
CN_BUFFER_UNDERRUN	Buffer underrun condition happened
CN_DVD_MEDIA_PADDING_SIZE	DVD media would be padded to 1GB size, additional info passed
CN_DVD_MEDIA_PADDING_BEGIN	DVD media padding burn process started
CN_DVD_MEDIA_PADDING_END	DVD media padding burn process completed
CN_CDVD_READ_CANCEL_QUERY	CD/DVD/Blu-Ray/HD-DVD media processing plug-in queries cancel status
CN_DVDRW_QUICK_FORMAT_BEGIN	DVD-RW quick format operation started
CN_DVDRW_QUICK_FORMAT_END	DVD-RW quick format operation completed
CN_DVD_TEST_WRITE_DISABLED	Test write is disabled (for DVD+R/RW, DVD-RAM or multisession DVD-RW)
CN_CDVD_DPM_BEGIN	CD/DVD/Blu-Ray/HD-DVD DPM processing started
CN_CDVD_DPM_END	CD/DVD/Blu-Ray/HD-DVD DPM processing completed
CD_CDVD_DPM_PROGRESS	CD/DVD/Blu-Ray/HD-DVD DPM operation progress
CN_CDVD_VERIFY_PROGRESS	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_SAO_TRACK_WRITE_BEGIN	SAO track write started
CN_SAO_TRACK_WRITE_END	SAO track write completed
CN_DVD_MEDIA_PADDING_WRITE_PRO...	DVD media padding burn progress indication
CN_CDVD_WRITE_BEGIN	CD/DVD/Blu-Ray/HD-DVD write operation started
CN_CDVD_WRITE_END	CD/DVD/Blu-Ray/HD-DVD write operation completed

CN_CDVD_VERIFY_BEGIN	CD/DVD/Blu-Ray/HD-DVD verify operation started
CN_CDVD_VERIFY_END	CD/DVD/Blu-Ray/HD-DVD verify operation completed
CN_BDRE_FORMAT_BEGIN	BD-RE format operation started
CN_BDRE_FORMAT_END	BD-RE format operation completed
CN_UDF_FILE_LOOKUP	UDF file found during fast lookup
CN_UDF_LOOKUP_FILE	UDF file found during fast lookup
CN_UDF_LOOKUP_DIR	UDF directory found during fast lookup

2.2.82 PCDB_FAILURE_INFORMATION Structure

C++

```
typedef struct _CDB_FAILURE_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__CDBSizeInUCHARs;
  UCHAR m__UCHAR__CDB[ 16 ];
  UCHAR m__UCHAR__SenseSizeInUCHARs;
  UCHAR m__UCHAR__Sense[ 32 ];
  UCHAR m__UCHAR__TransportStatus;
  UCHAR m__UCHAR__TargetStatus;
  UCHAR m__UCHAR__HostAdapterStatus;
} * PCDB_FAILURE_INFORMATION, CDB_FAILURE_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__CDBSizeInUCHARs;`	CDB size in UCHARs
`UCHAR m__UCHAR__CDB[ 16 ];`	CDB dump
`UCHAR m__UCHAR__SenseSizeInUCHARs;`	SCSI sense size in UCHARs
`UCHAR m__UCHAR__Sense[ 32 ];`	SCSI sense dump
`UCHAR m__UCHAR__TransportStatus;`	SCSI transport status
`UCHAR m__UCHAR__TargetStatus;`	SCSI target status
`UCHAR m__UCHAR__HostAdapterStatus;`	SCSI host adapter status

Description

Structure that represents CDB failure information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__CDBSizeInUCHARs	CDB size in UCHARs
m__UCHAR__CDB[16]	CDB dump
m__UCHAR__SenseSizeInUCHARs	SCSI sense size in UCHARs
m__UCHAR__Sense[32]	SCSI sense dump
m__UCHAR__TransportStatus	SCSI transport status
m__UCHAR__TargetStatus	SCSI target status
m__UCHAR__HostAdapterStatus	SCSI host adapter status

2.2.83 PDAO_DISC_LAYOUT Structure

C++

```
typedef struct _DAO_DISC_LAYOUT {
  LONG m__LONG__NumberOfEntries;
  DAO_DISC_LAYOUT_ENTRY m__DAO_DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];
} * PDAO_DISC_LAYOUT, DAO_DISC_LAYOUT;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of entries
`DAO_DISC_LAYOUT_ENTRY m__DAO_DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];`	DAO disc layout entries

Description

Structure that represents DAO disc layout

Member	Definition
m__LONG__NumberOfEntries	Number of entries
m__DAO_DISC_LAYOUT_ENTRY	DAO disc layout entries

2.2.84 PDAO_DISC_LAYOUT_ENTRY Structure

C++

```
typedef struct _DAO_DISC_LAYOUT_ENTRY {
  LONG m__LONG__TrackSizeInLBs;
  LONG m__LONG__TrackStartingLBA;
  BOOLEAN m__BOOLEAN__IsUnicode;
  union {
    CHAR m__CHAR__TrackName[ (MAX_PATH * sizeof(WCHAR) ) ];
    WCHAR m__WCHAR__TrackName[ (MAX_PATH) ];
  }
  PVOID m__PVOID__File;
  BOOLEAN m__BOOLEAN__IsDataTrack;
  BOOLEAN m__BOOLEAN__IsRawTrack;
  BOOLEAN m__BOOLEAN__IsAudioExTrack;
} * PDAO_DISC_LAYOUT_ENTRY, DAO_DISC_LAYOUT_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__TrackSizeInLBs;`	Track size in LBs (logical blocks)
`LONG m__LONG__TrackStartingLBA;`	Track starting LBA (logical block address)
`BOOLEAN m__BOOLEAN__IsUnicode;`	Is name in Unicode format or not (see below)
`CHAR m__CHAR__TrackName[ (MAX_PATH * sizeof(WCHAR) ) ];`	Track name (absolute path & name) in ANSI format (see above)
`WCHAR m__WCHAR__TrackName[ (MAX_PATH) ];`	Track name (absolute path & name) in Unicode format (see above)
`PVOID m__PVOID__File;`	Pointer to internally created disk file object (NULL initially)
`BOOLEAN m__BOOLEAN__IsDataTrack;`	Is this data track (audio otherwise)

BOOLEAN m__BOOLEAN__IsRawTrack;	Is this raw track (MDF image)
BOOLEAN m__BOOLEAN__IsAudioExTrack;	Is this extended audio track (2448 UCHARs/logical block)

Description

Structure that represents DAO disc layout entry (track)

Member	Definition
m__LONG__TrackSizeInLBs	Track size in LBs (logical blocks)
m__LONG__TrackStartingLBA	Track starting LBA (logical block address)
m__BOOLEAN__IsUnicode	Is name in Unicode format or not (see below)
m__CHAR__TrackName	Track name (absolute path & name) in ANSI or Unicode format (see above)
m__PVOID__File	Pointer to internally created disk file object (NULL initially)
m__BOOLEAN__IsDataTrack	Is this data track (audio otherwise)
m__BOOLEAN__IsRawTrack	Is this raw track (MDF image)
m__BOOLEAN__IsAudioExTrack	Is this extended audio track (2448 UCHARs/logical block)

2.2.85 PDISC_FILESYSTEM Structure

C++

```
typedef struct _DISC_FILESYSTEM {
  BOOLEAN m__BOOLEAN__UDFPresent;
  BOOLEAN m__BOOLEAN__JolietPresent;
  BOOLEAN m__BOOLEAN__ISO9660Present;
  BOOLEAN m__BOOLEAN__ElToritoBootRecordPresent;
} * PDISC_FILESYSTEM, DISC_FILESYSTEM;
```

File

StarBurn.h (see page 662)

Description

Structure with disc file system flags

Member Definition
--------- ----------
m__BOOLEAN__UDFPresent UDF file system is present
m__BOOLEAN__ISO9660Present ISO9660 file system is present
m__BOOLEAN__JolietPresent Joliet extension for ISO9660 file system is present
m__BOOLEAN__ElToritoBootRecordPresent El Torito boot record is present

2.2.86 PDISC_LAYOUT Structure

C++

```
typedef struct _DISC_LAYOUT {
  LONG m__LONG__NumberOfEntries;
```

```
  DISC_LAYOUT_ENTRY m__DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];
  LONG m__LONG__NumberOfRawRawPWTracks;
  LONG m__LONG__RawRawPWTrackSizeInLBs[ NUMBER_OF_TRACKS ];
} * PDISC_LAYOUT, DISC_LAYOUT;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of entries
`DISC_LAYOUT_ENTRY m__DISC_LAYOUT_ENTRY[ NUMBER_OF_TRACKS ];`	Disc layout entries
`LONG m__LONG__NumberOfRawRawPWTracks;`	Number of raw + raw P-W sub-channel tracks
`LONG m__LONG__RawRawPWTrackSizeInLBs[ NUMBER_OF_TRACKS ];`	Raw + raw P-W sub-channel track sizes

Description

Structure that represents disc layout

Member	Definition
m__LONG__NumberOfEntries	Number of entries
m__DISC_LAYOUT_ENTRY	Disc layout entries
m__LONG__NumberOfRawRawPWTracks	Number of raw + raw P-W sub-channel tracks
m__LONG__RawRawPWTrackSize	Raw + raw P-W sub-channel track sizes

2.2.87 PDISC_LAYOUT_ENTRY Structure

C++

```
typedef struct _DISC_LAYOUT_ENTRY {
  BOOLEAN m__BOOLEAN__IsAudio;
  BOOLEAN m__BOOLEAN__IsVideo;
  BOOLEAN m__BOOLEAN__IsRawRawPW;
  LONG m__LONG__TrackSizeInLBs;
  LONG m__LONG__PreGapSizeInLBs;
  LONG m__LONG__PostGapSizeInLBs;
  union {
    CHAR m__CHAR__TrackName[ MAX_PATH*sizeof(WCHAR) ];
    WCHAR m__WCHAR__TrackName[ MAX_PATH ];
  }
  PVOID m__PVOID__File;
  PVOID m__PVOID__Tree;
  PVOID m__PVOID__BackSideTree;
} * PDISC_LAYOUT_ENTRY, DISC_LAYOUT_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsAudio;`	Is this audio track
`BOOLEAN m__BOOLEAN__IsVideo;`	Is this video track
`BOOLEAN m__BOOLEAN__IsRawRawPW;`	Is this raw + raw P-W sub-channel track
`LONG m__LONG__TrackSizeInLBs;`	Track size in LBs (logical blocks)
`LONG m__LONG__PreGapSizeInLBs;`	PreGap size in LBs (logical blocks)
`LONG m__LONG__PostGapSizeInLBs;`	PostGap size in LBs (logical blocks)

`CHAR m__CHAR__TrackName[ MAX_PATH*sizeof(WCHAR) ];`	Track name (ansi)(absolute path & name), can be zero array if next pointer is valid
`WCHAR m__WCHAR__TrackName[ MAX_PATH ];`	Track name (unicode)(absolute path & name), can be zero array if next pointer is valid
`PVOID m__PVOID__File;`	Pointer to internally created disk file object
`PVOID m__PVOID__Tree;`	Pointer to object created with StarBurn_ISO9660JolietFileTree_Create (see page 288)(), can be NULL
`PVOID m__PVOID__BackSideTree;`	Pointer to undecorated ISO9660 file tree object

Description

Structure that represents disc layout entry (track)

Member	Definition
m__BOOLEAN__IsAudio	Is this audio track
m__BOOLEAN__IsVideo	Is this video track
m__BOOLEAN__IsRawRawPW	Is this raw + raw P-W sub-channel track
m__LONG__TrackSizeInLBs	Track size in LBs (logical blocks)
m__LONG__PreGapSizeInLBs	PreGap size in LBs (logical blocks)
m__LONG__PostGapSizeInLBs	PostGap size in LBs (logical blocks)
m__CHAR__TrackName	Track name (absolute path & name), can be zero array if next pointer is valid
m__PVOID__File	Pointer to internally created disk file object
m__PVOID__Tree	Pointer to object created with StarBurn_ISO9660JolietFileTree_Create (see page 288)(), can be NULL
m__PVOID__BackSideTree	Pointer to undecorated ISO9660 file tree object

2.2.88 PDISC_TYPE Enumeration

C++

```
typedef enum _DISC_TYPE {
  DISC_TYPE_UNKNOWN = 0,
  DISC_TYPE_CDROM = 1,
  DISC_TYPE_CDR = 2,
  DISC_TYPE_CDRW = 3,
  DISC_TYPE_DVDROM = 4,
  DISC_TYPE_DVDR = 5,
  DISC_TYPE_DVDRAM = 6,
  DISC_TYPE_DVDRWRO = 7,
  DISC_TYPE_DVDRWSR = 8,
  DISC_TYPE_DVDPLUSRW = 9,
  DISC_TYPE_DDCDROM = 10,
  DISC_TYPE_DDCDR = 11,
  DISC_TYPE_DDCDRW = 12,
  DISC_TYPE_DVDPLUSR = 13,
  DISC_TYPE_NO_MEDIA = 14,
  DISC_TYPE_DVDPLUSR_DL = 15,
  DISC_TYPE_DVDR_DL = 16,
  DISC_TYPE_BDROM = 17,
  DISC_TYPE_BDR = 18,
  DISC_TYPE_BDRE = 19,
  DISC_TYPE_HDDVDROM = 20,
  DISC_TYPE_HDDVDR = 21,
  DISC_TYPE_HDDVDRW = 22
} * PDISC_TYPE, DISC_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
DISC_TYPE_UNKNOWN = 0	Unknown disc type
DISC_TYPE_CDROM = 1	CD-ROM
DISC_TYPE_CDR = 2	CD-R
DISC_TYPE_CDRW = 3	CD-RW
DISC_TYPE_DVDROM = 4	DVD-ROM
DISC_TYPE_DVDR = 5	DVD-R
DISC_TYPE_DVDRAM = 6	DVD-RAM
DISC_TYPE_DVDRWRO = 7	DVD-RW RO (Restricted Overwrite)
DISC_TYPE_DVDRWSR = 8	DVD-RW SR (Sequential Recording)
DISC_TYPE_DVDPLUSRW = 9	DVD+RW
DISC_TYPE_DDCDROM = 10	DD (Double Density) CD-ROM
DISC_TYPE_DDCDR = 11	DD (Double Density) CD-R
DISC_TYPE_DDCDRW = 12	DD (Double Density) CD-RW
DISC_TYPE_DVDPLUSR = 13	DVD+R
DISC_TYPE_NO_MEDIA = 14	No media is inserted to the disc drive
DISC_TYPE_DVDPLUSR_DL = 15	DVD+R DL (Double Layer)
DISC_TYPE_DVDR_DL = 16	DVD-R DL (Dual Layer)
DISC_TYPE_BDROM = 17	BD-ROM (Blu-Ray ROM)
DISC_TYPE_BDR = 18	BD-R (Blu-Ray Disc Recordable)
DISC_TYPE_BDRE = 19	BD-RE (Blu-Ray Disc Recordable Erasable)
DISC_TYPE_HDDVDROM = 20	HD-DVD ROM
DISC_TYPE_HDDVDR = 21	HD-DVD Recordable
DISC_TYPE_HDDVDRW = 22	HD-DVD ReWritable

Description

Enum that represents inserted disc type

Member	Definition
DISC_TYPE_UNKNOWN	Unknown disc type
DISC_TYPE_CDROM,	CD-ROM
DISC_TYPE_CDR,	CD-R
DISC_TYPE_CDRW,	CD-RW
DISC_TYPE_DVDROM,	DVD-ROM
DISC_TYPE_DVDR,	DVD-R
DISC_TYPE_DVDRAM,	DVD-RAM
DISC_TYPE_DVDRWRO,	DVD-RW RO (Restricted Overwrite)
DISC_TYPE_DVDRWSR,	DVD-RW SR (Sequential Recording)
DISC_TYPE_DVDPLUSRW,	DVD+RW
DISC_TYPE_DDCDROM,	DD (Double Density) CD-ROM
DISC_TYPE_DDCDR,	DD (Double Density) CD-R
DISC_TYPE_DDCRW	DD (Double Density) CD-RW
DISC_TYPE_DVDPLUSR	DVD+R
DISC_TYPE_NO_MEDIA	No media is inserted to the disc drive
DISC_TYPE_DVDPLUSR_DL	DVD+R DL (Double Layer)
DISC_TYPE_DVDR_DL	DVD-R DL (Dual Layer)

DISC_TYPE_BDROM	BD-ROM (Blu-Ray ROM)
DISC_TYPE_BDR	BD-R (Blu-Ray Disc Recordable)
DISC_TYPE_BDRE	BD-RE (Blu-Ray Disc Recordable Erasable)
DISC_TYPE_HDDVDROM	HD-DVD ROM
DISC_TYPE_HDDVDR	HD-DVD Recordable
DISC_TYPE_HDDVDRW	HD-DVD ReWritable

2.2.89 PDVD_VIDEO_CONTROL_BLOCK Structure

C++

```
typedef struct _DVD_VIDEO_CONTROL_BLOCK {
  UDF_CONTROL_BLOCK m__UDF_CONTROL_BLOCK;
  UDF_TREE_ITEM m__UDF_TREE_ITEM__Directory[ 4 ];
  UDF_TREE_ITEM m__UDF_TREE_ITEM__File[ 1192 ];
  LONG m__LONG__NumberOfFiles;
  LONG m__LONG__NumberOfTitles;
  LONG m__LONG__VtsIndex[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ][ 2 ];
  BOOLEAN m__BOOLEAN__IsMainMenuVob;
  BOOLEAN m__BOOLEAN__IsTitleMenuVob[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ];
} * PDVD_VIDEO_CONTROL_BLOCK, DVD_VIDEO_CONTROL_BLOCK;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UDF_CONTROL_BLOCK m__UDF_CONTROL_BLOCK;`	Main control block for ISO9660/UDF bridge file system
`UDF_TREE_ITEM m__UDF_TREE_ITEM__Directory[ 4 ];`	1st element of the array is not used (reserved), 2nd goes for ROOT, 3d for AUDIO_TS and 4th for VIDEO_TS directory 1 (reserved) + 1 (ROOT) + 1 (AUDIO_TS) + 1 (VIDEO_TS) = 4
`UDF_TREE_ITEM m__UDF_TREE_ITEM__File[ 1192 ];`	1st element of the array is not used (reserved), then goes 3 entries for VIDEO_TS.IFO, VIDEO_TS.VOB and VIDEO_TS.BUP. Then goes up to 99 titles each can contain up to 10 chapters and IFO and BUP files 1 (reserved) + 3 (VIDEO_TS.IFO, VIDEO_TS.VOB and VIDEO_TS.BUP) + 99 * 12 (VTS_xx_0.IFO, VTS_xx_0.VOB .. VTS_xx_9.VOB + VTS_xx_0.BUP) = 1 + 3 + 1188 = 1192
`LONG m__LONG__NumberOfFiles;`	Number of files really used in m__UDF_TREE_ITEM__File[ ] array
`LONG m__LONG__NumberOfTitles;`	Number of titles in DVD-Video compilation
`LONG m__LONG__VtsIndex[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ][ 2 ];`	Indexes for VTS_xx_0.IFO and VTS_xx_0.BUP for every title
`BOOLEAN m__BOOLEAN__IsMainMenuVob;`	Is VIDEO_TS.VOB file present (does whole movie has a main menu)
`BOOLEAN m__BOOLEAN__IsTitleMenuVob[ STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES ];`	Is VTS_xx_0.VOB file present (does title set has a menu)

Description

Structure that represents DVD-Video control block and pointer to DVD-Video control block

2.2.90 PERASE_TYPE Enumeration

C++

```
typedef enum _ERASE_TYPE {
  ERASE_TYPE_BLANK_DISC_FULL = 0,
  ERASE_TYPE_BLANK_DISC_FAST,
  ERASE_TYPE_BLANK_TRACK,
```

```
  ERASE_TYPE_UNRESERVE_TRACK,
  ERASE_TYPE_BLANK_TRACK_TAIL,
  ERASE_TYPE_UNCLOSE_LAST_SESSION,
  ERASE_TYPE_BLANK_SESSION
} * PERASE_TYPE, ERASE_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
ERASE_TYPE_BLANK_DISC_FULL = 0	Completely erase the rewritable disc
ERASE_TYPE_BLANK_DISC_FAST	Erase only TOC, PMA and first track lead in on the rewritable disc
ERASE_TYPE_BLANK_TRACK	Erase track only
ERASE_TYPE_UNRESERVE_TRACK	Unreserve track that was reserved before
ERASE_TYPE_BLANK_TRACK_TAIL	Erase track tail
ERASE_TYPE_UNCLOSE_LAST_SESSION	Unclose last closed session
ERASE_TYPE_BLANK_SESSION	Erase last session

Description

Enum that represents erase type

Member	Definition
ERASE_TYPE_BLANK_DISC_FULL	Completely erase the rewritable disc
ERASE_TYPE_BLANK_DISC_FAST	Erase only TOC, PMA and first track lead in on the rewritable disc
ERASE_TYPE_BLANK_TRACK	Erase track only
ERASE_TYPE_UNRESERVE_TRACK	Unreserve track that was reserved before
ERASE_TYPE_BLANK_TRACK_TAIL	Erase track tail
ERASE_TYPE_UNCLOSE_LAST_SESSION	Unclose last closed session
ERASE_TYPE_BLANK_SESSION	Erase last session

2.2.91 PEXCEPTION_NUMBER Enumeration

C++

```
typedef enum _EXCEPTION_NUMBER {
  EN_SUCCESS = 0,
  EN_FAILURE,
  EN_INVALID_INPUT_PARAMETER,
  EN_INVALID_STATE,
  EN_MEMORY_ALLOCATION_FAILED,
  EN_SYSTEM_CALL_FAILED,
  EN_SCSI_TRANSPORT_FAILED,
  EN_SCSI_DEVICE_BUSY,
  EN_SCSI_CDB_FAILED,
  EN_SCSI_DEVICE_INVALID_TYPE,
  EN_INVALID_RESPONSE,
  EN_BUFFER_UNDERRUN,
  EN_INVALID_EXCEPTION,
  EN_ACCESS_TO_FEATURE_DENIED,
  EN_USER_EXCEPTION,
  EN_PATH_TOO_LONG,
  EN_UNDER_CONSTRUCTION,
  EN_NOT_FOUND,
  EN_FILE_TOO_BIG,
  EN_NOT_IMPLEMENTED,
  EN_RANGE,
```

```
  EN_REGISTRATION_FAILED,
  EN_UNSUPPORTED_AUDIO,
  EN_BUFFER_TOO_SMALL,
  EN_SYSTEM_CALL_FAILED_EX,
  EN_ERROR_RECOVERY_FAILED,
  EN_UNRECOGNIZED_MEDIA,
  EN_GENERAL_SEEK_ERROR,
  EN_GENERAL_READ_ERROR,
  EN_ILLEGAL_OPERATION_FOR_TRACK,
  EN_UNSUPPORTED_READ_MODE,
  EN_REQUEST_TOO_LARGE,
  EN_FULL_ERASE_REQUIRED,
  EN_VERIFY_FAILED,
  EN_DPM_FAILED,
  EN_DEVICE_SHARING_VIOLATION,
  EN_CALL_IS_OBSOLETE,
  EN_WRONG_OS_VERSION,
  EN_INVALID_FIELD_IN_CDB,
  EN_CACHE_IS_TOO_SMALL
} * PEXCEPTION_NUMBER, EXCEPTION_NUMBER;
```

File

StarBurn.h (see page 662)

Members

Members	Description
EN_SUCCESS = 0	Nothing really happened, operation completed successfully
EN_FAILURE	Undefined error happened, something goes really wrong
EN_INVALID_INPUT_PARAMETER	User input parameter is not valid
EN_INVALID_STATE	The state of the object is not valid for current operation
EN_MEMORY_ALLOCATION_FAILED	Memory allocation failed
EN_SYSTEM_CALL_FAILED	System call failed, check SystemError pointer to system error value
EN_SCSI_TRANSPORT_FAILED	SCSI transport internal error
EN_SCSI_DEVICE_BUSY	SCSI device is busy for a while
EN_SCSI_CDB_FAILED	SCSI CDB delivery failed, check CDB_FAILURE_INFORMATION (see page 480) for details
EN_SCSI_DEVICE_INVALID_TYPE	SCSI device of this type is not supported
EN_INVALID_RESPONSE	Something really unsupported returned
EN_BUFFER_UNDERRUN	Buffer underrun happened
EN_INVALID_EXCEPTION	Exception was not allocated
EN_ACCESS_TO_FEATURE_DENIED	Access to feature denied b/s of the wrong version
EN_USER_EXCEPTION	This is not a real exception just the result of user interaction
EN_PATH_TOO_LONG	File path is too long
EN_UNDER_CONSTRUCTION	This feature is still under construction
EN_NOT_FOUND	Operation could not be performed b/s either device or requested parameter not found
EN_FILE_TOO_BIG	File is too big for the requested operation
EN_NOT_IMPLEMENTED	Feature is not implemented yet
EN_RANGE	Passed range is not valid
EN_REGISTRATION_FAILED	Registration procedure completed with errors
EN_UNSUPPORTED_AUDIO	Unsupported audio format used as either input or output
EN_BUFFER_TOO_SMALL	Buffer size supplied by caller is not sufficient
EN_SYSTEM_CALL_FAILED_EX	Middle-layer error happened, check SystemError pointer for specific error value
EN_ERROR_RECOVERY_FAILED	Error recovery failed
EN_UNRECOGNIZED_MEDIA	Current media type is not recognized
EN_GENERAL_SEEK_ERROR	General seek error on CD/DVD/Blu-Ray/HD-DVD media
EN_GENERAL_READ_ERROR	General read error on CD/DVD/Blu-Ray/HD-DVD media
EN_ILLEGAL_OPERATION_FOR_TRACK	Illegal operation for track
EN_UNSUPPORTED_READ_MODE	Currently selected read mode is not supported by device
EN_REQUEST_TOO_LARGE	Request size is too large to be handled
EN_FULL_ERASE_REQUIRED	Full erase required before recording to inserted media
EN_VERIFY_FAILED	Verify operation found different memory buffers
EN_DPM_FAILED	DPM associated call failed

EN_DEVICE_SHARING_VIOLATION	Device failed to open b/s of sharing violation
EN_CALL_IS_OBSOLETE	Function is obsolete
EN_WRONG_OS_VERSION	OS version is not supported
EN_INVALID_FIELD_IN_CDB	Invalid field in CDB
EN_CACHE_IS_TOO_SMALL	Allocated cache size is too small to handle "write" operations

Description

Enum that represents exception number

Member	Definition
EN_SUCCESS	Nothing really happened, operation completed successfully
EN_FAILURE	Undefined error happened, something goes really wrong
EN_INVALID_INPUT_PARAMETER	User input parameter is not valid
EN_INVALID_STATE	The state of the object is not valid for current operation
EN_MEMORY_ALLOCATION_FAILED	Memory allocation failed
EN_SYSTEM_CALL_FAILED	System call failed, check SystemError pointer to system error value
EN_SCSI_TRANSPORT_FAILED	SCSI transport internal error
EN_SCSI_DEVICE_BUSY	SCSI device is busy for a while
EN_SCSI_CDB_FAILED	SCSI CDB delivery failed, check CDB_FAILURE_INFORMATION (see page 480) for details
EN_SCSI_DEVICE_INVALID_TYPE	SCSI device of this type is not supported
EN_INVALID_RESPONSE	Something really unsupported returned
EN_BUFFER_UNDERRUN	Buffer underrun happened
EN_INVALID_EXCEPTION	Exception was not allocated
EN_ACCESS_TO_FEATURE_DENIED	Access to feature denied b/s of the wrong version
EN_USER_EXCEPTION	This is not a real exception just the result of user interaction
EN_PATH_TOO_LONG	File path is too long
EN_UNDER_CONSTRUCTION	This feature is still under construction
EN_NOT_FOUND	Operation could not be performed b/s either device or requested parameter not found
EN_FILE_TOO_BIG	File is too big for requested operation
EN_NOT_IMPLEMENTED	Feature is not implemented yet
EN_RANGE	Passed range is not valid
EN_REGISTRATION_FAILED	Registration procedure completed with errors
EN_UNSUPPORTED_AUDIO	Unsupported audio format used as either input or output
EN_BUFFER_TOO_SMALL	Buffer size supplied by caller is not sufficient
EN_SYSTEM_CALL_FAILED_EX	Middle-layer error happened, check SystemError pointer for specific error value
EN_ERROR_RECOVERY_FAILED	Error recovery failed
EN_UNRECOGNIZED_MEDIA	Current media type is not recognized
EN_GENERAL_SEEK_ERROR	General seek error on CD/DVD/Blu-Ray/HD-DVD media
EN_GENERAL_READ_ERROR	General read error on CD/DVD/Blu-Ray/HD-DVD media
EN_ILLEGAL_OPERATION_FOR_TRACK	Illegal operation for track
EN_UNSUPPORTED_READ_MODE	Currently selected read mode is not supported by device
EN_REQUEST_TOO_LARGE	Request size is too large to be handled

EN_FULL_ERASE_REQUIRED	Full erase required before recording to inserted media
EN_VERIFY_FAILED	Verify operation found different memory buffers
EN_DPM_FAILED	DPM associated call failed
EN_DEVICE_SHARING_VIOLATION	Device failed to open b/s of sharing violation
EN_CALL_IS_OBSOLETE	Function is obsolete
EN_WRONG_OS_VERSION	OS version is not supported
EN_INVALID_FIELD_IN_CDB	Invalid field in CDB
EN_CACHE_IS_TOO_SMALL	Allocated cache size is too small to handle "write" operations

2.2.92 PFILE_TIME Enumeration

C++

```
typedef enum _FILE_TIME {
  FILE_TIME_CREATION = 0,
  FILE_TIME_LAST_ACCESS,
  FILE_TIME_LAST_WRITE
} * PFILE_TIME, FILE_TIME;
```

File

StarBurn.h (see page 662)

Members

Members	Description
FILE_TIME_CREATION = 0	Original creation file time will be included into the file system image
FILE_TIME_LAST_ACCESS	Last access time will be included into the file system image
FILE_TIME_LAST_WRITE	Last write tile will be included into the file system image

Description

Enum that represents file time that will be included in the ISO9660/Joliet image

Member	Definition
FILE_TIME_CREATION	Original creation file time will be included into the file system image
FILE_TIME_LAST_ACCESS	Last access time will be included into the file system image
FILE_TIME_LAST_WRITE	Last write tile will be included into the file system image

2.2.93 PFILE_TREE Enumeration

C++

```
typedef enum _FILE_TREE {
  FILE_TREE_ISO9660 = 0,
  FILE_TREE_JOLIET
} * PFILE_TREE, FILE_TREE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
FILE_TREE_ISO9660 = 0	ISO9660 file tree
FILE_TREE_JOLIET	ISO9660 file tree + Joliet extensions

Description

Enum that represents file tree

Member	Definition
FILE_TREE_ISO9660	ISO9660 file tree
FILE_TREE_JOLIET	ISO9660 file tree + Joliet extensions

2.2.94 PFULL_TOC_ENTRY_RAW Structure

C++

```
typedef struct _FULL_TOC_ENTRY_RAW {
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__CONTROL : 4;
  UCHAR m__UCHAR__ADR : 4;
  UCHAR m__UCHAR__TNO;
  UCHAR m__UCHAR__POINT;
  UCHAR m__UCHAR__Min;
  UCHAR m__UCHAR__Sec;
  UCHAR m__UCHAR__Frame;
  UCHAR m__UCHAR__Zero;
  UCHAR m__UCHAR__PMIN;
  UCHAR m__UCHAR__PSEC;
  UCHAR m__UCHAR__PFRAME;
} * PFULL_TOC_ENTRY_RAW, FULL_TOC_ENTRY_RAW;
```

File

StarBurn.h (see page 662)

Members

Members	Description
UCHAR m__UCHAR__SessionNumber;	Session number
UCHAR m__UCHAR__CONTROL : 4;	Control
UCHAR m__UCHAR__ADR : 4;	Address
UCHAR m__UCHAR__TNO;	Track number
UCHAR m__UCHAR__POINT;	Point
UCHAR m__UCHAR__Min;	Minute
UCHAR m__UCHAR__Sec;	Second
UCHAR m__UCHAR__Frame;	Frame
UCHAR m__UCHAR__Zero;	Zero (always 0)
UCHAR m__UCHAR__PMIN;	Starting minute (from MSF)
UCHAR m__UCHAR__PSEC;	Starting second (from MSF)
UCHAR m__UCHAR__PFRAME;	Starting frame (from MSF)

Description

Structure that represents full TOC entry

Member	Definition
m__UCHAR__SessionNumber	Session number

m__UCHAR__CONTROL : 4	Control
m__UCHAR__ADR : 4	Address
m__UCHAR__TNO	Track number
m__UCHAR__POINT	Point
m__UCHAR__Min	Minute
m__UCHAR__Sec	Second
m__UCHAR__Frame	Frame
m__UCHAR__Zero	Zero (always 0)
m__UCHAR__PMIN	Starting minute (from MSF)
m__UCHAR__PSEC	Starting second (from MSF)
m__UCHAR__PFRAME	Starting frame (from MSF)

2.2.95 PISO9660_DATE_TIME Structure

C++

```
typedef struct _ISO9660_DATE_TIME {
  UCHAR m__UCHAR__Year;
  UCHAR m__UCHAR__Month;
  UCHAR m__UCHAR__Day;
  UCHAR m__UCHAR__Hour;
  UCHAR m__UCHAR__Minute;
  UCHAR m__UCHAR__Second;
  UCHAR m__UCHAR__GMTOffset;
} * PISO9660_DATE_TIME, ISO9660_DATE_TIME;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UCHAR m__UCHAR__Year;`	Year
`UCHAR m__UCHAR__Month;`	Month
`UCHAR m__UCHAR__Day;`	Day
`UCHAR m__UCHAR__Hour;`	Hour
`UCHAR m__UCHAR__Minute;`	Minute
`UCHAR m__UCHAR__Second;`	Second
`UCHAR m__UCHAR__GMTOffset;`	Offset from GMT in hours

Description

Structure that represents ISO9660 date and time

Member	Definition
m__UCHAR__Year	Year
m__UCHAR__Month	Month
M__UCHAR__Day	Day
m__UCHAR__Hour	Hour
m__UCHAR__Minute	Minute
m__UCHAR__Second	Second

m__UCHAR__GMTOffset	Offset from GMT in hours

2.2.96 PISO9660_KIDTYPE Enumeration

C++

```
typedef enum _ISO9660_KIDTYPE {
  KIDTYPE_UNKNOWN,
  KIDTYPE_IMPORTED,
  KIDTYPE_FROMDISK,
  KIDTYPE_VIRTUAL
} * PISO9660_KIDTYPE, ISO9660_KIDTYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
KIDTYPE_UNKNOWN	Unknown kid type
KIDTYPE_IMPORTED	Node was imported from previous track
KIDTYPE_FROMDISK	Node was added from the disk
KIDTYPE_VIRTUAL	Node was created as memory file or virtual directory

Description

Enum that represent type of ISO9660 tree node

Member	Definition
KIDTYPE_UNKNOWN	Unknown kid type
KIDTYPE_IMPORTED	Node was imported from previous track
KIDTYPE_FROMDISK	Node was added from the disk
KIDTYPE_VIRTUAL	Node was created as memory file or virtual directory

2.2.97 PNAME_COLLISION_INFO Structure

C++

```
typedef struct _NAME_COLLISION_INFO {
  BOOLEAN m__BOOLEAN__IsJolietName;
  UCHAR m__UCHAR__Name[ MAX_PATH * 2 ];
  PCHAR m__PCHAR__OldAbsoluteName;
  ULONG m__ULONG__OldFileAttributes;
  PCHAR m__PCHAR__NewAbsoluteName;
  ULONG m__ULONG__NewFileAttributes;
  NAME_COLLISION_SOLUTION m__NAME_COLLISION_SOLUTION;
} * PNAME_COLLISION_INFO, NAME_COLLISION_INFO;
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsJolietName;	Indicates the format of the m__UCHAR__Name

`UCHAR m__UCHAR__Name[ MAX_PATH * 2 ];`	The node name (CHAR for ISO9660 or USHORT for Joliet)
`PCHAR m__PCHAR__OldAbsoluteName;`	Absolute file name of the old file node
`ULONG m__ULONG__OldFileAttributes;`	Old file attributes
`PCHAR m__PCHAR__NewAbsoluteName;`	Absolute file name of the new file node
`ULONG m__ULONG__NewFileAttributes;`	New file attributes
`NAME_COLLISION_SOLUTION m__NAME_COLLISION_SOLUTION;`	The solution of the collision (what action has been performed)

Description

Structure that represents information about name collision

Member	Definition
m__BOOLEAN__IsJolietName	Indicates the format of the m__UCHAR__Name
m__UCHAR__Name	The node name (CHAR for ISO9660 or USHORT for Joliet)
m__PCHAR__OldAbsoluteName	Absolute file name of the old file node
m__ULONG__OldFileAttributes	Old file attributes
m__PCHAR__NewAbsoluteName	Absolute file name of the new file node
m__ULONG__NewFileAttributes	New file attributes
m__NAME_COLLISION_SOLUTION	The solution of the collision (what action has been performed)

2.2.98 PNAME_COLLISION_SOLUTION Enumeration

C++

```
typedef enum _NAME_COLLISION_SOLUTION {
  NAME_COLLISION_SKIP_NEW = 0,
  NAME_COLLISION_REPLACE_OLD = 1,
  NAME_COLLISION_RENAME_NEW = 2,
  NAME_COLLISION_MERGE_FOLDERS = 3,
  NAME_COLLISION_FORCE_DWORD = 0xFFFFFFFF
} * PNAME_COLLISION_SOLUTION, NAME_COLLISION_SOLUTION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`NAME_COLLISION_SKIP_NEW = 0`	Remove new node and leave old node
`NAME_COLLISION_REPLACE_OLD = 1`	Remove old node and insert new node
`NAME_COLLISION_RENAME_NEW = 2`	Leave old node and insert new node with the new name
`NAME_COLLISION_MERGE_FOLDERS = 3`	Add files from new folder into an old folder
`NAME_COLLISION_FORCE_DWORD = 0xFFFFFFFF`	Force type size to 4 bytes

Description

Enum that represents the solutions of name collisions

Member	Definition
NAME_COLLISION_SKIP_NEW	Remove new node and leave old node
NAME_COLLISION_REPLACE_OLD	Remove old node and insert new node
NAME_COLLISION_RENAME_NEW	Leave old node and insert new node with the new name
NAME_COLLISION_MERGE_FOLDERS	Add files from new folder into an old folder

2.2.99 PPQ_SUBCHANNEL Structure

C++

```
typedef struct _PQ_SUBCHANNEL {
  UCHAR m__UCHAR__ADR : 4;
  UCHAR m__UCHAR__CONTROL : 4;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__IndexNumber;
  UCHAR m__UCHAR__Min;
  UCHAR m__UCHAR__Sec;
  UCHAR m__UCHAR__Frame;
  UCHAR m__UCHAR__ZERO;
  UCHAR m__UCHAR__AMIN;
  UCHAR m__UCHAR__ASEC;
  UCHAR m__UCHAR__AFRAME;
  UCHAR m__UCHAR__CRC1_Reserved1;
  UCHAR m__UCHAR__CRC2_Reserved2;
  UCHAR m__UCHAR__Reserved3[ 3 ];
  UCHAR m__UCHAR__Reserved4 : 7;
  UCHAR m__UCHAR__PSubChannel : 1;
} * PPQ_SUBCHANNEL, PQ_SUBCHANNEL;
```

File

StarBurn.h (see page 662)

Members

Members	Description
UCHAR m__UCHAR__ADR : 4;	Track ADR
UCHAR m__UCHAR__CONTROL : 4;	Track CONTROL
UCHAR m__UCHAR__TrackNumber;	Track Number
UCHAR m__UCHAR__IndexNumber;	Index Number
UCHAR m__UCHAR__Min;	Min (from the beginning of the track)
UCHAR m__UCHAR__Sec;	Sec (from the beginning of the track)
UCHAR m__UCHAR__Frame;	Frame (from the beginning of the track)
UCHAR m__UCHAR__ZERO;	MBZ
UCHAR m__UCHAR__AMIN;	Min (from the beginning of the disc)
UCHAR m__UCHAR__ASEC;	Sec (from the beginning of the disc)
UCHAR m__UCHAR__AFRAME;	Frame (from the beginning of the disc)
UCHAR m__UCHAR__CRC1_Reserved1;	CRC1 or MBZ
UCHAR m__UCHAR__CRC2_Reserved2;	CRC2 or MBZ
UCHAR m__UCHAR__Reserved3[3];	MBZ
UCHAR m__UCHAR__Reserved4 : 7;	MBZ
UCHAR m__UCHAR__PSubChannel : 1;	P sub-channel bit

Description

Structure that represents formatted PQ sub-channel

Member	Definition
m__UCHAR__ADR	Track ADR
m__UCHAR__CONTROL	Track CONTROL
m__UCHAR__TrackNumber	Track Number
m__UCHAR__IndexNumber	Index Number
m__UCHAR__Min	Min (from the beginning of the track)
m__UCHAR__Sec	Sec (from the beginning of the track)

m__UCHAR__Frame	Frame (from the beginning of the track)
m__UCHAR__ZERO	MBZ
m__UCHAR__AMIN	Min (from the beginning of the disc)
m__UCHAR__ASEC	Sec (from the beginning of the disc)
m__UCHAR__AFRAME	Frame (from the beginning of the disc)
m__UCHAR__CRC1_Reserved1	CRC1 or MBZ
m__UCHAR__CRC2_Reserved2	CRC2 or MBZ
m__UCHAR__Reserved3[3]	MBZ
m__UCHAR__Reserved4	MBZ
m__UCHAR__PSubChannel	P sub-channel bit

2.2.100 PPSTARBURN_BDRE_FORMAT_PROFILE Structure

C++

```
typedef struct _STARBURN_BDRE_FORMAT_PROFILE {
  LONG m__LONG__FormatProfileNumber;
  ULONG m__ULONG__NumberOfBlocksOrUserAreaDataSize;
  UCHAR m__UCHAR__CertificationType : 2;
  UCHAR m__UCHAR__FormatType : 6;
  ULONG m__ULONG__SpareAreaSizeInClustersOrBlockLength;
} ** PPSTARBURN_BDRE_FORMAT_PROFILE, * PSTARBURN_BDRE_FORMAT_PROFILE,
STARBURN_BDRE_FORMAT_PROFILE;
```

File

StarBurn.h (see page 662)

Description

Structure that represents format profile for BD-RE media

Member Definition
--------- ----------
LONG m__LONG__FormatProfileNumber Format profile number
UCHAR m__ULONG__NumberOfBlocksOrUserAreaDataSize Number of logical block(s) or user area data size
UCHAR m__UCHAR__CertificationType Certification type
UCHAR m__UCHAR__FormatType Format type
UCHAR m__ULONG__SpareAreaSizeInClustersOrBlockLength Spare area size in cluster(s) or block length

2.2.101 PPSTARBURN_STARWAVE_CALLBACK_REASON Enumeration

C++

```
typedef enum _STARBURN_STARWAVE_CALLBACK_REASON {
  STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0,
  STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS
} STARBURN_STARWAVE_CALLBACK_REASON, * PSTARBURN_STARWAVE_CALLBACK_REASON, **
PPSTARBURN_STARWAVE_CALLBACK_REASON;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

Description

StarWave callback reasons we'll be using

Member	Definition
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

2.2.102 PPSTARBURN_STARWAVE_COMPRESSION Enumeration

C++

```
typedef enum _STARBURN_STARWAVE_COMPRESSION {
  STARBURN_STARWAVE_COMPRESSION_NONE = 0,
  STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000
} STARBURN_STARWAVE_COMPRESSION, * PSTARBURN_STARWAVE_COMPRESSION, **
PPSTARBURN_STARWAVE_COMPRESSION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

Member	Definition
STARBURN_STARWAVE_COMPRESSION_NONE	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSS...	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98	WMA compressed stream, VBR at Quality 98
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K	WMA compressed stream, CBR at 32 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K	WMA compressed stream, CBR at 320 Kbps

2.2.103 PPSTARPORT_DEVICE_LIST Structure

C++

```
typedef struct _STARPORT_DEVICE_LIST {
  LONG m__LONG__NumberOfEntries;
  STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];
} ** PPSTARPORT_DEVICE_LIST, * PSTARPORT_DEVICE_LIST, STARPORT_DEVICE_LIST;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of StarPort device list entries
`STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];`	StarPort device list entries

Description

Structure that represents StarPort device list

Member	Definition
m__LONG__NumberOfEntries	Number of StarPort device list entries
m__STARPORT_DEVICE_LIST_ENTRY	StarPort device list entries

2.2.104 PPSTARPORT_DEVICE_LIST_ENTRY Structure

C++

```
typedef struct _STARPORT_DEVICE_LIST_ENTRY {
  LONG m__LONG__Index;
  LONG m__LONG__TargetId;
  CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];
} ** PPSTARPORT_DEVICE_LIST_ENTRY, * PSTARPORT_DEVICE_LIST_ENTRY,
STARPORT_DEVICE_LIST_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__Index;`	StarPort device index
`LONG m__LONG__TargetId;`	StarPort device TargetId
`CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];`	StarPort device name

Description

Structure that represents StarPort device list entry

Member	Definition
m__LONG__Index	StarPort device index
m__LONG__TargetId	StarPort device TargetId

m__CHAR__Name	StarPort device name

2.2.105 PPSTARPORT_DEVICE_TYPE Enumeration

C++

```
typedef enum _STARPORT_DEVICE_TYPE {
  STARPORT_DEVICE_TYPE_UNKNOWN = 0,
  STARPORT_DEVICE_TYPE_RAM,
  STARPORT_DEVICE_TYPE_HDD,
  STARPORT_DEVICE_TYPE_DVD,
  STARPORT_DEVICE_TYPE_AOE,
  STARPORT_DEVICE_TYPE_ISCSI
} ** PPSTARPORT_DEVICE_TYPE, * PSTARPORT_DEVICE_TYPE, STARPORT_DEVICE_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARPORT_DEVICE_TYPE_UNKNOWN = 0	Unknown device type
STARPORT_DEVICE_TYPE_RAM	RAM disk
STARPORT_DEVICE_TYPE_HDD	Virtual hard disk
STARPORT_DEVICE_TYPE_DVD	Virtual DVD
STARPORT_DEVICE_TYPE_AOE	AoE (ATA-over-Ethernet)
STARPORT_DEVICE_TYPE_ISCSI	iSCSI (SCSI-over-IP)

Description

Enum that represents StarPort device type

Member	Definition
STARPORT_DEVICE_TYPE_UNKNOWN	Unknown device type
STARPORT_DEVICE_TYPE_RAM	RAM disk
STARPORT_DEVICE_TYPE_HDD	Virtual hard disk
STARPORT_DEVICE_TYPE_DVD	Virtual DVD
STARPORT_DEVICE_TYPE_AOE	AoE (ATA-over-Ethernet)
STARPORT_DEVICE_TYPE_ISCSI	iSCSI (SCSI-over-IP)

2.2.106 PPSTARWAVE2_COMPRESSION Enumeration

C++

```
typedef enum _STARWAVE2_COMPRESSION {
  STARWAVE2_COMPRESSION_NONE = 0,
  STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10,
  STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25,
  STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50,
  STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75,
  STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90,
  STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98,
  STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000,
```

```
  STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000,
  STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000,
  STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000,
  STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000,
  STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000,
  STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000,
  STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000,
  STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000,
  STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000
} STARWAVE2_COMPRESSION, * PSTARWAVE2_COMPRESSION, ** PPSTARWAVE2_COMPRESSION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARWAVE2_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98
STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates

All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

2.2.107 PPSTARWAVE2_COMPRESSION_PROFILE Structure

C++

```
typedef struct _STARWAVE2_COMPRESSION_PROFILE {
  UINT m__UINT__CompressionType;
  PCHAR m__PCHAR__CustomFactoryID;
  union {
    int m__int__CBRBitRate;
    int m__int__VBRQuality;
  }
  int m__int__ABRBitRate;
  int m__int__MaxBitRate;
  union {
    int m__int__MinBitRate;
    int m__int__MaxFrameWindow;
  }
  LPVOID m__LPVOID__CustomData;
} STARWAVE2_COMPRESSION_PROFILE, * PSTARWAVE2_COMPRESSION_PROFILE, **
PPSTARWAVE2_COMPRESSION_PROFILE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
UINT m__UINT__CompressionType;	(STARBURN_STARWAVE2_COMPRESS_TYPE (see page 560))
int m__int__CBRBitRate;	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320
int m__int__VBRQuality;	(MP3 compression 0..9) (WMA compression 1..100)
int m__int__ABRBitRate;	(MP3 && OGG compression)
int m__int__MaxBitRate;	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320, for CBR mode this setting is ignored set this field to 0 if you don't want to use it
int m__int__MinBitRate;	(MP3 && OGG) 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320 set this field to 0 if you don't want to use it
int m__int__MaxFrameWindow;	WMA, Frame size in milliseconds

2.2.108 PQ_SUBCHANNEL Structure

C++

```
typedef struct _PQ_SUBCHANNEL {
  UCHAR m__UCHAR__ADR : 4;
  UCHAR m__UCHAR__CONTROL : 4;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__IndexNumber;
  UCHAR m__UCHAR__Min;
  UCHAR m__UCHAR__Sec;
  UCHAR m__UCHAR__Frame;
  UCHAR m__UCHAR__ZERO;
  UCHAR m__UCHAR__AMIN;
  UCHAR m__UCHAR__ASEC;
  UCHAR m__UCHAR__AFRAME;
  UCHAR m__UCHAR__CRC1_Reserved1;
  UCHAR m__UCHAR__CRC2_Reserved2;
  UCHAR m__UCHAR__Reserved3[ 3 ];
  UCHAR m__UCHAR__Reserved4 : 7;
  UCHAR m__UCHAR__PSubChannel : 1;
} * PPQ_SUBCHANNEL, PQ_SUBCHANNEL;
```

File

StarBurn.h (see page 662)

Members

Members	Description
UCHAR m__UCHAR__ADR : 4;	Track ADR
UCHAR m__UCHAR__CONTROL : 4;	Track CONTROL
UCHAR m__UCHAR__TrackNumber;	Track Number
UCHAR m__UCHAR__IndexNumber;	Index Number
UCHAR m__UCHAR__Min;	Min (from the beginning of the track)
UCHAR m__UCHAR__Sec;	Sec (from the beginning of the track)
UCHAR m__UCHAR__Frame;	Frame (from the beginning of the track)
UCHAR m__UCHAR__ZERO;	MBZ
UCHAR m__UCHAR__AMIN;	Min (from the beginning of the disc)
UCHAR m__UCHAR__ASEC;	Sec (from the beginning of the disc)
UCHAR m__UCHAR__AFRAME;	Frame (from the beginning of the disc)
UCHAR m__UCHAR__CRC1_Reserved1;	CRC1 or MBZ
UCHAR m__UCHAR__CRC2_Reserved2;	CRC2 or MBZ
UCHAR m__UCHAR__Reserved3[3];	MBZ
UCHAR m__UCHAR__Reserved4 : 7;	MBZ
UCHAR m__UCHAR__PSubChannel : 1;	P sub-channel bit

Description

Structure that represents formatted PQ sub-channel

Member	Definition
m__UCHAR__ADR	Track ADR
m__UCHAR__CONTROL	Track CONTROL
m__UCHAR__TrackNumber	Track Number
m__UCHAR__IndexNumber	Index Number
m__UCHAR__Min	Min (from the beginning of the track)
m__UCHAR__Sec	Sec (from the beginning of the track)
m__UCHAR__Frame	Frame (from the beginning of the track)
m__UCHAR__ZERO	MBZ
m__UCHAR__AMIN	Min (from the beginning of the disc)
m__UCHAR__ASEC	Sec (from the beginning of the disc)
m__UCHAR__AFRAME	Frame (from the beginning of the disc)
m__UCHAR__CRC1_Reserved1	CRC1 or MBZ
m__UCHAR__CRC2_Reserved2	CRC2 or MBZ
m__UCHAR__Reserved3[3]	MBZ
m__UCHAR__Reserved4	MBZ
m__UCHAR__PSubChannel	P sub-channel bit

2.2.109 PREAD_MODE Enumeration

C++

```
typedef enum _READ_MODE {
  READ_MODE_COOKED = 0,
  READ_MODE_RAW,
  READ_MODE_RAW_PQ,
  READ_MODE_RAW_RAW_PW,
  READ_MODE_PQ,
  READ_MODE_RAW_PW
} * PREAD_MODE, READ_MODE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
READ_MODE_COOKED = 0	Cooked data
READ_MODE_RAW	Raw data
READ_MODE_RAW_PQ	Raw data + PQ sub-channel
READ_MODE_RAW_RAW_PW	Raw data + raw P-W sub-channel
READ_MODE_PQ	PQ sub-channel only (no main channel data)
READ_MODE_RAW_PW	Raw P-W sub-channel only (no main channel data)

Description

Enum that represents raw read modes

Member	Definition
READ_MODE_COOKED	Cooked data
READ_MODE_RAW	Raw data
READ_MODE_RAW_PQ	Raw data + PQ sub-channel
READ_MODE_RAW_RAW_PW	Raw data + raw P-W sub-channel
READ_MODE_PQ	PQ sub-channel only (no main channel data)
READ_MODE_RAW_PW	Raw P-W sub-channel only (no main channel data)

2.2.110 PSCSI_DEVICE_ADDRESS Structure

C++

```
typedef struct _SCSI_DEVICE_ADDRESS {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__PortID;
  UCHAR m__UCHAR__BusID;
  UCHAR m__UCHAR__TargetID;
  UCHAR m__UCHAR__LUN;
} * PSCSI_DEVICE_ADDRESS, SCSI_DEVICE_ADDRESS;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__PortID;`	Port Id - Logical SCSI adapter ID if ASPI is used
`UCHAR m__UCHAR__BusID;`	Bus ID - 0 if ASPI is used
`UCHAR m__UCHAR__TargetID;`	Target Id
`UCHAR m__UCHAR__LUN;`	LUN (Logical Unit Number)

Description

Structure that represents SCSI device address

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__PortID	Port ID - Logical SCSI adapter ID if ASPI is used
m__UCHAR__BusID	Bus ID - 0 if ASPI is used
m__UCHAR__TargetID	Target ID
m__UCHAR__LUN	LUN (Logical Unit Number)

2.2.111 PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS Structure

C++

```
typedef struct _STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS {
  ULONG m__ULONG__SizeInUCHARs;
  BOOLEAN m__BOOLEAN__IsCDROMRead;
  BOOLEAN m__BOOLEAN__IsCDRRead;
  BOOLEAN m__BOOLEAN__IsCDRWrite;
  BOOLEAN m__BOOLEAN__IsCDRWRead;
  BOOLEAN m__BOOLEAN__IsCDRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDROMRead;
  BOOLEAN m__BOOLEAN__IsDVDRRead;
  BOOLEAN m__BOOLEAN__IsDVDRWrite;
  BOOLEAN m__BOOLEAN__IsDVDRDLRead;
  BOOLEAN m__BOOLEAN__IsDVDRDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDRWRead;
  BOOLEAN m__BOOLEAN__IsDVDRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDRWDLRead;
  BOOLEAN m__BOOLEAN__IsDVDRWDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDRAMRead;
  BOOLEAN m__BOOLEAN__IsDVDRAMWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite;
  BOOLEAN m__BOOLEAN__IsBDROMRead;
  BOOLEAN m__BOOLEAN__IsBDRRead;
  BOOLEAN m__BOOLEAN__IsBDRWrite;
  BOOLEAN m__BOOLEAN__IsBDRDLRead;
  BOOLEAN m__BOOLEAN__IsBDRDLWrite;
  BOOLEAN m__BOOLEAN__IsBDRERead;
  BOOLEAN m__BOOLEAN__IsBDREWrite;
  BOOLEAN m__BOOLEAN__IsBDREDLRead;
  BOOLEAN m__BOOLEAN__IsBDREDLWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDROMRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRDLRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRWRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite;
} * PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS, STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS;
```

File

StarBurn.h (see page 662)

Members

Members	Description
ULONG m__ULONG__SizeInUCHARs;	This structure size in UCHARs
BOOLEAN m__BOOLEAN__IsCDROMRead;	Is CD-ROM read capable
BOOLEAN m__BOOLEAN__IsCDRRead;	Is CD-R read capable
BOOLEAN m__BOOLEAN__IsCDRWrite;	Is CD-R write capable

BOOLEAN m__BOOLEAN__IsCDRWRead;	Is CD-RW read capable
BOOLEAN m__BOOLEAN__IsCDRWWrite;	Is CD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDROMRead;	Is DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsDVDRRead;	Is DVD-R read capable
BOOLEAN m__BOOLEAN__IsDVDRWrite;	Is DVD-R write capable
BOOLEAN m__BOOLEAN__IsDVDRDLRead;	Is DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsDVDRDLWrite;	Is DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsDVDRWRead;	Is DVD-RW read capable
BOOLEAN m__BOOLEAN__IsDVDRWWrite;	Is DVD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDRWDLRead;	Is DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDRWDLWrite;	Is DVD-RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDRAMRead;	Is DVD-RAM read capable
BOOLEAN m__BOOLEAN__IsDVDRAMWrite;	Is DVD-RAM write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead;	Is DVD+RW read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite;	Is DVD+RW write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead;	Is DVD+RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite;	Is DVD+RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRRead;	Is DVD+R read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite;	Is DVD+R write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead;	Is DVD+R DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite;	Is DVD+R DL write capable
BOOLEAN m__BOOLEAN__IsBDROMRead;	Is BD-ROM read capable
BOOLEAN m__BOOLEAN__IsBDRRead;	Is BD-R read capable
BOOLEAN m__BOOLEAN__IsBDRWrite;	Is BD-R write capable
BOOLEAN m__BOOLEAN__IsBDRDLRead;	Is BD-R DL read capable
BOOLEAN m__BOOLEAN__IsBDRDLWrite;	Is BD-R DL write capable
BOOLEAN m__BOOLEAN__IsBDRERead;	Is BD-RE read capable
BOOLEAN m__BOOLEAN__IsBDREWrite;	Is BD-RE write capable
BOOLEAN m__BOOLEAN__IsBDREDLRead;	Is BD-RE DL read capable
BOOLEAN m__BOOLEAN__IsBDREDLWrite;	Is BD-RE DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDROMRead;	Is HD-DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsHDDVDRRead;	Is HD-DVD-R read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWrite;	Is HD-DVD-R write capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLRead;	Is HD-DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite;	Is HD-DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWRead;	Is HD-DVD-RW read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWWrite;	Is HD-DVD-RW write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead;	Is HD-DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite;	Is HD-DVD-RW DL write capable

Description

Structure that represents advanced supported media formats

Member Definition
--------- ----------
ULONG m__ULONG__SizeInUCHARs This structure size in UCHARs
BOOLEAN m__BOOLEAN__IsCDROMRead Is CD-ROM read capable
BOOLEAN m__BOOLEAN__IsCDRRead Is CD-R read capable
BOOLEAN m__BOOLEAN__IsCDRWrite Is CD-R write capable
BOOLEAN m__BOOLEAN__IsCDRWRead Is CD-RW read capable
BOOLEAN m__BOOLEAN__IsCDRWWrite Is CD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDROMRead Is DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsDVDRRead Is DVD-R read capable

BOOLEAN m__BOOLEAN__IsDVDRWrite Is DVD-R write capable
BOOLEAN m__BOOLEAN__IsDVDRDLRead Is DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsDVDRDLWrite Is DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsDVDRWRead Is DVD-RW read capable
BOOLEAN m__BOOLEAN__IsDVDRWWrite Is DVD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDRWDLRead Is DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDRWDLWrite Is DVD-RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDRAMRead Is DVD-RAM read capable
BOOLEAN m__BOOLEAN__IsDVDRAMWrite Is DVD-RAM write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead Is DVD+RW read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite Is DVD+RW write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead Is DVD+RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite Is DVD+RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRRead Is DVD+R read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite Is DVD+R write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead Is DVD+R DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite Is DVD+R DL write capable
BOOLEAN m__BOOLEAN__IsBDROMRead Is BD-ROM read capable
BOOLEAN m__BOOLEAN__IsBDRRead Is BD-R read capable
BOOLEAN m__BOOLEAN__IsBDRWrite Is BD-R write capable
BOOLEAN m__BOOLEAN__IsBDRDLRead Is BD-R DL read capable
BOOLEAN m__BOOLEAN__IsBDRDLWrite Is BD-R DL write capable
BOOLEAN m__BOOLEAN__IsBDRERead Is BD-RE read capable
BOOLEAN m__BOOLEAN__IsBDREWrite Is BD-RE write capable
BOOLEAN m__BOOLEAN__IsBDREDLRead Is BD-RE DL read capable
BOOLEAN m__BOOLEAN__IsBDREDLWrite Is BD-RE DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDROMMRead Is HD-DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsHDDVDRRead Is HD-DVD-R read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWrite Is HD-DVD-R write capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLRead Is HD-DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite Is HD-DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWRead Is HD-DVD-RW read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWWrite Is HD-DVD-RW write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead Is HD-DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite Is HD-DVD-RW DL write capable

2.2.112 PSTARBURN_BDRE_FORMAT_PROFILE Structure

C++

```
typedef struct _STARBURN_BDRE_FORMAT_PROFILE {
  LONG m__LONG__FormatProfileNumber;
```

```
  ULONG m__ULONG__NumberOfBlocksOrUserAreaDataSize;
  UCHAR m__UCHAR__CertificationType : 2;
  UCHAR m__UCHAR__FormatType : 6;
  ULONG m__ULONG__SpareAreaSizeInClustersOrBlockLength;
} ** PPSTARBURN_BDRE_FORMAT_PROFILE, * PSTARBURN_BDRE_FORMAT_PROFILE,
STARBURN_BDRE_FORMAT_PROFILE;
```

File

StarBurn.h (see page 662)

Description

Structure that represents format profile for BD-RE media

Member Definition
--------- ----------
LONG m__LONG__FormatProfileNumber Format profile number
UCHAR m__ULONG__NumberOfBlocksOrUserAreaDataSize Number of logical block(s) or user area data size
UCHAR m__UCHAR__CertificationType Certification type
UCHAR m__UCHAR__FormatType Format type
UCHAR m__ULONG__SpareAreaSizeInClustersOrBlockLength Spare area size in cluster(s) or block length

2.2.113 PSTARBURN_CD_MODE Enumeration

C++

```
typedef enum _STARBURN_CD_MODE {
  CD_MODE_UNKNOWN = 0,
  CD_MODE_AUDIO,
  CD_MODE_VIDEO,
  CD_MODE_DATA,
  CD_MODE_RESERVED
} * PSTARBURN_CD_MODE, STARBURN_CD_MODE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
CD_MODE_UNKNOWN = 0	Unknown CD mode
CD_MODE_AUDIO	Audio CD mode
CD_MODE_VIDEO	Video CD mode
CD_MODE_DATA	Data CD mode
CD_MODE_RESERVED	Reserved

Description

Enum that represents CD modes type

Member	Definition
CD_MODE_UNKNOWN	Unknown CD mode
CD_MODE_AUDIO	Audio CD mode
CD_MODE_VIDEO	Video CD mode

CD_MODE_DATA	Data CD mode
CD_MODE_RESERVED	Reserved

2.2.114 PSTARBURN_DISC_ATIP_INFORMATION Structure

C++

```
typedef struct _STARBURN_DISC_ATIP_INFORMATION {
  USHORT m__USHORT__DataLength;
  UCHAR m__UCHAR__Reserved1[2];
  UCHAR m__UCHAR__RefrenceSpeed : 3;
  UCHAR m__UCHAR__Reserved2 : 1;
  UCHAR m__UCHAR__IndicativeTargetWritingPower : 4;
  UCHAR m__UCHAR__Reserved3 : 6;
  UCHAR m__UCHAR__URU : 1;
  UCHAR m__UCHAR__Zero : 1;
  UCHAR m__UCHAR__IsA3Valid : 1;
  UCHAR m__UCHAR__IsA2Valid : 1;
  UCHAR m__UCHAR__IsA1Valid : 1;
  UCHAR m__UCHAR__DiscSubType : 3;
  UCHAR m__UCHAR__DiscType : 1;
  UCHAR m__UCHAR__One : 1;
  UCHAR m__UCHAR__Reserved4;
  UCHAR m__UCHAR__LeadInStartTime[3];
  UCHAR m__UCHAR__Reserved5;
  UCHAR m__UCHAR__LastPossibleLeadOutStartTime[3];
  UCHAR m__UCHAR__Reserved6;
  UCHAR m__UCHAR__A1Values[3];
  UCHAR m__UCHAR__Reserved7;
  UCHAR m__UCHAR__A2Values[3];
  UCHAR m__UCHAR__Reserved8;
  UCHAR m__UCHAR__A3Values[3];
  UCHAR m__UCHAR__Reserved9;
  UCHAR m__UCHAR__S4Values[3];
  UCHAR m__UCHAR__Reserved10;
} * PSTARBURN_DISC_ATIP_INFORMATION, STARBURN_DISC_ATIP_INFORMATION;
```

File

StarBurn.h (see page 662)

Description

Structure that represents Disc ATIP information

Member Definition
--------- ----------
m__USHORT__DataLength
m__UCHAR__Reserved1
m__UCHAR__RefrenceSpeed
m__UCHAR__Reserved2
m__UCHAR__IndicativeTargetWritingPower
m__UCHAR__Reserved3
m__UCHAR__URU
m__UCHAR__Zero
m__UCHAR__IsA3Valid

m__UCHAR__IsA2Valid
m__UCHAR__IsA1Valid
m__UCHAR__DiscSubType
m__UCHAR__DiscType
m__UCHAR__One
m__UCHAR__Reserved4
m__UCHAR__LeadInStartTime
m__UCHAR__Reserved5
m__UCHAR__LastPossibleLeadOutStartTime
m__UCHAR__Reserved6
m__UCHAR__A1Values
m__UCHAR__Reserved7
m__UCHAR__A2Values
m__UCHAR__Reserved8
m__UCHAR__A3Values
m__UCHAR__Reserved9
m__UCHAR__S4Values
m__UCHAR__Reserved10

2.2.115 PSTARBURN_DISC_INFORMATION Structure

C++

```
typedef struct _STARBURN_DISC_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__DiscStatus;
  UCHAR m__UCHAR__LastSessionStatus;
  BOOLEAN m__BOOLEAN__IsErasable;
  UCHAR m__UCHAR__FirstTrackNumber;
  UCHAR m__UCHAR__NumberOfSessions;
  UCHAR m__UCHAR__LastSessionFirstTrack;
  UCHAR m__UCHAR__LastSessionLastTrack;
  BOOLEAN m__BOOLEAN__IsGEN;
  BOOLEAN m__BOOLEAN__IsDBCValid;
  BOOLEAN m__BOOLEAN__IsDIDValid;
  UCHAR m__UCHAR__DiscType;
  UCHAR m__UCHAR__DiscIdentification[ 4 ];
  UCHAR m__UCHAR__LastSessionLeadIn[ 4 ];
  UCHAR m__UCHAR__LastPossibleStartTime[ 4 ];
  UCHAR m__UCHAR__DiscBarCode[ 8 ];
  UCHAR m__UCHAR__NumberOfOPCEntries;
} * PSTARBURN_DISC_INFORMATION, STARBURN_DISC_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__DiscStatus;`	Disc status (See DISC_STATUS_XXX constants)
`UCHAR m__UCHAR__LastSessionStatus;`	Last session status (See LAST_SESSION_XXX constants)
`BOOLEAN m__BOOLEAN__IsErasable;`	Is disc erasable (CD-RW or DVD-RW or DVD+RW or DVD-RAM)

UCHAR m__UCHAR__FirstTrackNumber;	First track number on the disc
UCHAR m__UCHAR__NumberOfSessions;	Number of sessions on the disc
UCHAR m__UCHAR__LastSessionFirstTrack;	First track number of last complete session on the disc
UCHAR m__UCHAR__LastSessionLastTrack;	Last track number of last complete session on the disc
BOOLEAN m__BOOLEAN__IsGEN;	Unrestricted use, when set to one - every application can write to the disc
BOOLEAN m__BOOLEAN__IsDBCValid;	Is Disc Bar Code field valid
BOOLEAN m__BOOLEAN__IsDIDValid;	Is Disc Identification field Valid
UCHAR m__UCHAR__DiscType;	Disc type (See DISC_TYPE_XXX constants)
UCHAR m__UCHAR__DiscIdentification[4];	Disc identification number recorded in the PMA, see m__UCHAR__IsDIDValid
UCHAR m__UCHAR__LastSessionLeadIn[4];	MSF of last session lead in start time
UCHAR m__UCHAR__LastPossibleStartTime[4];	Last possible start time for lead out on this disc
UCHAR m__UCHAR__DiscBarCode[8];	Disc bar code stored here if the device is capable of reading disc bar codes
UCHAR m__UCHAR__NumberOfOPCEntries;	Number of optimum power calibration entries at the end of this structure

Description

Structure that represents Disc information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__DiscStatus	Disc status (See DISC_STATUS_XXX constants)
m__UCHAR__LastSessionStatus	Last session status (See LAST_SESSION_XXX constants)
m__BOOLEAN__IsErasable	Is disc erasable (CD-RW or DVD-RW or DVD+RW or DVD-RAM)
m__UCHAR__FirstTrackNumber	First track number on the disc
m__UCHAR__NumberOfSessions	Number of sessions on the disc
m__UCHAR__LastSessionFirstTrack	First track number of last complete session on the disc
m__UCHAR__LastSessionLastTrack	Last track number of last complete session on the disc
m__BOOLEAN__IsGEN	Unrestricted use, when set to one - every application can write to the disc
m__BOOLEAN__IsDBCValid	Is Disc Bar Code field valid
m__BOOLEAN__IsDIDValid	Is Disc Identification field Valid
m__UCHAR__DiscType	Disc type (See DISC_TYPE_XXX constants)
m__UCHAR__DiscIdentification	Disc identification number recorded in the PMA, see m__UCHAR__IsDIDValid
m__UCHAR__LastSessionLeadId	MSF of last session lead in start time
m__UCHAR__LastPossibleStartTime	Last possible start time for lead out on this disc
m__UCHAR__DiscBarCode	Disc bar code stored here if the device is capable of reading disc bar codes
m__UCHAR__NumberOfOPCEntries	Number of optimum power calibration entries at the end of this structure

2.2.116 PSTARBURN_STARWAVE_CALLBACK_REASON Enumeration

C++

```
typedef enum _STARBURN_STARWAVE_CALLBACK_REASON {
  STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0,
  STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS
} STARBURN_STARWAVE_CALLBACK_REASON, * PSTARBURN_STARWAVE_CALLBACK_REASON, **
PPSTARBURN_STARWAVE_CALLBACK_REASON;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

Description

StarWave callback reasons we'll be using

Member	Definition
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

2.2.117 PSTARBURN_STARWAVE_COMPRESSION Enumeration

C++

```
typedef enum _STARBURN_STARWAVE_COMPRESSION {
  STARBURN_STARWAVE_COMPRESSION_NONE = 0,
  STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000
} STARBURN_STARWAVE_COMPRESSION, * PSTARBURN_STARWAVE_COMPRESSION, **
PPSTARBURN_STARWAVE_COMPRESSION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps

STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

Member	Definition
STARBURN_STARWAVE_COMPRESSION_NONE	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSS...	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98	WMA compressed stream, VBR at Quality 98
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K	WMA compressed stream, CBR at 32 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K	WMA compressed stream, CBR at 320 Kbps

2.2.118 PSTARBURN_STARWAVE2_CONVERSION_MODE Enumeration

C++

```
typedef enum _STARBURN_STARWAVE2_CONVERSION_MODE {
  STARBURN_STARWAVE2_OGG_MODE0,
  STARBURN_STARWAVE2_OGG_MODE1,
  STARBURN_STARWAVE2_OGG_MODE2,
  STARBURN_STARWAVE2_OGG_MODE_MAX = STARBURN_STARWAVE2_OGG_MODE2,
  STARBURN_STARWAVE2_MP3_MODE0,
  STARBURN_STARWAVE2_MP3_MODE1,
  STARBURN_STARWAVE2_MP3_MODE2,
  STARBURN_STARWAVE2_MP3_MODE_MAX = STARBURN_STARWAVE2_MP3_MODE2
```

```
} STARBURN_STARWAVE2_CONVERSION_MODE, * PSTARBURN_STARWAVE2_CONVERSION_MODE;
```

File

StarBurn.h (see page 662)

Description

Structure that represents conversion mode

Member	Definition
OGG_MODE0	VBR, variable bit rate
OGG_MODE1	ABR, average bit rate
OGG_MODE2	CBR, constant bit rate
OGG_MODE_MAX	Now equal to OGG_MODE2
MP3_MODE0	CBR, constant bit rate
MP3_MODE1	ABR, average bit rate
MP3_MODE2	VBR, variable bit rate
MP3_MODE_MAX	Now equal to MP3_MODE2

2.2.119 PSTARBURN_STARWAVE2_INIT_PARAMS Structure

C++

```
typedef struct _STARBURN_STARWAVE2_INIT_PARAMS {
  STARBURN_STARWAVE2_CONVERSION_MODE m__STARBURN_STARWAVE2_CONVERSION_MODE;
  FLOAT m__FLOAT__flQuality;
  LONG m__LONG__nQuality;
  LONG m__LONG__maxBitrate;
  LONG m__LONG__nominalBitrate;
  LONG m_LONG__minBitrate;
  STARBURN_STARWAVE2_QUALITY_MODE m__STARBURN_STARWAVE2_QUALITY_MODE;
} STARBURN_STARWAVE2_INIT_PARAMS, * PSTARBURN_STARWAVE2_INIT_PARAMS;
```

File

StarBurn.h (see page 662)

Description

Structure that represents initialization parameters for audio conversion

Member	Definition
m__STARBURN_STARWAVE2_CONVERSION_MODE	Conversion mode
m__FLOAT__flQuality	Quality level from 0. (lo) to 1. (hi) (use for OGG_MODE0)
m__LONG__nQuality	Quality level from 1 to 9 (use for MP3_MODE2)
m__LONG__maxBitrate	Maximum bit rate (use for OGG_MODE1/OGG_MODE2)
m__LONG__nominalBitrate	Nominal bit rate (use for OGG_MODE1/OGG_MODE2/MP3_MODE0/MP3_MODE1)
m__LONG__minBitrate	Minimum bit rate (use for OGG_MODE1/OGG_MODE2)

m__StarBurn_StarWave2_QUALITY_MODE	Quality mode (use for /MP3_MODE0/MP3_MODE1/MP3_MODE2)

2.2.120 PSTARBURN_STARWAVE2_QUALITY_MODE Enumeration

C++

```
typedef enum _STARBURN_STARWAVE2_QUALITY_MODE {
  STARBURN_STARWAVE2_QM_FAST = 0,
  STARBURN_STARWAVE2_QM_STANDARD,
  STARBURN_STARWAVE2_QM_HIGH
} STARBURN_STARWAVE2_QUALITY_MODE, * PSTARBURN_STARWAVE2_QUALITY_MODE;
```

File

StarBurn.h (see page 662)

Description

Enum that represents MP3 quality for conversion

Member	Definition
StarBurn_StarWave2_QM_FAST	Fast mode
StarBurn_StarWave2_QM_STANDARD	Standard mode
StarBurn_StarWave2_QM_HIGH	High quality mode

2.2.121 PSTARBURN_TRACK_INFORMATION Structure

C++

```
typedef struct _STARBURN_TRACK_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__TrackMode;
  BOOLEAN m__BOOLEAN__IsCopy;
  BOOLEAN m__BOOLEAN__IsDamage;
  UCHAR m__UCHAR__DataMode;
  BOOLEAN m__BOOLEAN__IsFixedPacket;
  BOOLEAN m__BOOLEAN__IsPacket;
  BOOLEAN m__BOOLEAN__IsBlank;
  BOOLEAN m__BOOLEAN__IsReserved;
  BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;
  LONG m__LONG__TrackStartAddress;
  LONG m__LONG__NextWritableAddress;
  LONG m__LONG__FreeLBs;
  LONG m__LONG__FixedPacketSizeInLBs;
} * PSTARBURN_TRACK_INFORMATION, STARBURN_TRACK_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__TrackNumber;`	Track number of this track
`UCHAR m__UCHAR__SessionNumber;`	Session number that contains this track
`UCHAR m__UCHAR__TrackMode;`	Track mode
`BOOLEAN m__BOOLEAN__IsCopy;`	Is this track second or higher generation copy
`BOOLEAN m__BOOLEAN__IsDamage;`	Is track damaged
`UCHAR m__UCHAR__DataMode;`	Data mode on the track
`BOOLEAN m__BOOLEAN__IsFixedPacket;`	Is fixed packet used on this track
`BOOLEAN m__BOOLEAN__IsPacket;`	Is packet recording used on the track
`BOOLEAN m__BOOLEAN__IsBlank;`	Is track blank
`BOOLEAN m__BOOLEAN__IsReserved;`	Is track reserved
`BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;`	Is NWA (Next Writable Address) valid
`LONG m__LONG__TrackStartAddress;`	Starting address (LBA or MSF) for this track
`LONG m__LONG__NextWritableAddress;`	NWA (Next Writable Address)
`LONG m__LONG__FreeLBs;`	Free logical blocks on this track
`LONG m__LONG__FixedPacketSizeInLBs;`	Fixed packet size in LBs

Description

Structure that represents Track information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number of this track
m__UCHAR__SessionNumber	Session number that contains this track
m__UCHAR__TrackMode	Track mode
m__BOOLEAN__IsCopy	Is this track second or higher generation copy
m__BOOLEAN__IsDamage	Is track damaged
m__UCHAR__DataMode	Data mode on the track
m__BOOLEAN__IsFixedPacket	Is fixed packet used on this track
m__BOOLEAN__IsPacket	Is packet recording used on the track
m__BOOLEAN__IsBlank	Is track blank
m__BOOLEAN__IsReserved	Is track reserved
m__BOOLEAN__IsNextWritable...	Is NWA (Next Writable Address) valid
m__LONG__TrackStartAddress	Starting address (LBA or MSF) for this track
m__LONG__NextWritableAddress	NWA (Next Writable Address)
m__LONG__FreeLBs	Free logical blocks on this track
m__LONG__FixedPacketSize...	Fixed packet size in UCHARs

2.2.122 PSTARBURN_TRACK_INFORMATION_EX Structure

C++

```
typedef struct _STARBURN_TRACK_INFORMATION_EX {
  BOOLEAN m__BOOLEAN__IsValid;
```

```
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__TrackMode;
  BOOLEAN m__BOOLEAN__IsCopy;
  BOOLEAN m__BOOLEAN__IsDamage;
  UCHAR m__UCHAR__DataMode;
  BOOLEAN m__BOOLEAN__IsFixedPacket;
  BOOLEAN m__BOOLEAN__IsPacket;
  BOOLEAN m__BOOLEAN__IsBlank;
  BOOLEAN m__BOOLEAN__IsReserved;
  BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;
  LONG m__LONG__TrackStartAddress;
  LONG m__LONG__LastRecordedAddress;
  LONG m__LONG__NextWritableAddress;
  LONG m__LONG__FreeLBs;
  LONG m__LONG__FixedPacketSizeInLBs;
} * PSTARBURN_TRACK_INFORMATION_EX, STARBURN_TRACK_INFORMATION_EX;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__TrackNumber;`	Track number of this track
`UCHAR m__UCHAR__SessionNumber;`	Session number that contains this track
`UCHAR m__UCHAR__TrackMode;`	Track mode
`BOOLEAN m__BOOLEAN__IsCopy;`	Is this track second or higher generation copy
`BOOLEAN m__BOOLEAN__IsDamage;`	Is track damaged
`UCHAR m__UCHAR__DataMode;`	Data mode on the track
`BOOLEAN m__BOOLEAN__IsFixedPacket;`	Is fixed packet used on this track
`BOOLEAN m__BOOLEAN__IsPacket;`	Is packet recording used on the track
`BOOLEAN m__BOOLEAN__IsBlank;`	Is track blank
`BOOLEAN m__BOOLEAN__IsReserved;`	Is track reserved
`BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;`	Is NWA (Next Writable Address) valid
`LONG m__LONG__TrackStartAddress;`	Starting address (LBA or MSF) for this track
`LONG m__LONG__LastRecordedAddress;`	Last Recorded Address (LBA or MSF) for this track
`LONG m__LONG__NextWritableAddress;`	NWA (Next Writable Address)
`LONG m__LONG__FreeLBs;`	Free logical blocks on this track
`LONG m__LONG__FixedPacketSizeInLBs;`	Fixed packet size in LBs

Description

Structure that represents Track information extended

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number of this track
m__UCHAR__SessionNumber	Session number that contains this track
m__UCHAR__TrackMode	Track mode
m__BOOLEAN__IsCopy	Is this track second or higher generation copy
m__BOOLEAN__IsDamage	Is track damaged
m__UCHAR__DataMode	Data mode on the track
m__BOOLEAN__IsFixedPacket	Is fixed packet used on this track
m__BOOLEAN__IsPacket	Is packet recording used on the track
m__BOOLEAN__IsBlank	Is track blank
m__BOOLEAN__IsReserved	Is track reserved

m__BOOLEAN__IsNextWritable...	Is NWA (Next Writable Address) valid
m__LONG__TrackStartAddress	Starting address (LBA or MSF) for this track
m__LONG__LastRecordedAddress	Last Recorded Address (LBA or MSF) for this track
m__LONG__NextWritableAddress	NWA (Next Writable Address)
m__LONG__FreeLBs	Free logical blocks on this track
m__LONG__FixedPacketSize...	Fixed packet size in UCHARs

2.2.123 PSTARPORT_DEVICE_LIST Structure

C++

```
typedef struct _STARPORT_DEVICE_LIST {
  LONG m__LONG__NumberOfEntries;
  STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];
} ** PPSTARPORT_DEVICE_LIST, * PSTARPORT_DEVICE_LIST, STARPORT_DEVICE_LIST;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of StarPort device list entries
`STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];`	StarPort device list entries

Description

Structure that represents StarPort device list

Member	Definition
m__LONG__NumberOfEntries	Number of StarPort device list entries
m__STARPORT_DEVICE_LIST_ENTRY	StarPort device list entries

2.2.124 PSTARPORT_DEVICE_LIST_ENTRY Structure

C++

```
typedef struct _STARPORT_DEVICE_LIST_ENTRY {
  LONG m__LONG__Index;
  LONG m__LONG__TargetId;
  CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];
} ** PPSTARPORT_DEVICE_LIST_ENTRY, * PSTARPORT_DEVICE_LIST_ENTRY,
STARPORT_DEVICE_LIST_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__Index;`	StarPort device index
`LONG m__LONG__TargetId;`	StarPort device TargetId
`CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];`	StarPort device name

Description

Structure that represents StarPort device list entry

Member	Definition
m__LONG__Index	StarPort device index
m__LONG__TargetId	StarPort device TargetId
m__CHAR__Name	StarPort device name

2.2.125 PSTARPORT_DEVICE_TYPE Enumeration

C++

```
typedef enum _STARPORT_DEVICE_TYPE {
  STARPORT_DEVICE_TYPE_UNKNOWN = 0,
  STARPORT_DEVICE_TYPE_RAM,
  STARPORT_DEVICE_TYPE_HDD,
  STARPORT_DEVICE_TYPE_DVD,
  STARPORT_DEVICE_TYPE_AOE,
  STARPORT_DEVICE_TYPE_ISCSI
} ** PPSTARPORT_DEVICE_TYPE, * PSTARPORT_DEVICE_TYPE, STARPORT_DEVICE_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARPORT_DEVICE_TYPE_UNKNOWN = 0	Unknown device type
STARPORT_DEVICE_TYPE_RAM	RAM disk
STARPORT_DEVICE_TYPE_HDD	Virtual hard disk
STARPORT_DEVICE_TYPE_DVD	Virtual DVD
STARPORT_DEVICE_TYPE_AOE	AoE (ATA-over-Ethernet)
STARPORT_DEVICE_TYPE_ISCSI	iSCSI (SCSI-over-IP)

Description

Enum that represents StarPort device type

Member	Definition
STARPORT_DEVICE_TYPE_UNKNOWN	Unknown device type
STARPORT_DEVICE_TYPE_RAM	RAM disk
STARPORT_DEVICE_TYPE_HDD	Virtual hard disk
STARPORT_DEVICE_TYPE_DVD	Virtual DVD
STARPORT_DEVICE_TYPE_AOE	AoE (ATA-over-Ethernet)
STARPORT_DEVICE_TYPE_ISCSI	iSCSI (SCSI-over-IP)

2.2.126 PSTARWAVE2_COMPRESSION Enumeration

C++

```
typedef enum _STARWAVE2_COMPRESSION {
  STARWAVE2_COMPRESSION_NONE = 0,
  STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10,
  STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25,
  STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50,
  STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75,
  STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90,
  STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98,
  STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000,
  STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000,
  STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000,
  STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000,
  STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000,
  STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000,
  STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000,
  STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000,
  STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000,
  STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000
} STARWAVE2_COMPRESSION, * PSTARWAVE2_COMPRESSION, ** PPSTARWAVE2_COMPRESSION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARWAVE2_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98
STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates

All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

2.2.127 PSTARWAVE2_COMPRESSION_PROFILE Structure

C++

```
typedef struct _STARWAVE2_COMPRESSION_PROFILE {
  UINT m__UINT__CompressionType;
  PCHAR m__PCHAR__CustomFactoryID;
  union {
    int m__int__CBRBitRate;
    int m__int__VBRQuality;
  }
  int m__int__ABRBitRate;
  int m__int__MaxBitRate;
  union {
    int m__int__MinBitRate;
    int m__int__MaxFrameWindow;
  }
  LPVOID m__LPVOID__CustomData;
} STARWAVE2_COMPRESSION_PROFILE, * PSTARWAVE2_COMPRESSION_PROFILE, **
PPSTARWAVE2_COMPRESSION_PROFILE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UINT m__UINT__CompressionType;`	(STARBURN_STARWAVE2_COMPRESS_TYPE (see page 560))
`int m__int__CBRBitRate;`	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320
`int m__int__VBRQuality;`	(MP3 compression 0..9) (WMA compression 1..100)
`int m__int__ABRBitRate;`	(MP3 && OGG compression)
`int m__int__MaxBitRate;`	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320, for CBR mode this setting is ignored set this field to 0 if you don't want to use it
`int m__int__MinBitRate;`	(MP3 && OGG) 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320 set this field to 0 if you don't want to use it
`int m__int__MaxFrameWindow;`	WMA, Frame size in milliseconds

2.2.128 PTOC_ENTRY Structure

C++

```
typedef struct _TOC_ENTRY {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  LONG m__LONG__StartingLBA;
  UCHAR m__UCHAR__StartingMSF[ 3 ];
  LONG m__LONG__EndingLBA;
  UCHAR m__UCHAR__EndingMSF[ 3 ];
  BOOLEAN m__BOOLEAN__IsMCNAvailable;
  BOOLEAN m__BOOLEAN__IsISRCAvailable;
  BOOLEAN m__BOOLEAN__IsFourChannelsAudio;
  BOOLEAN m__BOOLEAN__IsPreEmphasisAudio;
  BOOLEAN m__BOOLEAN__IsData;
  BOOLEAN m__BOOLEAN__IsAudio;
  BOOLEAN m__BOOLEAN__IsDigitalCopyProhibited;
  UCHAR m__UCHAR__TrackMode;
  UCHAR m__UCHAR__MODE2Form;
  ULONG m__ULONG__LBSizeInUCHARs;
  LONG m__LONG__Index00;
```

```
} * PTOC_ENTRY, TOC_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsValid;	Is this data valid (structure was really filled)
UCHAR m__UCHAR__TrackNumber;	Track number
UCHAR m__UCHAR__SessionNumber;	Session number that this track belongs to
LONG m__LONG__StartingLBA;	Starting LBA
UCHAR m__UCHAR__StartingMSF[3];	Starting MSF
LONG m__LONG__EndingLBA;	Ending LBA
UCHAR m__UCHAR__EndingMSF[3];	Ending MSF
BOOLEAN m__BOOLEAN__IsMCNAvailable;	Is Media Catalog Number available
BOOLEAN m__BOOLEAN__IsISRCAvailable;	Is International Standard Recording Code available
BOOLEAN m__BOOLEAN__IsFourChannelsAudio;	Is this four channels audio track
BOOLEAN m__BOOLEAN__IsPreEmphasisAudio;	Is this pre-emphasis audio track
BOOLEAN m__BOOLEAN__IsData;	Is this data track
BOOLEAN m__BOOLEAN__IsAudio;	Is this audio track
BOOLEAN m__BOOLEAN__IsDigitalCopyProhibited;	Is digital copy prohibited for this track
UCHAR m__UCHAR__TrackMode;	Track mode (0 for audio track, 1 for MODE1 and 2 for MODE2)
UCHAR m__UCHAR__MODE2Form;	MODE2 Form (0 for Formless, 1 for Form1 and 2 for Form2)
ULONG m__ULONG__LBSizeInUCHARs;	LB (logical block) size in UCHARs on this track
LONG m__LONG__Index00;	Index00 LBA (if present)

Description

Structure that represents TOC entry

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number
m__UCHAR__SessionNumber	Session number that this track belongs to
m__LONG__StartingLBA	Starting LBA
m__UCHAR__StartingMSF	Starting MSF
m__LONG__EndingLBA	Ending LBA
m__UCHAR__EndingMSF	Ending MSF
m__BOOLEAN__IsMCNAvailable	Is Media Catalog Number available
m__BOOLEAN__IsISRCAvailable	Is International Standard Recording Code available
m__BOOLEAN__IsFourChannelsAudio	Is this four channels audio track
m__BOOLEAN__IsPreEmphasisAudio	Is this pre-emphasis audio track
m__BOOLEAN__IsData	Is this data track
m__BOOLEAN__IsAudio	Is this audio track
m__BOOLEAN__IsDigitalCopy...	Is digital copy prohibited for this track
m__UCHAR__TrackMode	Track mode (0 for audio track, 1 for MODE1 and 2 for MODE2)
m__UCHAR__MODE2Form	MODE2 Form (0 for Formless, 1 for Form1 and 2 for Form2)
m__ULONG__LBSizeInUCHARs	LB (logical block) size in UCHARs on this track
m__LONG__Index00	Index00 LBA (if present)

2.2.129 PTOC_INFORMATION Structure

C++

```
typedef struct _TOC_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  BOOLEAN m__BOOLEAN__IsDVD;
  USHORT m__USHORT__ProtectedDVDRegions;
  UCHAR m__UCHAR__BusKeyForDiscKey[ 5 ];
  UCHAR m__UCHAR__NumberOfSessions;
  UCHAR m__UCHAR__NumberOfTracks;
  UCHAR m__UCHAR__NumberOfUnsortedEntries;
  TOC_ENTRY m__TOC_ENTRY[ NUMBER_OF_TRACKS ];
  FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW[ NUMBER_OF_TRACKS ];
  FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW__Unsorted[ NUMBER_OF_RAW_TRACKS ];
} * PTOC_INFORMATION, TOC_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`BOOLEAN m__BOOLEAN__IsDVD;`	Is this DVD media or not
`USHORT m__USHORT__ProtectedDVDRegions;`	Number of protected DVD regions
`UCHAR m__UCHAR__BusKeyForDiscKey[ 5 ];`	Bus key for disc key (CSS)
`UCHAR m__UCHAR__NumberOfSessions;`	Number of sessions
`UCHAR m__UCHAR__NumberOfTracks;`	Number of tracks
`UCHAR m__UCHAR__NumberOfUnsortedEntries;`	Number of entries in unsorted raw TOC
`TOC_ENTRY m__TOC_ENTRY[ NUMBER_OF_TRACKS ];`	TOC entries (decoded and extended)
`FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW[ NUMBER_OF_TRACKS ];`	Full TOC raw entries sorted, each entry corresponds to m__TOC_ENTRY with the same TNO
`FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW__Unsorted[ NUMBER_OF_RAW_TRACKS ];`	Full TOC raw entries unsorted

Description

Structure that represents TOC information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__BOOLEAN__IsDVD	Is this DVD media or not
m__USHORT__ProtectedDVDRegions	Number of protected DVD regions
m__UCHAR__BusKeyForDiscKey	Bus key for disc key (CSS)
m__UCHAR__NumberOfSessions	Number of sessions
m__UCHAR__NumberOfTracks	Number of tracks
m__UCHAR__NumberOfUnsortedEntries	Number of entries in unsorted raw TOC
m__TOC_ENTRY	TOC entries (decoded and extended)
m__FULL_TOC_ENTRY_RAW	Full TOC raw entries sorted, each entry corresponds to m__TOC_ENTRY with the same TNO
m__FULL_TOC_ENTRY_RAW__Unsorted	Full TOC raw entries unsorted

2.2.130 PUDF_CONTROL_BLOCK Structure

C++

```
typedef struct _UDF_CONTROL_BLOCK {
  void * m__PVOID__Head;
  void * m__PVOID__SystemStructures;
  void * m__PVOID__Tail;
  void * m__PVOID__Body;
} * PUDF_CONTROL_BLOCK, UDF_CONTROL_BLOCK;
```

File

StarBurn.h (see page 662)

Members

Members	Description
void * m__PVOID__Head;	Pointer to the head of the linked list
void * m__PVOID__SystemStructures;	Pointer to the allocated and formatted UDF system structures
void * m__PVOID__Tail;	Pointer to the tail of the linked list
void * m__PVOID__Body;	Pointer to block body itself

Description

Structure that represents UDF control block

Member	Definition
m__PVOID__Head	Pointer to the head of the linked list
m__PVOID__SystemStructures	Pointer to the allocated and formatted UDF system structures
m__PVOID__Tail	Pointer to the tail of the linked list
m__PVOID__Body	Pointer to block body itself

2.2.131 PUDF_FILE_EXTENT Structure

C++

```
typedef struct _UDF_FILE_EXTENT {
  ULONG m__ULONG__BeginLBA;
  ULONG m__ULONG__EndLBA;
} UDF_FILE_EXTENT, * PUDF_FILE_EXTENT;
```

File

StarBurn.h (see page 662)

Description

Structure that describes a single file extent

Member	Definition
m__ULONG__BeginLBA	First LBA of the extent
m__ULONG__EndLBA	Last LBA of the extent

2.2.132 PUDF_FILE_HANDLE Structure

C++

```
typedef struct _UDF_FILE_HANDLE {
  HANDLE m__HANDLE;
} * PUDF_FILE_HANDLE, UDF_FILE_HANDLE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
HANDLE m__HANDLE;	Win32 file handle

Description

Structure that represents UDF file handle

Member	Definition
m__HANDLE	Win32 file handle

2.2.133 PUDF_FILE_LOOKUP_ENTRY Structure

C++

```
typedef struct _UDF_FILE_LOOKUP_ENTRY {
  ULONG m__ULONG__Size;
  PWCHAR m__PCHAR__FileName;
  UINT m__UINT__ExtentCount;
  PUDF_FILE_EXTENT m__Extents;
} UDF_FILE_LOOKUP_ENTRY, * PUDF_FILE_LOOKUP_ENTRY;
```

File

StarBurn.h (see page 662)

Description

Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function

Member	Definition
m__ULONG__Size	Entry size
m__PWCHAR__FileName	File name
m__UINT__ExtentCount	Number of file extents
m__Extents	Array of file extents

2.2.134 PUDF_LOOKUP_DIR_ENTRY Structure

C++

```
typedef struct _UDF_LOOKUP_DIR_ENTRY {
  PWCHAR m__PWCHAR__DirName;
  PVOID m__PVOID__ParentContext;
} UDF_LOOKUP_DIR_ENTRY, * PUDF_LOOKUP_DIR_ENTRY;
```

File

StarBurn.h (see page 662)

Description

Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function

Member	Definition
m__PWCHAR__DirName	Directory name
m__PVOID__ParentContext	The user context assigned to this directory

2.2.135 PUDF_LOOKUP_FILE_ENTRY Structure

C++

```
typedef struct _UDF_LOOKUP_FILE_ENTRY {
  ULONG m__ULONG__EntrySize;
  PWCHAR m__PWCHAR__FileName;
  ULONGLONG m__QWORD__FileSize;
  UINT m__UINT__ExtentCount;
  PUDF_FILE_EXTENT m__Extents;
} UDF_LOOKUP_FILE_ENTRY, * PUDF_LOOKUP_FILE_ENTRY;
```

File

StarBurn.h (see page 662)

Description

Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function

Member	Definition
m__ULONG__EntrySize	Entry size
m__PWCHAR__FileName	File name
m__QWORD__FileSize	File size in bytes
m__UINT__ExtentCount	Number of file extents
m__Extents	Array of file extents

2.2.136 PUDF_TREE_ITEM Structure

C++

```
typedef struct _UDF_TREE_ITEM {
  unsigned long m__ULONG__FileEntryRBA;
  unsigned long m__ULONG__FileIdentifierRBA;
  unsigned long m__ULONG__FileIdentifierParentOrContentRBA;
  unsigned long m__ULONG__LastTouchedRBA;
  unsigned long m__ULONG__GUID;
  unsigned char m__UCHAR__IsDirectory;
  unsigned char m__UCHAR__IsCached;
  unsigned short m__USHORT__NumberOfKidsAsParents;
  char m__CHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];
  UDF_FILE_HANDLE m__UDF_FILE_HANDLE;
  unsigned char * m__PUCHAR__File;
  unsigned __int64 m__ULONGLONG__SizeInUCHARs;
  unsigned long m__ULONG__SizeInLogicalBlocks;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Next;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Prev;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Kids;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Parent;
  unsigned char m__UCHAR__FileEntryDescriptor[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];
  unsigned char * m__PUCHAR__FileIdentifierDescriptor;
  unsigned long m__ULONG__FileIdentifierDescriptorSizeInUCHARs;
  unsigned char m__UCHAR__FileContent[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];
  void * m__PVOID__Context;
  ISO9660_DATE_TIME m__ISO9660_DATE_TIME;
  WCHAR m__WCHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];
} * PUDF_TREE_ITEM, UDF_TREE_ITEM;
```

File

StarBurn.h (see page 662)

Members

Members	Description
unsigned long m__ULONG__FileEntryRBA;	File entry relative block address
unsigned long m__ULONG__FileIdentifierRBA;	File identifier relative block address
unsigned long m__ULONG__FileIdentifierParentOrContentRBA;	File identifier parent or content relative block address
unsigned long m__ULONG__LastTouchedRBA;	Last touched relative block address (last occupied)
unsigned long m__ULONG__GUID;	Globally unique identifier
unsigned char m__UCHAR__IsDirectory;	Is this directory (0x01) or file (0x00)
unsigned char m__UCHAR__IsCached;	Is this entry content cached (located in memory) or not cached (located on the disk)
unsigned short m__USHORT__NumberOfKidsAsParents;	Number of kids that have their own kids
char m__CHAR__Name[UDF_NAME_SIZE_IN_UCHARS];	Name of this node
UDF_FILE_HANDLE m__UDF_FILE_HANDLE;	UDF file handle of this node
unsigned char * m__PUCHAR__File;	Pointer for file content (for cached files)
unsigned __int64 m__ULONGLONG__SizeInUCHARs;	Node content size in UCHARs
unsigned long m__ULONG__SizeInLogicalBlocks;	Node content size in logical blocks
struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Next;	Pointer to the next UDF tree item in the linked list
struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Prev;	Pointer to the previous UDF tree item in the linked list
struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Kids;	Pointer to the kids linked list
struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Parent;	Pointer to the parent of the current UDF tree item
unsigned char m__UCHAR__FileEntryDescriptor[UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS];	Array of UCHARs holding UDF file entry descriptor for current UDF tree item
unsigned char * m__PUCHAR__FileIdentifierDescriptor;	Pointer to allocated file identifier
unsigned long m__ULONG__FileIdentifierDescriptorSizeInUCHARs;	Size of allocated FID in bytes
unsigned char m__UCHAR__FileContent[UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS];	Array of UCHARs holding file content (alternative cached data)
void * m__PVOID__Context;	Pointer to context value

ISO9660_DATE_TIME m__ISO9660_DATE_TIME;	ISO9660 date and time
WCHAR m__WCHAR__Name[UDF_NAME_SIZE_IN_UCHARS];	Name of this node

Description

Structure that represents UDF tree item

Member	Definition
m__ULONG__FileEntryRBA	File entry relative block address
m__ULONG__FileIdentifierRBA	File identifier relative block address
m__ULONG__FileIdentifier...	File identifier parent or content relative block address
m__ULONG__LastTouchedRBA	Last touched relative block address (last occupied)
m__ULONG__GUID	Globally unique identifier
m__UCHAR__IsDirectory	Is this directory (0x01) or file (0x00)
m__UCHAR__IsCached	Is this entry content cached (located in memory) or not cached (located on the disk)
m__USHORT_NumberOfKidsAsParents	Number of kids that have their own kids
m__CHAR__Name	Name of this node
m__UDF_FILE_HANDLE	UDF file handle of this node
m__PUCHAR__File	Pointer to file content (for cached files)
m__ULONGLONG__SizeInUCHARs	Node content size in UCHARs
m__ULONG__SizeInLogicalBlocks	Node content size in logical blocks
m__PUDF_TREE_ITEM__Next	Pointer to the next UDF tree item in the linked list
m__PUDF_TREE_ITEM__Prev	Pointer to the previous UDF tree item in the linked list
m__PUDF_TREE_ITEM__Kids	Pointer to the kids liked list
m__PUDF_TREE_ITEM__Parent	Pointer to the parent of the current UDF tree item
m__UCHAR__FileEntryDescriptor	Array if UCHARs holding UDF file entry descriptor for current UDF tree item
m__UCHAR__FileIdentifier...	Array of UCHARs holding UDF file identifier descriptor for current UDF tree item
m__UCHAR__FileIdentifier...	Array of UCHARs holding UDF file identifier descriptor for parent of the current UDF tree item
m__UCHAR__FileContent	Array of UCHARs holding file content (alternative cached data)
m__PVOID__Context	Pointer to context value
m__ISO9660_DATE_TIME	ISO9660 date and time

2.2.137 PWAVE_FILE_HEADER Structure

C++

```
typedef struct _WAVE_FILE_HEADER {
  ULONG m__ULONG__Riff;
  ULONG m__ULONG__Length;
  ULONG m__ULONG__Wave;
  ULONG m__ULONG__FormatTag;
  ULONG m__ULONG__FormatLength;
  WAVE_FORMAT_CHUNK m__WAVE_FORMAT_CHUNK;
  ULONG m__ULONG__DataTag;
  ULONG m__ULONG__DataLength;
} * PWAVE_FILE_HEADER, WAVE_FILE_HEADER;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`ULONG m__ULONG__Riff;`	Riff signature
`ULONG m__ULONG__Length;`	Length of data section (w/o header) in UCHARs
`ULONG m__ULONG__Wave;`	Wave signature
`ULONG m__ULONG__FormatTag;`	Format tag
`ULONG m__ULONG__FormatLength;`	Format length (size of WAVE_FORMAT_CHUNK (see page 580) in UCHARs)
`WAVE_FORMAT_CHUNK m__WAVE_FORMAT_CHUNK;`	WAVE format chunk (see WAVE_FORMAT_CHUNK (see page 580) for more details)
`ULONG m__ULONG__DataTag;`	Data tag
`ULONG m__ULONG__DataLength;`	Data length in UCHARs

Description

Structure that represents WAVE file header

Member	Definition
m__ULONG__Riff	Riff signature
m__ULONG__Length	Length of data section (w/o header) in UCHARs
m__ULONG__Wave	Wave signature
m__ULONG__FormatTag	Format tag
m__ULONG__FormatLength	Format length (size of WAVE_FORMAT_CHUNK (see page 580) in UCHARs)
m__WAVE_FORMAT_CHUNK	WAVE format chunk (see WAVE_FORMAT_CHUNK (see page 580) for more details)
m__ULONG__DataTag	Data tag
m__ULONG__DataLenght	Data length in UCHARs

2.2.138 PWAVE_FORMAT_CHUNK Structure

C++

```
typedef struct _WAVE_FORMAT_CHUNK {
  USHORT m__USHORT__Format;
  USHORT m__USHORT__Channels;
  ULONG m__ULONG__SamplesPerSecond;
  ULONG m__ULONG__AverageSamplesPerSecond;
  USHORT m__USHORT__Alignment;
  USHORT m__USHORT__BitsPerSample;
} * PWAVE_FORMAT_CHUNK, WAVE_FORMAT_CHUNK;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`USHORT m__USHORT__Format;`	Format
`USHORT m__USHORT__Channels;`	Number of channels
`ULONG m__ULONG__SamplesPerSecond;`	Number of samples per second
`ULONG m__ULONG__AverageSamplesPerSecond;`	Number of average samples per second

USHORT m__USHORT__Alignment;	Alignment
USHORT m__USHORT__BitsPerSample;	Number of bits in single sample

Description

Structure that represents WAVE format chunk

Member	Definition
m__USHORT__Format	Format
m__USHORT__Channels	Number of channels
m__ULONG__SamplesPerSecond	Number of samples per second
m__ULONG__AverageSamplesPerSecond	Number of average samples per second
m__USHORT__Alignment	Alignment
m__USHORT__BitsPerSample	Number of bits in single sample

2.2.139 PWRITE_MODE Enumeration

C++

```
typedef enum _WRITE_MODE {
  WRITE_MODE_TRACK_AT_ONCE = 0,
  WRITE_MODE_SESSION_AT_ONCE,
  WRITE_MODE_DISC_AT_ONCE_PQ,
  WRITE_MODE_DISC_AT_ONCE_RAW_PW
} * PWRITE_MODE, WRITE_MODE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
WRITE_MODE_TRACK_AT_ONCE = 0	Track-At-Once
WRITE_MODE_SESSION_AT_ONCE	Session-At-Once
WRITE_MODE_DISC_AT_ONCE_PQ	Disc-At-Once PQ
WRITE_MODE_DISC_AT_ONCE_RAW_PW	Disc-At-Once raw P-W

Description

Enum that represents write modes

Member	Definition
WRITE_MODE_TRACK_AT_ONCE	Track-At-Once
WRITE_MODE_SESSION_AT_ONCE	Session-At-Once
WRITE_MODE_DISC_AT_ONCE_PQ	Disc-At-Once PQ
WRITE_MODE_DISC_AT_ONCE_RAW_PW	Disc-At-Once raw P-W

2.2.140 READ_MODE Enumeration

C++

```
typedef enum _READ_MODE {
  READ_MODE_COOKED = 0,
  READ_MODE_RAW,
  READ_MODE_RAW_PQ,
  READ_MODE_RAW_RAW_PW,
  READ_MODE_PQ,
  READ_MODE_RAW_PW
} * PREAD_MODE, READ_MODE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
READ_MODE_COOKED = 0	Cooked data
READ_MODE_RAW	Raw data
READ_MODE_RAW_PQ	Raw data + PQ sub-channel
READ_MODE_RAW_RAW_PW	Raw data + raw P-W sub-channel
READ_MODE_PQ	PQ sub-channel only (no main channel data)
READ_MODE_RAW_PW	Raw P-W sub-channel only (no main channel data)

Description

Enum that represents raw read modes

Member	Definition
READ_MODE_COOKED	Cooked data
READ_MODE_RAW	Raw data
READ_MODE_RAW_PQ	Raw data + PQ sub-channel
READ_MODE_RAW_RAW_PW	Raw data + raw P-W sub-channel
READ_MODE_PQ	PQ sub-channel only (no main channel data)
READ_MODE_RAW_PW	Raw P-W sub-channel only (no main channel data)

2.2.141 SCSI_DEVICE_ADDRESS Structure

C++

```
typedef struct _SCSI_DEVICE_ADDRESS {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__PortID;
  UCHAR m__UCHAR__BusID;
  UCHAR m__UCHAR__TargetID;
  UCHAR m__UCHAR__LUN;
} * PSCSI_DEVICE_ADDRESS, SCSI_DEVICE_ADDRESS;
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsValid;	Is this data valid (structure was really filled)
UCHAR m__UCHAR__PortID;	Port Id - Logical SCSI adapter ID if ASPI is used
UCHAR m__UCHAR__BusID;	Bus ID - 0 if ASPI is used
UCHAR m__UCHAR__TargetID;	Target Id
UCHAR m__UCHAR__LUN;	LUN (Logical Unit Number)

Description

Structure that represents SCSI device address

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__PortID	Port ID - Logical SCSI adapter ID if ASPI is used
m__UCHAR__BusID	Bus ID - 0 if ASPI is used
m__UCHAR__TargetID	Target ID
m__UCHAR__LUN	LUN (Logical Unit Number)

2.2.142
STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS Structure

C++

```
typedef struct _STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS {
  ULONG m__ULONG__SizeInUCHARs;
  BOOLEAN m__BOOLEAN__IsCDROMRead;
  BOOLEAN m__BOOLEAN__IsCDRRead;
  BOOLEAN m__BOOLEAN__IsCDRWrite;
  BOOLEAN m__BOOLEAN__IsCDRWRead;
  BOOLEAN m__BOOLEAN__IsCDRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDROMRead;
  BOOLEAN m__BOOLEAN__IsDVDRRead;
  BOOLEAN m__BOOLEAN__IsDVDRWrite;
  BOOLEAN m__BOOLEAN__IsDVDRDLRead;
  BOOLEAN m__BOOLEAN__IsDVDRDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDRWRead;
  BOOLEAN m__BOOLEAN__IsDVDRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDRWDLRead;
  BOOLEAN m__BOOLEAN__IsDVDRWDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDRAMRead;
  BOOLEAN m__BOOLEAN__IsDVDRAMWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead;
  BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite;
  BOOLEAN m__BOOLEAN__IsBDROMRead;
  BOOLEAN m__BOOLEAN__IsBDRRead;
  BOOLEAN m__BOOLEAN__IsBDRWrite;
  BOOLEAN m__BOOLEAN__IsBDRDLRead;
  BOOLEAN m__BOOLEAN__IsBDRDLWrite;
  BOOLEAN m__BOOLEAN__IsBDRERead;
```

```
  BOOLEAN m__BOOLEAN__IsBDREWrite;
  BOOLEAN m__BOOLEAN__IsBDREDLRead;
  BOOLEAN m__BOOLEAN__IsBDREDLWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDROMRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRDLRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRWRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWWrite;
  BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead;
  BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite;
} * PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS, STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS;
```

File

StarBurn.h (see page 662)

Members

Members	Description
ULONG m__ULONG__SizeInUCHARs;	This structure size in UCHARs
BOOLEAN m__BOOLEAN__IsCDROMRead;	Is CD-ROM read capable
BOOLEAN m__BOOLEAN__IsCDRRead;	Is CD-R read capable
BOOLEAN m__BOOLEAN__IsCDRWrite;	Is CD-R write capable
BOOLEAN m__BOOLEAN__IsCDRWRead;	Is CD-RW read capable
BOOLEAN m__BOOLEAN__IsCDRWWrite;	Is CD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDROMRead;	Is DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsDVDRRead;	Is DVD-R read capable
BOOLEAN m__BOOLEAN__IsDVDRWrite;	Is DVD-R write capable
BOOLEAN m__BOOLEAN__IsDVDRDLRead;	Is DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsDVDRDLWrite;	Is DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsDVDRWRead;	Is DVD-RW read capable
BOOLEAN m__BOOLEAN__IsDVDRWWrite;	Is DVD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDRWDLRead;	Is DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDRWDLWrite;	Is DVD-RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDRAMRead;	Is DVD-RAM read capable
BOOLEAN m__BOOLEAN__IsDVDRAMWrite;	Is DVD-RAM write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead;	Is DVD+RW read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite;	Is DVD+RW write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead;	Is DVD+RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite;	Is DVD+RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRRead;	Is DVD+R read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite;	Is DVD+R write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead;	Is DVD+R DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite;	Is DVD+R DL write capable
BOOLEAN m__BOOLEAN__IsBDROMRead;	Is BD-ROM read capable
BOOLEAN m__BOOLEAN__IsBDRRead;	Is BD-R read capable
BOOLEAN m__BOOLEAN__IsBDRWrite;	Is BD-R write capable
BOOLEAN m__BOOLEAN__IsBDRDLRead;	Is BD-R DL read capable
BOOLEAN m__BOOLEAN__IsBDRDLWrite;	Is BD-R DL write capable
BOOLEAN m__BOOLEAN__IsBDRERead;	Is BD-RE read capable
BOOLEAN m__BOOLEAN__IsBDREWrite;	Is BD-RE write capable
BOOLEAN m__BOOLEAN__IsBDREDLRead;	Is BD-RE DL read capable
BOOLEAN m__BOOLEAN__IsBDREDLWrite;	Is BD-RE DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDROMRead;	Is HD-DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsHDDVDRRead;	Is HD-DVD-R read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWrite;	Is HD-DVD-R write capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLRead;	Is HD-DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite;	Is HD-DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWRead;	Is HD-DVD-RW read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWWrite;	Is HD-DVD-RW write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead;	Is HD-DVD-RW DL read capable

BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite;	Is HD-DVD-RW DL write capable

Description

Structure that represents advanced supported media formats

Member Definition
--------- ----------
ULONG m__ULONG__SizeInUCHARs This structure size in UCHARs
BOOLEAN m__BOOLEAN__IsCDROMRead Is CD-ROM read capable
BOOLEAN m__BOOLEAN__IsCDRRead Is CD-R read capable
BOOLEAN m__BOOLEAN__IsCDRWrite Is CD-R write capable
BOOLEAN m__BOOLEAN__IsCDRWRead Is CD-RW read capable
BOOLEAN m__BOOLEAN__IsCDRWWrite Is CD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDROMRead Is DVD-ROM read capable
BOOLEAN m__BOOLEAN__IsDVDRRead Is DVD-R read capable
BOOLEAN m__BOOLEAN__IsDVDRWrite Is DVD-R write capable
BOOLEAN m__BOOLEAN__IsDVDRDLRead Is DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsDVDRDLWrite Is DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsDVDRWRead Is DVD-RW read capable
BOOLEAN m__BOOLEAN__IsDVDRWWrite Is DVD-RW write capable
BOOLEAN m__BOOLEAN__IsDVDRWDLRead Is DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDRWDLWrite Is DVD-RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDRAMRead Is DVD-RAM read capable
BOOLEAN m__BOOLEAN__IsDVDRAMWrite Is DVD-RAM write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWRead Is DVD+RW read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWWrite Is DVD+RW write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLRead Is DVD+RW DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWDLWrite Is DVD+RW DL write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRRead Is DVD+R read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRWrite Is DVD+R write capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLRead Is DVD+R DL read capable
BOOLEAN m__BOOLEAN__IsDVDPLUSRDLWrite Is DVD+R DL write capable
BOOLEAN m__BOOLEAN__IsBDROMRead Is BD-ROM read capable
BOOLEAN m__BOOLEAN__IsBDRRead Is BD-R read capable
BOOLEAN m__BOOLEAN__IsBDRWrite Is BD-R write capable
BOOLEAN m__BOOLEAN__IsBDRDLRead Is BD-R DL read capable
BOOLEAN m__BOOLEAN__IsBDRDLWrite Is BD-R DL write capable
BOOLEAN m__BOOLEAN__IsBDRERead Is BD-RE read capable
BOOLEAN m__BOOLEAN__IsBDREWrite Is BD-RE write capable
BOOLEAN m__BOOLEAN__IsBDREDLRead Is BD-RE DL read capable
BOOLEAN m__BOOLEAN__IsBDREDLWrite Is BD-RE DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDROMMRead Is HD-DVD-ROM read capable

BOOLEAN m__BOOLEAN__IsHDDVDRRead Is HD-DVD-R read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWrite Is HD-DVD-R write capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLRead Is HD-DVD-R DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRDLWrite Is HD-DVD-R DL write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWRead Is HD-DVD-RW read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWWrite Is HD-DVD-RW write capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLRead Is HD-DVD-RW DL read capable
BOOLEAN m__BOOLEAN__IsHDDVDRWDLWrite Is HD-DVD-RW DL write capable

2.2.143 STARBURN_BDRE_FORMAT_PROFILE Structure

C++

```
typedef struct _STARBURN_BDRE_FORMAT_PROFILE {
  LONG m__LONG__FormatProfileNumber;
  ULONG m__ULONG__NumberOfBlocksOrUserAreaDataSize;
  UCHAR m__UCHAR__CertificationType : 2;
  UCHAR m__UCHAR__FormatType : 6;
  ULONG m__ULONG__SpareAreaSizeInClustersOrBlockLength;
} ** PPSTARBURN_BDRE_FORMAT_PROFILE, * PSTARBURN_BDRE_FORMAT_PROFILE,
STARBURN_BDRE_FORMAT_PROFILE;
```

File

StarBurn.h (see page 662)

Description

Structure that represents format profile for BD-RE media

Member Definition
--------- ----------
LONG m__LONG__FormatProfileNumber Format profile number
UCHAR m__ULONG__NumberOfBlocksOrUserAreaDataSize Number of logical block(s) or user area data size
UCHAR m__UCHAR__CertificationType Certification type
UCHAR m__UCHAR__FormatType Format type
UCHAR m__ULONG__SpareAreaSizeInClustersOrBlockLength Spare area size in cluster(s) or block length

2.2.144 STARBURN_CD_MODE Enumeration

C++

```
typedef enum _STARBURN_CD_MODE {
  CD_MODE_UNKNOWN = 0,
  CD_MODE_AUDIO,
  CD_MODE_VIDEO,
  CD_MODE_DATA,
  CD_MODE_RESERVED
} * PSTARBURN_CD_MODE, STARBURN_CD_MODE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
CD_MODE_UNKNOWN = 0	Unknown CD mode
CD_MODE_AUDIO	Audio CD mode
CD_MODE_VIDEO	Video CD mode
CD_MODE_DATA	Data CD mode
CD_MODE_RESERVED	Reserved

Description

Enum that represents CD modes type

Member	Definition
CD_MODE_UNKNOWN	Unknown CD mode
CD_MODE_AUDIO	Audio CD mode
CD_MODE_VIDEO	Video CD mode
CD_MODE_DATA	Data CD mode
CD_MODE_RESERVED	Reserved

2.2.145 STARBURN_DISC_ATIP_INFORMATION Structure

C++

```
typedef struct _STARBURN_DISC_ATIP_INFORMATION {
  USHORT m__USHORT__DataLength;
  UCHAR m__UCHAR__Reserved1[2];
  UCHAR m__UCHAR__RefrenceSpeed : 3;
  UCHAR m__UCHAR__Reserved2 : 1;
  UCHAR m__UCHAR__IndicativeTargetWritingPower : 4;
  UCHAR m__UCHAR__Reserved3 : 6;
  UCHAR m__UCHAR__URU : 1;
  UCHAR m__UCHAR__Zero : 1;
  UCHAR m__UCHAR__IsA3Valid : 1;
  UCHAR m__UCHAR__IsA2Valid : 1;
  UCHAR m__UCHAR__IsA1Valid : 1;
  UCHAR m__UCHAR__DiscSubType : 3;
  UCHAR m__UCHAR__DiscType : 1;
  UCHAR m__UCHAR__One : 1;
  UCHAR m__UCHAR__Reserved4;
  UCHAR m__UCHAR__LeadInStartTime[3];
  UCHAR m__UCHAR__Reserved5;
  UCHAR m__UCHAR__LastPossibleLeadOutStartTime[3];
  UCHAR m__UCHAR__Reserved6;
  UCHAR m__UCHAR__A1Values[3];
  UCHAR m__UCHAR__Reserved7;
  UCHAR m__UCHAR__A2Values[3];
  UCHAR m__UCHAR__Reserved8;
  UCHAR m__UCHAR__A3Values[3];
  UCHAR m__UCHAR__Reserved9;
  UCHAR m__UCHAR__S4Values[3];
  UCHAR m__UCHAR__Reserved10;
} * PSTARBURN_DISC_ATIP_INFORMATION, STARBURN_DISC_ATIP_INFORMATION;
```

File

StarBurn.h (see page 662)

Description

Structure that represents Disc ATIP information

Member Definition
--- ---
m__USHORT__DataLength
m__UCHAR__Reserved1
m__UCHAR__RefrenceSpeed
m__UCHAR__Reserved2
m__UCHAR__IndicativeTargetWritingPower
m__UCHAR__Reserved3
m__UCHAR__URU
m__UCHAR__Zero
m__UCHAR__IsA3Valid
m__UCHAR__IsA2Valid
m__UCHAR__IsA1Valid
m__UCHAR__DiscSubType
m__UCHAR__DiscType
m__UCHAR__One
m__UCHAR__Reserved4
m__UCHAR__LeadInStartTime
m__UCHAR__Reserved5
m__UCHAR__LastPossibleLeadOutStartTime
m__UCHAR__Reserved6
m__UCHAR__A1Values
m__UCHAR__Reserved7
m__UCHAR__A2Values
m__UCHAR__Reserved8
m__UCHAR__A3Values
m__UCHAR__Reserved9
m__UCHAR__S4Values
m__UCHAR__Reserved10

2.2.146 STARBURN_DISC_INFORMATION Structure

C++

```
typedef struct _STARBURN_DISC_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__DiscStatus;
  UCHAR m__UCHAR__LastSessionStatus;
  BOOLEAN m__BOOLEAN__IsErasable;
  UCHAR m__UCHAR__FirstTrackNumber;
```

```
  UCHAR m__UCHAR__NumberOfSessions;
  UCHAR m__UCHAR__LastSessionFirstTrack;
  UCHAR m__UCHAR__LastSessionLastTrack;
  BOOLEAN m__BOOLEAN__IsGEN;
  BOOLEAN m__BOOLEAN__IsDBCValid;
  BOOLEAN m__BOOLEAN__IsDIDValid;
  UCHAR m__UCHAR__DiscType;
  UCHAR m__UCHAR__DiscIdentification[ 4 ];
  UCHAR m__UCHAR__LastSessionLeadIn[ 4 ];
  UCHAR m__UCHAR__LastPossibleStartTime[ 4 ];
  UCHAR m__UCHAR__DiscBarCode[ 8 ];
  UCHAR m__UCHAR__NumberOfOPCEntries;
} * PSTARBURN_DISC_INFORMATION, STARBURN_DISC_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__DiscStatus;`	Disc status (See DISC_STATUS_XXX constants)
`UCHAR m__UCHAR__LastSessionStatus;`	Last session status (See LAST_SESSION_XXX constants)
`BOOLEAN m__BOOLEAN__IsErasable;`	Is disc erasable (CD-RW or DVD-RW or DVD+RW or DVD-RAM)
`UCHAR m__UCHAR__FirstTrackNumber;`	First track number on the disc
`UCHAR m__UCHAR__NumberOfSessions;`	Number of sessions on the disc
`UCHAR m__UCHAR__LastSessionFirstTrack;`	First track number of last complete session on the disc
`UCHAR m__UCHAR__LastSessionLastTrack;`	Last track number of last complete session on the disc
`BOOLEAN m__BOOLEAN__IsGEN;`	Unrestricted use, when set to one - every application can write to the disc
`BOOLEAN m__BOOLEAN__IsDBCValid;`	Is Disc Bar Code field valid
`BOOLEAN m__BOOLEAN__IsDIDValid;`	Is Disc Identification field Valid
`UCHAR m__UCHAR__DiscType;`	Disc type (See DISC_TYPE_XXX constants)
`UCHAR m__UCHAR__DiscIdentification[ 4 ];`	Disc identification number recorded in the PMA, see m__UCHAR__IsDIDValid
`UCHAR m__UCHAR__LastSessionLeadIn[ 4 ];`	MSF of last session lead in start time
`UCHAR m__UCHAR__LastPossibleStartTime[ 4 ];`	Last possible start time for lead out on this disc
`UCHAR m__UCHAR__DiscBarCode[ 8 ];`	Disc bar code stored here if the device is capable of reading disc bar codes
`UCHAR m__UCHAR__NumberOfOPCEntries;`	Number of optimum power calibration entries at the end of this structure

Description

Structure that represents Disc information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__DiscStatus	Disc status (See DISC_STATUS_XXX constants)
m__UCHAR__LastSessionStatus	Last session status (See LAST_SESSION_XXX constants)
m__BOOLEAN__IsErasable	Is disc erasable (CD-RW or DVD-RW or DVD+RW or DVD-RAM)
m__UCHAR__FirstTrackNumber	First track number on the disc
m__UCHAR__NumberOfSessions	Number of sessions on the disc
m__UCHAR__LastSessionFirstTrack	First track number of last complete session on the disc
m__UCHAR__LastSessionLastTrack	Last track number of last complete session on the disc
m__BOOLEAN__IsGEN	Unrestricted use, when set to one - every application can write to the disc
m__BOOLEAN__IsDBCValid	Is Disc Bar Code field valid
m__BOOLEAN__IsDIDValid	Is Disc Identification field Valid
m__UCHAR__DiscType	Disc type (See DISC_TYPE_XXX constants)
m__UCHAR__DiscIdentification	Disc identification number recorded in the PMA, see m__UCHAR__IsDIDValid

m__UCHAR__LastSessionLeadId	MSF of last session lead in start time
m__UCHAR__LastPossibleStartTime	Last possible start time for lead out on this disc
m__UCHAR__DiscBarCode	Disc bar code stored here if the device is capable of reading disc bar codes
m__UCHAR__NumberOfOPCEntries	Number of optimum power calibration entries at the end of this structure

2.2.147 STARBURN_ELTORITO_MEDIA Enumeration

C++

```
typedef enum _STARBURN_ELTORITO_MEDIA {
  ELTORITO_MEDIA_CUSTOM = 0x00,
  ELTORITO_MEDIA_FLOPPY120 = 0x01,
  ELTORITO_MEDIA_FLOPPY144 = 0x02,
  ELTORITO_MEDIA_FLOPPY288 = 0x03,
  ELTORITO_MEDIA_HARDDISK = 0x04
} STARBURN_ELTORITO_MEDIA;
```

File

StarBurn.h (see page 662)

Members

Members	Description
ELTORITO_MEDIA_CUSTOM = 0x00	Custom bootable media emulation
ELTORITO_MEDIA_FLOPPY120 = 0x01	1.2 MB floppy bootable media emulation
ELTORITO_MEDIA_FLOPPY144 = 0x02	1.44 MB floppy bootable media emulation
ELTORITO_MEDIA_FLOPPY288 = 0x03	2.88 MB floppy bootable media emulation
ELTORITO_MEDIA_HARDDISK = 0x04	Hard disk bootable media emulation

Description

Media emulation types

2.2.148 STARBURN_ELTORITO_PLATFORM Enumeration

C++

```
typedef enum _STARBURN_ELTORITO_PLATFORM {
  ELTORITO_PLATFORM_80X86 = 0x00,
  ELTORITO_PLATFORM_POWERPC = 0x01,
  ELTORITO_PLATFORM_MAC = 0x02,
  ELTORITO_PLATFORM_EFI = 0xEF
} STARBURN_ELTORITO_PLATFORM;
```

File

StarBurn.h (see page 662)

Members

Members	Description
ELTORITO_PLATFORM_80X86 = 0x00	x86 platform identifier
ELTORITO_PLATFORM_POWERPC = 0x01	PowerPC platform identifier
ELTORITO_PLATFORM_MAC = 0x02	MAC platform identifier
ELTORITO_PLATFORM_EFI = 0xEF	EFI platform identifier

Description

ElTorito platform types

2.2.149 STARBURN_ISO9660_DIRECTORY_INFO Structure

C++

```
typedef struct _STARBURN_ISO9660_DIRECTORY_INFO {
  unsigned char ISOType;
  unsigned long NumberOfKids;
  unsigned long TotalNumberOfFiles;
  unsigned long TotalNumberOfDirs;
  unsigned __int64 TotalChildrenSizeInLBs;
} STARBURN_ISO9660_DIRECTORY_INFO;
```

File

StarBurn.h (see page 662)

Description

This is type STARBURN_ISO9660_DIRECTORY_INFO.

2.2.150 STARBURN_ISO9660_FILE_INFO Structure

C++

```
typedef struct _STARBURN_ISO9660_FILE_INFO {
  char Name[ ( STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS + 1 ) ];
  unsigned short UnicodeName[ ( STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS + 1 ) ];
  unsigned short UnicodePathName[ ( STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS + 1 ) ];
  BOOL IsImported;
  unsigned __int64 ContentSizeInUCHARs;
  unsigned long SizeInLogicalBlocks;
  unsigned long GUID;
  unsigned long Attributes;
  unsigned long BeginLBA;
  unsigned long EndLBA;
} STARBURN_ISO9660_FILE_INFO;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`unsigned long BeginLBA;`	File location on disc or in the image it is valid only for imported nodes i.e. when IsImported is TRUE

Description

ISO9660 file information

2.2.151 STARBURN_STARWAVE_CALLBACK_REASON Enumeration

C++

```
typedef enum _STARBURN_STARWAVE_CALLBACK_REASON {
  STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0,
  STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS
} STARBURN_STARWAVE_CALLBACK_REASON, * PSTARBURN_STARWAVE_CALLBACK_REASON, **
```

```
PPSTARBURN_STARWAVE_CALLBACK_REASON;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN = 0	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

Description

StarWave callback reasons we'll be using

Member	Definition
STARBURN_STARWAVE_CALLBACK_REASON_UNKNOWN	Unknown call reason
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Progress indication reason

2.2.152 STARBURN_STARWAVE_COMPRESSION Enumeration

C++

```
typedef enum _STARBURN_STARWAVE_COMPRESSION {
  STARBURN_STARWAVE_COMPRESSION_NONE = 0,
  STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90,
  STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000,
  STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000
} STARBURN_STARWAVE_COMPRESSION, * PSTARBURN_STARWAVE_COMPRESSION, **
PPSTARBURN_STARWAVE_COMPRESSION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARBURN_STARWAVE_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98

STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

Member	Definition
STARBURN_STARWAVE_COMPRESSION_NONE	Lossless PCM WAV uncompressed stream
STARBURN_STARWAVE_COMPRESSION_WMA_LOSS...	Lossless WMA compressed stream, VBR at Quality 100
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q10	WMA compressed stream, VBR at Quality 10
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q25	WMA compressed stream, VBR at Quality 25
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q50	WMA compressed stream, VBR at Quality 50
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q75	WMA compressed stream, VBR at Quality 75
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q90	WMA compressed stream, VBR at Quality 90
STARBURN_STARWAVE_COMPRESSION_WMA_VBR_Q98	WMA compressed stream, VBR at Quality 98
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_32K	WMA compressed stream, CBR at 32 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_48K	WMA compressed stream, CBR at 48 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_64K	WMA compressed stream, CBR at 64 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_80K	WMA compressed stream, CBR at 80 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_96K	WMA compressed stream, CBR at 96 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_128K	WMA compressed stream, CBR at 128 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_160K	WMA compressed stream, CBR at 160 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_192K	WMA compressed stream, CBR at 192 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_256K	WMA compressed stream, CBR at 256 Kbps
STARBURN_STARWAVE_COMPRESSION_WMA_CBR_320K	WMA compressed stream, CBR at 320 Kbps

2.2.153 STARBURN_STARWAVE2_COMPRESS_TYPE Enumeration

C++

```
typedef enum _STARBURN_STARWAVE2_COMPRESS_TYPE {
  COMPRESS_TYPE_INVALID_VALUE = 0,
  COMPRESS_TYPE_MP3_CBR,
  COMPRESS_TYPE_MP3_ABR,
  COMPRESS_TYPE_MP3_VBR,
  COMPRESS_TYPE_OGG_CBR,
  COMPRESS_TYPE_OGG_ABR,
  COMPRESS_TYPE_OGG_VBR,
```

```
  COMPRESS_TYPE_OGG_APPROXIMATE,
  COMPRESS_TYPE_WMA_LOSSLESS_VBR,
  COMPRESS_TYPE_WMA_CBR,
  COMPRESS_TYPE_WMA_VBR,
  COMPRESS_TYPE_UNCOMPRESSED_WAV_44100_STEREO,
  COMPRESS_TYPE_CUSTOM,
  COMPRESS_TYPE_FORCE_INT = 0xFFFFFFFF
} STARBURN_STARWAVE2_COMPRESS_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
COMPRESS_TYPE_INVALID_VALUE = 0	Bad type
COMPRESS_TYPE_MP3_CBR	MP3 Types
COMPRESS_TYPE_OGG_CBR	OGG Types
COMPRESS_TYPE_WMA_LOSSLESS_VBR	WMA Types
COMPRESS_TYPE_UNCOMPRESSED_WAV_44100_STEREO	WAV Types
COMPRESS_TYPE_CUSTOM	... Custom Types
COMPRESS_TYPE_FORCE_INT = 0xFFFFFFFF	Force type size to 4 bytes

Description

Compress settings

2.2.154 STARBURN_STARWAVE2_CONVERSION_MODE Enumeration

C++

```
typedef enum _STARBURN_STARWAVE2_CONVERSION_MODE {
  STARBURN_STARWAVE2_OGG_MODE0,
  STARBURN_STARWAVE2_OGG_MODE1,
  STARBURN_STARWAVE2_OGG_MODE2,
  STARBURN_STARWAVE2_OGG_MODE_MAX = STARBURN_STARWAVE2_OGG_MODE2,
  STARBURN_STARWAVE2_MP3_MODE0,
  STARBURN_STARWAVE2_MP3_MODE1,
  STARBURN_STARWAVE2_MP3_MODE2,
  STARBURN_STARWAVE2_MP3_MODE_MAX = STARBURN_STARWAVE2_MP3_MODE2
} STARBURN_STARWAVE2_CONVERSION_MODE, * PSTARBURN_STARWAVE2_CONVERSION_MODE;
```

File

StarBurn.h (see page 662)

Description

Structure that represents conversion mode

Member	Definition
OGG_MODE0	VBR, variable bit rate
OGG_MODE1	ABR, average bit rate
OGG_MODE2	CBR, constant bit rate
OGG_MODE_MAX	Now equal to OGG_MODE2
MP3_MODE0	CBR, constant bit rate
MP3_MODE1	ABR, average bit rate
MP3_MODE2	VBR, variable bit rate

MP3_MODE_MAX	Now equal to MP3_MODE2

2.2.155 STARBURN_STARWAVE2_INIT_PARAMS Structure

C++

```
typedef struct _STARBURN_STARWAVE2_INIT_PARAMS {
  STARBURN_STARWAVE2_CONVERSION_MODE m__STARBURN_STARWAVE2_CONVERSION_MODE;
  FLOAT m__FLOAT__flQuality;
  LONG m__LONG__nQuality;
  LONG m__LONG__maxBitrate;
  LONG m__LONG__nominalBitrate;
  LONG m_LONG__minBitrate;
  STARBURN_STARWAVE2_QUALITY_MODE m__STARBURN_STARWAVE2_QUALITY_MODE;
} STARBURN_STARWAVE2_INIT_PARAMS, * PSTARBURN_STARWAVE2_INIT_PARAMS;
```

File

StarBurn.h (see page 662)

Description

Structure that represents initialization parameters for audio conversion

Member	Definition
m__STARBURN_STARWAVE2_CONVERSION_MODE	Conversion mode
m__FLOAT__flQuality	Quality level from 0. (lo) to 1. (hi) (use for OGG_MODE0)
m__LONG__nQuality	Quality level from 1 to 9 (use for MP3_MODE2)
m__LONG__maxBitrate	Maximum bit rate (use for OGG_MODE1/OGG_MODE2)
m__LONG__nominalBitrate	Nominal bit rate (use for OGG_MODE1/OGG_MODE2/MP3_MODE0/MP3_MODE1)
m__LONG__minBitrate	Minimum bit rate (use for OGG_MODE1/OGG_MODE2)
m__StarBurn_StarWave2_QUALITY_MODE	Quality mode (use for /MP3_MODE0/MP3_MODE1/MP3_MODE2)

2.2.156 STARBURN_STARWAVE2_QUALITY_MODE Enumeration

C++

```
typedef enum _STARBURN_STARWAVE2_QUALITY_MODE {
  STARBURN_STARWAVE2_QM_FAST = 0,
  STARBURN_STARWAVE2_QM_STANDARD,
  STARBURN_STARWAVE2_QM_HIGH
} STARBURN_STARWAVE2_QUALITY_MODE, * PSTARBURN_STARWAVE2_QUALITY_MODE;
```

File

StarBurn.h (see page 662)

Description

Enum that represents MP3 quality for conversion

Member	Definition
StarBurn_StarWave2_QM_FAST	Fast mode
StarBurn_StarWave2_QM_STANDARD	Standard mode
StarBurn_StarWave2_QM_HIGH	High quality mode

2.2.157 STARBURN_TRACK_INFORMATION Structure

C++

```
typedef struct _STARBURN_TRACK_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__TrackMode;
  BOOLEAN m__BOOLEAN__IsCopy;
  BOOLEAN m__BOOLEAN__IsDamage;
  UCHAR m__UCHAR__DataMode;
  BOOLEAN m__BOOLEAN__IsFixedPacket;
  BOOLEAN m__BOOLEAN__IsPacket;
  BOOLEAN m__BOOLEAN__IsBlank;
  BOOLEAN m__BOOLEAN__IsReserved;
  BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;
  LONG m__LONG__TrackStartAddress;
  LONG m__LONG__NextWritableAddress;
  LONG m__LONG__FreeLBs;
  LONG m__LONG__FixedPacketSizeInLBs;
} * PSTARBURN_TRACK_INFORMATION, STARBURN_TRACK_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
BOOLEAN m__BOOLEAN__IsValid;	Is this data valid (structure was really filled)
UCHAR m__UCHAR__TrackNumber;	Track number of this track
UCHAR m__UCHAR__SessionNumber;	Session number that contains this track
UCHAR m__UCHAR__TrackMode;	Track mode
BOOLEAN m__BOOLEAN__IsCopy;	Is this track second or higher generation copy
BOOLEAN m__BOOLEAN__IsDamage;	Is track damaged
UCHAR m__UCHAR__DataMode;	Data mode on the track
BOOLEAN m__BOOLEAN__IsFixedPacket;	Is fixed packet used on this track
BOOLEAN m__BOOLEAN__IsPacket;	Is packet recording used on the track
BOOLEAN m__BOOLEAN__IsBlank;	Is track blank
BOOLEAN m__BOOLEAN__IsReserved;	Is track reserved
BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;	Is NWA (Next Writable Address) valid
LONG m__LONG__TrackStartAddress;	Starting address (LBA or MSF) for this track
LONG m__LONG__NextWritableAddress;	NWA (Next Writable Address)
LONG m__LONG__FreeLBs;	Free logical blocks on this track
LONG m__LONG__FixedPacketSizeInLBs;	Fixed packet size in LBs

Description

Structure that represents Track information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number of this track
m__UCHAR__SessionNumber	Session number that contains this track
m__UCHAR__TrackMode	Track mode
m__BOOLEAN__IsCopy	Is this track second or higher generation copy
m__BOOLEAN__IsDamage	Is track damaged
m__UCHAR__DataMode	Data mode on the track
m__BOOLEAN__IsFixedPacket	Is fixed packet used on this track
m__BOOLEAN__IsPacket	Is packet recording used on the track
m__BOOLEAN__IsBlank	Is track blank
m__BOOLEAN__IsReserved	Is track reserved
m__BOOLEAN__IsNextWritable...	Is NWA (Next Writable Address) valid
m__LONG__TrackStartAddress	Starting address (LBA or MSF) for this track
m__LONG__NextWritableAddress	NWA (Next Writable Address)
m__LONG__FreeLBs	Free logical blocks on this track
m__LONG__FixedPacketSize...	Fixed packet size in UCHARs

2.2.158 STARBURN_TRACK_INFORMATION_EX Structure

C++

```
typedef struct _STARBURN_TRACK_INFORMATION_EX {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  UCHAR m__UCHAR__TrackMode;
  BOOLEAN m__BOOLEAN__IsCopy;
  BOOLEAN m__BOOLEAN__IsDamage;
  UCHAR m__UCHAR__DataMode;
  BOOLEAN m__BOOLEAN__IsFixedPacket;
  BOOLEAN m__BOOLEAN__IsPacket;
  BOOLEAN m__BOOLEAN__IsBlank;
  BOOLEAN m__BOOLEAN__IsReserved;
  BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;
  LONG m__LONG__TrackStartAddress;
  LONG m__LONG__LastRecordedAddress;
  LONG m__LONG__NextWritableAddress;
  LONG m__LONG__FreeLBs;
  LONG m__LONG__FixedPacketSizeInLBs;
} * PSTARBURN_TRACK_INFORMATION_EX, STARBURN_TRACK_INFORMATION_EX;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__TrackNumber;`	Track number of this track
`UCHAR m__UCHAR__SessionNumber;`	Session number that contains this track
`UCHAR m__UCHAR__TrackMode;`	Track mode

BOOLEAN m__BOOLEAN__IsCopy;	Is this track second or higher generation copy
BOOLEAN m__BOOLEAN__IsDamage;	Is track damaged
UCHAR m__UCHAR__DataMode;	Data mode on the track
BOOLEAN m__BOOLEAN__IsFixedPacket;	Is fixed packet used on this track
BOOLEAN m__BOOLEAN__IsPacket;	Is packet recording used on the track
BOOLEAN m__BOOLEAN__IsBlank;	Is track blank
BOOLEAN m__BOOLEAN__IsReserved;	Is track reserved
BOOLEAN m__BOOLEAN__IsNextWritableAddressValid;	Is NWA (Next Writable Address) valid
LONG m__LONG__TrackStartAddress;	Starting address (LBA or MSF) for this track
LONG m__LONG__LastRecordedAddress;	Last Recorded Address (LBA or MSF) for this track
LONG m__LONG__NextWritableAddress;	NWA (Next Writable Address)
LONG m__LONG__FreeLBs;	Free logical blocks on this track
LONG m__LONG__FixedPacketSizeInLBs;	Fixed packet size in LBs

Description

Structure that represents Track information extended

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number of this track
m__UCHAR__SessionNumber	Session number that contains this track
m__UCHAR__TrackMode	Track mode
m__BOOLEAN__IsCopy	Is this track second or higher generation copy
m__BOOLEAN__IsDamage	Is track damaged
m__UCHAR__DataMode	Data mode on the track
m__BOOLEAN__IsFixedPacket	Is fixed packet used on this track
m__BOOLEAN__IsPacket	Is packet recording used on the track
m__BOOLEAN__IsBlank	Is track blank
m__BOOLEAN__IsReserved	Is track reserved
m__BOOLEAN__IsNextWritable...	Is NWA (Next Writable Address) valid
m__LONG__TrackStartAddress	Starting address (LBA or MSF) for this track
m__LONG__LastRecordedAddress	Last Recorded Address (LBA or MSF) for this track
m__LONG__NextWritableAddress	NWA (Next Writable Address)
m__LONG__FreeLBs	Free logical blocks on this track
m__LONG__FixedPacketSize...	Fixed packet size in UCHARs

2.2.159 STARBURN_UDF_BRIDGE_TYPE Enumeration

C++

```
typedef enum _STARBURN_UDF_BRIDGE_TYPE {
  UDF_BRIDGE_TYPE_NONE = 0,
  UDF_BRIDGE_TYPE_DVDVIDEO,
  UDF_BRIDGE_TYPE_ISO9660,
  UDF_BRIDGE_TYPE_ISO9660_JOLIET,
  UDF_BRIDGE_TYPE_FORCE_DWORD = 0xFFFFFFFF
} STARBURN_UDF_BRIDGE_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
UDF_BRIDGE_TYPE_NONE = 0	Pure UDF
UDF_BRIDGE_TYPE_DVDVIDEO	UDF - DVDVIDEO/ISO9660 Bridge
UDF_BRIDGE_TYPE_ISO9660	UDF - ISO9660 Bridge
UDF_BRIDGE_TYPE_ISO9660_JOLIET	UDF - ISO9660/Joliet Bridge
UDF_BRIDGE_TYPE_FORCE_DWORD = 0xFFFFFFFF	Force size to 4 Bytes

Description

Restore original structure packing

2.2.160 STARBURN_UDF2_DIRECTORY_INFO Structure

C++

```
typedef struct _STARBURN_UDF2_DIRECTORY_INFO {
  unsigned long NumberOfKids;
  unsigned long NumberOfKidsDirectories;
} STARBURN_UDF2_DIRECTORY_INFO;
```

File

StarBurn.h (see page 662)

Description

UDF directory information

2.2.161 STARBURN_UDF2_FILE_DATE_TIME Structure

C++

```
typedef struct _STARBURN_UDF2_FILE_DATE_TIME {
  unsigned short Year;
  unsigned char Month;
  unsigned char Day;
  unsigned char Hour;
  unsigned char Minute;
  unsigned char Second;
  unsigned char Millisecond;
  short Offset : 12;
  short IsLocal : 1;
  short Reserved : 3;
} STARBURN_UDF2_FILE_DATE_TIME;
```

File

StarBurn.h (see page 662)

Members

Members	Description
short Offset : 12;	UTC offset in minutes (applicable only if Type is set to 1)
short IsLocal : 1;	Flag, that indicates is it local time or UTC
short Reserved : 3;	Should be set to 0

Description

Structure that represents UDF2 file date and time

Member	Definition
Year	Year
Month	Month
Day	Day
Hour	Hour
Minute	Minute
Second	Second
Millisecnd	Millisecond
Offset	UTC offset in minutes

2.2.162 STARBURN_UDF2_FILE_DATE_TIME_TYPE Enumeration

C++

```
typedef enum _STARBURN_UDF2_FILE_DATE_TIME_TYPE {
  UDF_FILE_DATE_TIME_TYPE_CREATION = 0,
  UDF_FILE_DATE_TIME_TYPE_LAST_ACCESS,
  UDF_FILE_DATE_TIME_TYPE_LAST_WRITE
} STARBURN_UDF2_FILE_DATE_TIME_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
UDF_FILE_DATE_TIME_TYPE_CREATION = 0	Node originally created
UDF_FILE_DATE_TIME_TYPE_LAST_ACCESS	Node last "touched
UDF_FILE_DATE_TIME_TYPE_LAST_WRITE	Node last written to

Description

Enum that represents StarBurn date/time type for files in UDF

Member Definition
--------- ----------
UDF_FILE_DATE_TIME_TYPE_CREATION File originally created
UDF_FILE_DATE_TIME_TYPE_LAST_ACCESS File last accessed
UDF_FILE_DATE_TIME_TYPE_LAST_WRITE File last written to

2.2.163 STARBURN_UDF2_FILE_INFO Structure

C++

```
typedef struct _STARBURN_UDF2_FILE_INFO {
  CHAR Name[ ( STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS + 1 ) ];
```

```
  WCHAR UnicodeName[ ( STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS + 1 ) ];
  CHAR PathName[ ( STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS + 1 ) ];
  WCHAR UnicodePathName[ ( STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS + 1 ) ];
  BOOL IsUnicode;
  BOOL IsImported;
  unsigned __int64 ImportedContentSizeInUCHARs;
  unsigned long GUID;
  unsigned long Attributes;
  unsigned long SizeInLogicalBlocks;
  unsigned __int64 SizeInUCHARs;
  unsigned long BeginLBA;
  unsigned long EndLBA;
} STARBURN_UDF2_FILE_INFO;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`unsigned __int64 ImportedContentSizeInUCHARs;`	NAPALM_FILE_DATE_TIME DateTime;
`unsigned long BeginLBA;`	File location on disc or in the image it is valid only for imported nodes i.e. when IsImported is TRUE

Description

UDF file information

2.2.164 STARPORT_DEVICE_LIST Structure

C++

```
typedef struct _STARPORT_DEVICE_LIST {
  LONG m__LONG__NumberOfEntries;
  STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];
} ** PPSTARPORT_DEVICE_LIST, * PSTARPORT_DEVICE_LIST, STARPORT_DEVICE_LIST;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__NumberOfEntries;`	Number of StarPort device list entries
`STARPORT_DEVICE_LIST_ENTRY m__STARPORT_DEVICE_LIST_ENTRY[ 1 ];`	StarPort device list entries

Description

Structure that represents StarPort device list

Member	Definition
m__LONG__NumberOfEntries	Number of StarPort device list entries
m__STARPORT_DEVICE_LIST_ENTRY	StarPort device list entries

2.2.165 STARPORT_DEVICE_LIST_ENTRY Structure

C++

```
typedef struct _STARPORT_DEVICE_LIST_ENTRY {
```

```
  LONG m__LONG__Index;
  LONG m__LONG__TargetId;
  CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];
} ** PPSTARPORT_DEVICE_LIST_ENTRY, * PSTARPORT_DEVICE_LIST_ENTRY,
STARPORT_DEVICE_LIST_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`LONG m__LONG__Index;`	StarPort device index
`LONG m__LONG__TargetId;`	StarPort device TargetId
`CHAR m__CHAR__Name[ STARPORT_DEVICE_NAME_SIZE_IN_UCHARS ];`	StarPort device name

Description

Structure that represents StarPort device list entry

Member	Definition
m__LONG__Index	StarPort device index
m__LONG__TargetId	StarPort device TargetId
m__CHAR__Name	StarPort device name

2.2.166 STARPORT_DEVICE_TYPE Enumeration

C++

```
typedef enum _STARPORT_DEVICE_TYPE {
  STARPORT_DEVICE_TYPE_UNKNOWN = 0,
  STARPORT_DEVICE_TYPE_RAM,
  STARPORT_DEVICE_TYPE_HDD,
  STARPORT_DEVICE_TYPE_DVD,
  STARPORT_DEVICE_TYPE_AOE,
  STARPORT_DEVICE_TYPE_ISCSI
} ** PPSTARPORT_DEVICE_TYPE, * PSTARPORT_DEVICE_TYPE, STARPORT_DEVICE_TYPE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`STARPORT_DEVICE_TYPE_UNKNOWN = 0`	Unknown device type
`STARPORT_DEVICE_TYPE_RAM`	RAM disk
`STARPORT_DEVICE_TYPE_HDD`	Virtual hard disk
`STARPORT_DEVICE_TYPE_DVD`	Virtual DVD
`STARPORT_DEVICE_TYPE_AOE`	AoE (ATA-over-Ethernet)
`STARPORT_DEVICE_TYPE_ISCSI`	iSCSI (SCSI-over-IP)

Description

Enum that represents StarPort device type

Member	Definition
STARPORT_DEVICE_TYPE_UNKNOWN	Unknown device type
STARPORT_DEVICE_TYPE_RAM	RAM disk

STARPORT_DEVICE_TYPE_HDD	Virtual hard disk
STARPORT_DEVICE_TYPE_DVD	Virtual DVD
STARPORT_DEVICE_TYPE_AOE	AoE (ATA-over-Ethernet)
STARPORT_DEVICE_TYPE_ISCSI	iSCSI (SCSI-over-IP)

2.2.167 STARWAVE2_COMPRESSION Enumeration

C++

```
typedef enum _STARWAVE2_COMPRESSION {
  STARWAVE2_COMPRESSION_NONE = 0,
  STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1,
  STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10,
  STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25,
  STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50,
  STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75,
  STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90,
  STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98,
  STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000,
  STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000,
  STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000,
  STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000,
  STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000,
  STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000,
  STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000,
  STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000,
  STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000,
  STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000
} STARWAVE2_COMPRESSION, * PSTARWAVE2_COMPRESSION, ** PPSTARWAVE2_COMPRESSION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
STARWAVE2_COMPRESSION_NONE = 0	Lossless PCM WAV uncompressed stream
STARWAVE2_COMPRESSION_WMA_LOSSLESS_VBR_Q100 = 1	Lossless WMA compressed stream, VBR at Quality 100
STARWAVE2_COMPRESSION_WMA_VBR_Q10 = 10	WMA compressed stream, VBR at Quality 10
STARWAVE2_COMPRESSION_WMA_VBR_Q25 = 25	WMA compressed stream, VBR at Quality 25
STARWAVE2_COMPRESSION_WMA_VBR_Q50 = 50	WMA compressed stream, VBR at Quality 50
STARWAVE2_COMPRESSION_WMA_VBR_Q75 = 75	WMA compressed stream, VBR at Quality 75
STARWAVE2_COMPRESSION_WMA_VBR_Q90 = 90	WMA compressed stream, VBR at Quality 90
STARWAVE2_COMPRESSION_WMA_VBR_Q98 = 98	WMA compressed stream, VBR at Quality 98
STARWAVE2_COMPRESSION_WMA_CBR_32K = 32000	WMA compressed stream, CBR at 32 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_48K = 48000	WMA compressed stream, CBR at 48 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_64K = 64000	WMA compressed stream, CBR at 64 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_80K = 80000	WMA compressed stream, CBR at 80 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_96K = 96000	WMA compressed stream, CBR at 96 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_128K = 128000	WMA compressed stream, CBR at 128 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_160K = 160000	WMA compressed stream, CBR at 160 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_192K = 192000	WMA compressed stream, CBR at 192 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_256K = 256000	WMA compressed stream, CBR at 256 Kbps
STARWAVE2_COMPRESSION_WMA_CBR_320K = 320000	WMA compressed stream, CBR at 320 Kbps

Description

Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates

All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR

2.2.168 STARWAVE2_COMPRESSION_PROFILE Structure

C++

```
typedef struct _STARWAVE2_COMPRESSION_PROFILE {
  UINT m__UINT__CompressionType;
  PCHAR m__PCHAR__CustomFactoryID;
  union {
    int m__int__CBRBitRate;
    int m__int__VBRQuality;
  }
  int m__int__ABRBitRate;
  int m__int__MaxBitRate;
  union {
    int m__int__MinBitRate;
    int m__int__MaxFrameWindow;
  }
  LPVOID m__LPVOID__CustomData;
} STARWAVE2_COMPRESSION_PROFILE, * PSTARWAVE2_COMPRESSION_PROFILE, **
PPSTARWAVE2_COMPRESSION_PROFILE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`UINT m__UINT__CompressionType;`	(STARBURN_STARWAVE2_COMPRESS_TYPE (see page 560))
`int m__int__CBRBitRate;`	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320
`int m__int__VBRQuality;`	(MP3 compression 0..9) (WMA compression 1..100)
`int m__int__ABRBitRate;`	(MP3 && OGG compression)
`int m__int__MaxBitRate;`	32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320, for CBR mode this setting is ignored set this field to 0 if you don't want to use it
`int m__int__MinBitRate;`	(MP3 && OGG) 32, 40, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256 and 320 set this field to 0 if you don't want to use it
`int m__int__MaxFrameWindow;`	WMA, Frame size in milliseconds

2.2.169 TOC_ENTRY Structure

C++

```
typedef struct _TOC_ENTRY {
  BOOLEAN m__BOOLEAN__IsValid;
  UCHAR m__UCHAR__TrackNumber;
  UCHAR m__UCHAR__SessionNumber;
  LONG m__LONG__StartingLBA;
  UCHAR m__UCHAR__StartingMSF[ 3 ];
  LONG m__LONG__EndingLBA;
  UCHAR m__UCHAR__EndingMSF[ 3 ];
  BOOLEAN m__BOOLEAN__IsMCNAvailable;
  BOOLEAN m__BOOLEAN__IsISRCAvailable;
  BOOLEAN m__BOOLEAN__IsFourChannelsAudio;
  BOOLEAN m__BOOLEAN__IsPreEmphasisAudio;
  BOOLEAN m__BOOLEAN__IsData;
  BOOLEAN m__BOOLEAN__IsAudio;
  BOOLEAN m__BOOLEAN__IsDigitalCopyProhibited;
  UCHAR m__UCHAR__TrackMode;
  UCHAR m__UCHAR__MODE2Form;
  ULONG m__ULONG__LBSizeInUCHARs;
  LONG m__LONG__Index00;
} * PTOC_ENTRY, TOC_ENTRY;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`UCHAR m__UCHAR__TrackNumber;`	Track number
`UCHAR m__UCHAR__SessionNumber;`	Session number that this track belongs to
`LONG m__LONG__StartingLBA;`	Starting LBA
`UCHAR m__UCHAR__StartingMSF[ 3 ];`	Starting MSF
`LONG m__LONG__EndingLBA;`	Ending LBA
`UCHAR m__UCHAR__EndingMSF[ 3 ];`	Ending MSF
`BOOLEAN m__BOOLEAN__IsMCNAvailable;`	Is Media Catalog Number available
`BOOLEAN m__BOOLEAN__IsISRCAvailable;`	Is International Standard Recording Code available
`BOOLEAN m__BOOLEAN__IsFourChannelsAudio;`	Is this four channels audio track
`BOOLEAN m__BOOLEAN__IsPreEmphasisAudio;`	Is this pre-emphasis audio track
`BOOLEAN m__BOOLEAN__IsData;`	Is this data track
`BOOLEAN m__BOOLEAN__IsAudio;`	Is this audio track
`BOOLEAN m__BOOLEAN__IsDigitalCopyProhibited;`	Is digital copy prohibited for this track
`UCHAR m__UCHAR__TrackMode;`	Track mode (0 for audio track, 1 for MODE1 and 2 for MODE2)
`UCHAR m__UCHAR__MODE2Form;`	MODE2 Form (0 for Formless, 1 for Form1 and 2 for Form2)
`ULONG m__ULONG__LBSizeInUCHARs;`	LB (logical block) size in UCHARs on this track
`LONG m__LONG__Index00;`	Index00 LBA (if present)

Description

Structure that represents TOC entry

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__UCHAR__TrackNumber	Track number
m__UCHAR__SessionNumber	Session number that this track belongs to
m__LONG__StartingLBA	Starting LBA
m__UCHAR__StartingMSF	Starting MSF
m__LONG__EndingLBA	Ending LBA
m__UCHAR__EndingMSF	Ending MSF
m__BOOLEAN__IsMCNAvailable	Is Media Catalog Number available
m__BOOLEAN__IsISRCAvailable	Is International Standard Recording Code available
m__BOOLEAN__IsFourChannelsAudio	Is this four channels audio track
m__BOOLEAN__IsPreEmphasisAudio	Is this pre-emphasis audio track
m__BOOLEAN__IsData	Is this data track
m__BOOLEAN__IsAudio	Is this audio track
m__BOOLEAN__IsDigitalCopy...	Is digital copy prohibited for this track
m__UCHAR__TrackMode	Track mode (0 for audio track, 1 for MODE1 and 2 for MODE2)
m__UCHAR__MODE2Form	MODE2 Form (0 for Formless, 1 for Form1 and 2 for Form2)
m__ULONG__LBSizeInUCHARs	LB (logical block) size in UCHARs on this track
m__LONG__Index00	Index00 LBA (if present)

2.2.170 TOC_INFORMATION Structure

C++

```
typedef struct _TOC_INFORMATION {
  BOOLEAN m__BOOLEAN__IsValid;
  BOOLEAN m__BOOLEAN__IsDVD;
  USHORT m__USHORT__ProtectedDVDRegions;
  UCHAR m__UCHAR__BusKeyForDiscKey[ 5 ];
  UCHAR m__UCHAR__NumberOfSessions;
  UCHAR m__UCHAR__NumberOfTracks;
  UCHAR m__UCHAR__NumberOfUnsortedEntries;
  TOC_ENTRY m__TOC_ENTRY[ NUMBER_OF_TRACKS ];
  FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW[ NUMBER_OF_TRACKS ];
  FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW__Unsorted[ NUMBER_OF_RAW_TRACKS ];
} * PTOC_INFORMATION, TOC_INFORMATION;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`BOOLEAN m__BOOLEAN__IsValid;`	Is this data valid (structure was really filled)
`BOOLEAN m__BOOLEAN__IsDVD;`	Is this DVD media or not
`USHORT m__USHORT__ProtectedDVDRegions;`	Number of protected DVD regions
`UCHAR m__UCHAR__BusKeyForDiscKey[ 5 ];`	Bus key for disc key (CSS)
`UCHAR m__UCHAR__NumberOfSessions;`	Number of sessions
`UCHAR m__UCHAR__NumberOfTracks;`	Number of tracks
`UCHAR m__UCHAR__NumberOfUnsortedEntries;`	Number of entries in unsorted raw TOC
`TOC_ENTRY m__TOC_ENTRY[ NUMBER_OF_TRACKS ];`	TOC entries (decoded and extended)
`FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW[ NUMBER_OF_TRACKS ];`	Full TOC raw entries sorted, each entry corresponds to m__TOC_ENTRY with the same TNO
`FULL_TOC_ENTRY_RAW m__FULL_TOC_ENTRY_RAW__Unsorted[ NUMBER_OF_RAW_TRACKS ];`	Full TOC raw entries unsorted

Description

Structure that represents TOC information

Member	Definition
m__BOOLEAN__IsValid	Is this data valid (structure was really filled)
m__BOOLEAN__IsDVD	Is this DVD media or not
m__USHORT__ProtectedDVDRegions	Number of protected DVD regions
m__UCHAR__BusKeyForDiscKey	Bus key for disc key (CSS)
m__UCHAR__NumberOfSessions	Number of sessions
m__UCHAR__NumberOfTracks	Number of tracks
m__UCHAR__NumberOfUnsortedEntries	Number of entries in unsorted raw TOC
m__TOC_ENTRY	TOC entries (decoded and extended)
m__FULL_TOC_ENTRY_RAW	Full TOC raw entries sorted, each entry corresponds to m__TOC_ENTRY with the same TNO
m__FULL_TOC_ENTRY_RAW__Unsorted	Full TOC raw entries unsorted

2.2.171 UDF_CONTROL_BLOCK Structure

C++

```
typedef struct _UDF_CONTROL_BLOCK {
  void * m__PVOID__Head;
  void * m__PVOID__SystemStructures;
  void * m__PVOID__Tail;
  void * m__PVOID__Body;
} * PUDF_CONTROL_BLOCK, UDF_CONTROL_BLOCK;
```

File

StarBurn.h (see page 662)

Members

Members	Description
void * m__PVOID__Head;	Pointer to the head of the linked list
void * m__PVOID__SystemStructures;	Pointer to the allocated and formatted UDF system structures
void * m__PVOID__Tail;	Pointer to the tail of the linked list
void * m__PVOID__Body;	Pointer to block body itself

Description

Structure that represents UDF control block

Member	Definition
m__PVOID__Head	Pointer to the head of the linked list
m__PVOID__SystemStructures	Pointer to the allocated and formatted UDF system structures
m__PVOID__Tail	Pointer to the tail of the linked list
m__PVOID__Body	Pointer to block body itself

2.2.172 UDF_FILE_EXTENT Structure

C++

```
typedef struct _UDF_FILE_EXTENT {
  ULONG m__ULONG__BeginLBA;
  ULONG m__ULONG__EndLBA;
} UDF_FILE_EXTENT, * PUDF_FILE_EXTENT;
```

File

StarBurn.h (see page 662)

Description

Structure that describes a single file extent

Member	Definition
m__ULONG__BeginLBA	First LBA of the extent
m__ULONG__EndLBA	Last LBA of the extent

2.2.173 UDF_FILE_HANDLE Structure

C++

```
typedef struct _UDF_FILE_HANDLE {
  HANDLE m__HANDLE;
} * PUDF_FILE_HANDLE, UDF_FILE_HANDLE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
HANDLE m__HANDLE;	Win32 file handle

Description

Structure that represents UDF file handle

Member	Definition
m__HANDLE	Win32 file handle

2.2.174 UDF_FILE_LOOKUP_ENTRY Structure

C++

```
typedef struct _UDF_FILE_LOOKUP_ENTRY {
  ULONG m__ULONG__Size;
  PWCHAR m__PCHAR__FileName;
  UINT m__UINT__ExtentCount;
  PUDF_FILE_EXTENT m__Extents;
} UDF_FILE_LOOKUP_ENTRY, * PUDF_FILE_LOOKUP_ENTRY;
```

File

StarBurn.h (see page 662)

Description

Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function

Member	Definition
m__ULONG__Size	Entry size
m__PWCHAR__FileName	File name
m__UINT__ExtentCount	Number of file extents
m__Extents	Array of file extents

2.2.175 UDF_LOOKUP_DIR_ENTRY Structure

C++

```
typedef struct _UDF_LOOKUP_DIR_ENTRY {
  PWCHAR m__PWCHAR__DirName;
  PVOID m__PVOID__ParentContext;
} UDF_LOOKUP_DIR_ENTRY, * PUDF_LOOKUP_DIR_ENTRY;
```

File

StarBurn.h (see page 662)

Description

Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function

Member	Definition
m__PWCHAR__DirName	Directory name
m__PVOID__ParentContext	The user context assigned to this directory

2.2.176 UDF_LOOKUP_FILE_ENTRY Structure

C++

```
typedef struct _UDF_LOOKUP_FILE_ENTRY {
  ULONG m__ULONG__EntrySize;
  PWCHAR m__PWCHAR__FileName;
  ULONGLONG m__QWORD__FileSize;
  UINT m__UINT__ExtentCount;
  PUDF_FILE_EXTENT m__Extents;
} UDF_LOOKUP_FILE_ENTRY, * PUDF_LOOKUP_FILE_ENTRY;
```

File

StarBurn.h (see page 662)

Description

Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function

Member	Definition
m__ULONG__EntrySize	Entry size
m__PWCHAR__FileName	File name
m__QWORD__FileSize	File size in bytes
m__UINT__ExtentCount	Number of file extents
m__Extents	Array of file extents

2.2.177 UDF_OS_CLASS Enumeration

C++

```
typedef enum _UDF_OS_CLASS {
  UDF_OS_CLASS_UNDEFINED = 0x00,
  UDF_OS_CLASS_DOS = 0x01,
  UDF_OS_CLASS_OS_2 = 0x02,
  UDF_OS_CLASS_MACINTOSH_OS = 0x03,
  UDF_OS_CLASS_UNIX = 0x04,
  UDF_OS_CLASS_WINDOWS_9X = 0x05,
  UDF_OS_CLASS_WINDOWS_NT = 0x06,
  UDF_OS_CLASS_OS_400 = 0x07,
  UDF_OS_CLASS_BE_OS = 0x08,
  UDF_OS_CLASS_WINDOWS_CE = 0x09
} UDF_OS_CLASS;
```

File

StarBurn.h (see page 662)

Description

UDF OS class

2.2.178 UDF_TREE_ITEM Structure

C++

```
typedef struct _UDF_TREE_ITEM {
  unsigned long m__ULONG__FileEntryRBA;
  unsigned long m__ULONG__FileIdentifierRBA;
  unsigned long m__ULONG__FileIdentifierParentOrContentRBA;
  unsigned long m__ULONG__LastTouchedRBA;
  unsigned long m__ULONG__GUID;
  unsigned char m__UCHAR__IsDirectory;
  unsigned char m__UCHAR__IsCached;
  unsigned short m__USHORT__NumberOfKidsAsParents;
  char m__CHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];
  UDF_FILE_HANDLE m__UDF_FILE_HANDLE;
  unsigned char * m__PUCHAR__File;
  unsigned __int64 m__ULONGLONG__SizeInUCHARs;
  unsigned long m__ULONG__SizeInLogicalBlocks;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Next;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Prev;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Kids;
  struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Parent;
  unsigned char m__UCHAR__FileEntryDescriptor[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];
  unsigned char * m__PUCHAR__FileIdentifierDescriptor;
  unsigned long m__ULONG__FileIdentifierDescriptorSizeInUCHARs;
  unsigned char m__UCHAR__FileContent[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];
  void * m__PVOID__Context;
  ISO9660_DATE_TIME m__ISO9660_DATE_TIME;
  WCHAR m__WCHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];
} * PUDF_TREE_ITEM, UDF_TREE_ITEM;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`unsigned long m__ULONG__FileEntryRBA;`	File entry relative block address
`unsigned long m__ULONG__FileIdentifierRBA;`	File identifier relative block address

`unsigned long m__ULONG__FileIdentifierParentOrContentRBA;`	File identifier parent or content relative block address
`unsigned long m__ULONG__LastTouchedRBA;`	Last touched relative block address (last occupied)
`unsigned long m__ULONG__GUID;`	Globally unique identifier
`unsigned char m__UCHAR__IsDirectory;`	Is this directory (0x01) or file (0x00)
`unsigned char m__UCHAR__IsCached;`	Is this entry content cached (located in memory) or not cached (located on the disk)
`unsigned short m__USHORT__NumberOfKidsAsParents;`	Number of kids that have their own kids
`char m__CHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];`	Name of this node
`UDF_FILE_HANDLE m__UDF_FILE_HANDLE;`	UDF file handle of this node
`unsigned char * m__PUCHAR__File;`	Pointer for file content (for cached files)
`unsigned __int64 m__ULONGLONG__SizeInUCHARs;`	Node content size in UCHARs
`unsigned long m__ULONG__SizeInLogicalBlocks;`	Node content size in logical blocks
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Next;`	Pointer to the next UDF tree item in the linked list
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Prev;`	Pointer to the previous UDF tree item in the linked list
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Kids;`	Pointer to the kids linked list
`struct _UDF_TREE_ITEM * m__PUDF_TREE_ITEM__Parent;`	Pointer to the parent of the current UDF tree item
`unsigned char m__UCHAR__FileEntryDescriptor[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];`	Array of UCHARs holding UDF file entry descriptor for current UDF tree item
`unsigned char * m__PUCHAR__FileIdentifierDescriptor;`	Pointer to allocated file identifier
`unsigned long m__ULONG__FileIdentifierDescriptorSizeInUCHARs;`	Size of allocated FID in bytes
`unsigned char m__UCHAR__FileContent[ UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS ];`	Array of UCHARs holding file content (alternative cached data)
`void * m__PVOID__Context;`	Pointer to context value
`ISO9660_DATE_TIME m__ISO9660_DATE_TIME;`	ISO9660 date and time
`WCHAR m__WCHAR__Name[ UDF_NAME_SIZE_IN_UCHARS ];`	Name of this node

Description

Structure that represents UDF tree item

Member	Definition
m__ULONG__FileEntryRBA	File entry relative block address
m__ULONG__FileIdentifierRBA	File identifier relative block address
m__ULONG__FileIdentifier...	File identifier parent or content relative block address
m__ULONG__LastTouchedRBA	Last touched relative block address (last occupied)
m__ULONG__GUID	Globally unique identifier
m__UCHAR__IsDirectory	Is this directory (0x01) or file (0x00)
m__UCHAR__IsCached	Is this entry content cached (located in memory) or not cached (located on the disk)
m__USHORT_NumberOfKidsAsParents	Number of kids that have their own kids
m__CHAR__Name	Name of this node
m__UDF_FILE_HANDLE	UDF file handle of this node
m__PUCHAR__File	Pointer to file content (for cached files)
m__ULONGLONG__SizeInUCHARs	Node content size in UCHARs
m__ULONG__SizeInLogicalBlocks	Node content size in logical blocks
m__PUDF_TREE_ITEM__Next	Pointer to the next UDF tree item in the linked list
m__PUDF_TREE_ITEM__Prev	Pointer to the previous UDF tree item in the linked list
m__PUDF_TREE_ITEM__Kids	Pointer to the kids liked list
m__PUDF_TREE_ITEM__Parent	Pointer to the parent of the current UDF tree item
m__UCHAR__FileEntryDescriptor	Array if UCHARs holding UDF file entry descriptor for current UDF tree item
m__UCHAR__FileIdentifier...	Array of UCHARs holding UDF file identifier descriptor for current UDF tree item

m__UCHAR__FileIdentifier...	Array of UCHARs holding UDF file identifier descriptor for parent of the current UDF tree item
m__UCHAR__FileContent	Array of UCHARs holding file content (alternative cached data)
m__PVOID__Context	Pointer to context value
m__ISO9660_DATE_TIME	ISO9660 date and time

2.2.179 UDF_VERSION Enumeration

C++

```
typedef enum _UDF_VERSION {
  UDF_VERSION_1_02 = 0,
  UDF_VERSION_1_50,
  UDF_VERSION_2_00,
  UDF_VERSION_2_01,
  UDF_VERSION_2_50,
  UDF_VERSION_2_60
} UDF_VERSION;
```

File

StarBurn.h (see page 662)

Description

Enum that represents UDF version

Member	Definition
UDF_VERSION_1_02	UDF 1.02
UDF_VERSION_1_50	UDF 1.5
UDF_VERSION_2_00	UDF 2.0
UDF_VERSION_2_01	UDF 2.01
UDF_VERSION_2_50	UDF 2.5
UDF_VERSION_2_60	UDF 2.6

2.2.180 WAVE_FILE_HEADER Structure

C++

```
typedef struct _WAVE_FILE_HEADER {
  ULONG m__ULONG__Riff;
  ULONG m__ULONG__Length;
  ULONG m__ULONG__Wave;
  ULONG m__ULONG__FormatTag;
  ULONG m__ULONG__FormatLength;
  WAVE_FORMAT_CHUNK m__WAVE_FORMAT_CHUNK;
  ULONG m__ULONG__DataTag;
  ULONG m__ULONG__DataLength;
} * PWAVE_FILE_HEADER, WAVE_FILE_HEADER;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`ULONG m__ULONG__Riff;`	Riff signature
`ULONG m__ULONG__Length;`	Length of data section (w/o header) in UCHARs
`ULONG m__ULONG__Wave;`	Wave signature
`ULONG m__ULONG__FormatTag;`	Format tag
`ULONG m__ULONG__FormatLength;`	Format length (size of WAVE_FORMAT_CHUNK (see page 580) in UCHARs)
`WAVE_FORMAT_CHUNK m__WAVE_FORMAT_CHUNK;`	WAVE format chunk (see WAVE_FORMAT_CHUNK (see page 580) for more details)
`ULONG m__ULONG__DataTag;`	Data tag
`ULONG m__ULONG__DataLength;`	Data length in UCHARs

Description

Structure that represents WAVE file header

Member	Definition
m__ULONG__Riff	Riff signature
m__ULONG__Length	Length of data section (w/o header) in UCHARs
m__ULONG__Wave	Wave signature
m__ULONG__FormatTag	Format tag
m__ULONG__FormatLength	Format length (size of WAVE_FORMAT_CHUNK (see page 580) in UCHARs)
m__WAVE_FORMAT_CHUNK	WAVE format chunk (see WAVE_FORMAT_CHUNK (see page 580) for more details)
m__ULONG__DataTag	Data tag
m__ULONG__DataLenght	Data length in UCHARs

2.2.181 WAVE_FORMAT_CHUNK Structure

C++

```
typedef struct _WAVE_FORMAT_CHUNK {
  USHORT m__USHORT__Format;
  USHORT m__USHORT__Channels;
  ULONG m__ULONG__SamplesPerSecond;
  ULONG m__ULONG__AverageSamplesPerSecond;
  USHORT m__USHORT__Alignment;
  USHORT m__USHORT__BitsPerSample;
} * PWAVE_FORMAT_CHUNK, WAVE_FORMAT_CHUNK;
```

File

StarBurn.h (see page 662)

Members

Members	Description
`USHORT m__USHORT__Format;`	Format
`USHORT m__USHORT__Channels;`	Number of channels
`ULONG m__ULONG__SamplesPerSecond;`	Number of samples per second
`ULONG m__ULONG__AverageSamplesPerSecond;`	Number of average samples per second
`USHORT m__USHORT__Alignment;`	Alignment
`USHORT m__USHORT__BitsPerSample;`	Number of bits in single sample

Description

Structure that represents WAVE format chunk

Member	Definition
m__USHORT__Format	Format
m__USHORT__Channels	Number of channels
m__ULONG__SamplesPerSecond	Number of samples per second
m__ULONG__AverageSamplesPerSecond	Number of average samples per second
m__USHORT__Alignment	Alignment
m__USHORT__BitsPerSample	Number of bits in single sample

2.2.182 WRITE_MODE Enumeration

C++

```
typedef enum _WRITE_MODE {
  WRITE_MODE_TRACK_AT_ONCE = 0,
  WRITE_MODE_SESSION_AT_ONCE,
  WRITE_MODE_DISC_AT_ONCE_PQ,
  WRITE_MODE_DISC_AT_ONCE_RAW_PW
} * PWRITE_MODE, WRITE_MODE;
```

File

StarBurn.h (see page 662)

Members

Members	Description
WRITE_MODE_TRACK_AT_ONCE = 0	Track-At-Once
WRITE_MODE_SESSION_AT_ONCE	Session-At-Once
WRITE_MODE_DISC_AT_ONCE_PQ	Disc-At-Once PQ
WRITE_MODE_DISC_AT_ONCE_RAW_PW	Disc-At-Once raw P-W

Description

Enum that represents write modes

Member	Definition
WRITE_MODE_TRACK_AT_ONCE	Track-At-Once
WRITE_MODE_SESSION_AT_ONCE	Session-At-Once
WRITE_MODE_DISC_AT_ONCE_PQ	Disc-At-Once PQ
WRITE_MODE_DISC_AT_ONCE_RAW_PW	Disc-At-Once raw P-W

2.3 Types

The following table lists types in this documentation.

Types

Name	Description
CFuncStarWaveConversionCallback (see page 582)	type of callback function
PCALLBACK (see page 582)	Callback type (function of this type must be passed as callback)
PSTARBURN_CLOSEHANDLE_CALLBACK (see page 584)	This is type PSTARBURN_CLOSEHANDLE_CALLBACK.
PSTARBURN_CREATEFILE_CALLBACK (see page 584)	This is type PSTARBURN_CREATEFILE_CALLBACK.
PSTARBURN_DELETEFILE_CALLBACK (see page 585)	This is type PSTARBURN_DELETEFILE_CALLBACK.
PSTARBURN_FILE_OBJECT (see page 585)	Structure that represents StarBurn file object used internally with file I/O
PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK (see page 585)	This is type PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK.
PSTARBURN_ISO2_COLLISION_CALLBACK (see page 585)	StarBurn ISO name collision handling callback
PSTARBURN_ISO2_PROGRESS_CALLBACK (see page 586)	StarBurn ISO progress callback
PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK (see page 586)	StarBurn ISO Unicode name collision handling callback
PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK (see page 587)	StarBurn ISO UNICODE progress callback
PSTARBURN_ISO9660_READ_CALLBACK (see page 587)	StarBurn ISO2 read callback (for import)
PSTARBURN_READFILE_CALLBACK (see page 588)	This is type PSTARBURN_READFILE_CALLBACK.
PSTARBURN_SET_FILE_POINTER_CALLBACK (see page 588)	This is type PSTARBURN_SET_FILE_POINTER_CALLBACK.
PSTARBURN_STARWAVE_CALLBACK (see page 588)	StarWave callback passed to I/O functions. Return anything except zero from it to cancel I/O operation
PSTARBURN_UDF2_COLLISION_CALLBACK_EX (see page 589)	StarBurn UDF ansi name collision handling callback
PSTARBURN_UDF2_IMPORT_CALLBACK (see page 589)	StarBurn UDF2 import callback
PSTARBURN_UDF2_PROGRESS_CALLBACK (see page 590)	StarBurn UDF2 progress callback
PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX (see page 590)	StarBurn UDF Unicode name collision handling callback
PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK (see page 591)	StarBurn UDF2 UNICODE progress callback
PSTARBURN_WRITEFILE_CALLBACK (see page 591)	This is type PSTARBURN_WRITEFILE_CALLBACK.

2.3.1 CFuncStarWaveConversionCallback Type

C++

```
typedef void (__stdcall * CFuncStarWaveConversionCallback)(BYTE i_progress, BOOL* op_stop,
PVOID ip_user_data);
```

File

StarBurn.h (see page 662)

Description

type of callback function

2.3.2 PCALLBACK Type

C++

```
typedef VOID (__stdcall * PCALLBACK)(IN CALLBACK_NUMBER p__CALLBACK_NUMBER, IN PVOID
p__PVOID__CallbackContext, OUT PVOID p__PVOID__CallbackSpecial1, OUT PVOID
p__PVOID__CallbackSpecial2);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__CALLBACK_NUMBER	Callback number
p__PVOID__CallbackContext	Passed callback context
p__PVOID__CallbackSpecial1	Special parameter 1

p__PVOID__CallbackSpecial2	Special parameter 2

Description

Callback type (function of this type must be passed as callback)

Remarks

Callback number	Special parameter 1	Special parameter 2
CN_FILE_TREE_PROGRESS_ADD	(PCHAR) Adding item name	Not used
CN_FILE_TREE_PROGRESS_REMOVE	(PCHAR) Removing item name	Not used
CN_FILE_TREE_PROGRESS_IGNORE	(PCHAR) Ignoring item name	Not used
CN_FILE_TREE_PROGRESS_NAME_COLLISION	(PNAME_COLLISION_INFO (see page 511)) Collision info	Not used
CN_TARGET_FILE_ANALYZE_BEGIN	Not used	Not used
CN_TARGET_FILE_ANALYZE_END	(PLARGE_INTEGER) Target file size in UCHARs	(PLARGE_INTEGER) Target file size in LBs
CN_WAIT_CACHE_FULL_BEGIN	(PULONG) Wait time in milliseconds	Not used
CN_WAIT_CACHE_FULL_END	Not used	Not used
CN_SYNCHRONIZE_CACHE_BEGIN	Not used	Not used
CN_SYNCHRONIZE_CACHE_END	Not used	Not used
CN_FIND_DEVICE	(PSCSI_DEVICE_ADDRESS (see page 522)) SCSI device address	Not used
CN_CDVD_READ_PROGRESS	(PLARGE_INTEGER) Track size in LBs	(PLARGE_INTEGER) LBs readen
CN_CDVD_WRITE_PROGRESS	(PLARGE_INTEGER) File size in LBs	(PLARGE_INTEGER) LBs written
CN_CDVD_BUFFER_STATUS	(PULONG) Buffer size in UCHARs	(PULONG) Buffer free size in UCHARs
CN_CDVD_TRACK_BEGIN	(PULONG) Started track number	(PULONG) Total number of tracks
CN_CDVD_TRACK_END	(PULONG) Completed track number	(PULONG) Total number of track
CN_CDVD_SPLIT_BEGIN	(PLONG) Started split section number	(PLONG) Total number of tracks
CN_CDVD_SPLIT_END	(PLONG) Completed split section number	(PLONG) Total number of tracks
CN_CDVD_READ_BAD_BLOCK_HIT	(PLONG) LBA of bad blocks	(PULONG) Number of bad blocks
CN_CDVD_READ_ECCEDC_BAD_BLOCK_HIT	(PLONG) LBA of ECC/EDC bad blocks	(PULONG) Number of ECC/EDC bad blocks
CN_CDVD_READ_RETRY	(PLONG) LBA of retry reading blocks	(PULONG) Number of retry reading LBs
CN_DVDPLUSRW_FORMAT_BEGIN	Not used	Not used
CN_DVDPLUSRW_FORMAT_END	Not used	Not used
CN_DVDRAM_FORMAT_BEGIN	Not used	Not used
CN_DVDRAM_FORMAT_END	Not used	Not used
CN_BUFFER_UNDERRUN	Not used	Not used
CN_DVD_MEDIA_PADDING_SIZE	(PLARGE_INTEGER) Target file size in LBs	(PULONG) Track padding size in LBs
CN_DVD_MEDIA_PADDING_BEGIN	Not used	Not used

CN_DVD_MEDIA_PADDING_END	Not used	Not used
CN_CDVD_READ_CANCEL_QUERY	Undocumented	Undocumented
CN_DVDRW_QUICK_FORMAT_BEGIN	Not used	Not used
CN_DVDRW_QUICK_FORMAT_END	Not used	Not used
CN_DVD_TEST_WRITE_DISABLED	Not used	Not used
CN_CDVD_DPM_BEGIN	Undocumented	Undocumented
CN_CDVD_DPM_END	Undocumented	Undocumented
CN_CDVD_DPM_PROGRESS	Undocumented	Undocumented
CN_CDVD_VERIFY_PROGRESS	(PLONG) Processed LBs	(PLONG) Size in LBs
CN_SAO_TRACK_WRITE_BEGIN	(PLONG) Track index	Not used
CN_SAO_TRACK_WRITE_END	(PLONG) Track index	Not used
CN_DVD_MEDIA_PADDING_WRITE_PROGRESS	(PULONG) Track padding size in LBs	(PLONG) Padding left in LBs
CN_CDVD_WRITE_BEGIN	Not used	Not used
CN_CDVD_WRITE_END	Not used	Not used
CN_CDVD_VERIFY_BEGIN	Not used	Not used
CN_CDVD_VERIFY_END	Not used	Not used
CN_UDF_FILE_LOOKUP	(PUDF_FILE_LOOKUP_ENTRY (see page 543)) File information	Not used

2.3.3 PSTARBURN_CLOSEHANDLE_CALLBACK Type

C++

```
typedef BOOL (__stdcall * PSTARBURN_CLOSEHANDLE_CALLBACK)(HANDLE hObject, PDWORD pdwError);
```

File

StarBurn.h (see page 662)

Description

This is type PSTARBURN_CLOSEHANDLE_CALLBACK.

2.3.4 PSTARBURN_CREATEFILE_CALLBACK Type

C++

```
typedef HANDLE (__stdcall * PSTARBURN_CREATEFILE_CALLBACK)(LPCSTR lpFileName, DWORD
dwDesiredAccess, DWORD dwShareMode, DWORD dwCreationDisposition, DWORD
dwFlagsAndAttributes, PDWORD pdwError);
```

File

StarBurn.h (see page 662)

Description

This is type PSTARBURN_CREATEFILE_CALLBACK.

2.3.5 PSTARBURN_DELETEFILE_CALLBACK Type

C++

```
typedef BOOL (__stdcall * PSTARBURN_DELETEFILE_CALLBACK)(LPCSTR lpFileName, PDWORD
pdwError);
```

File

StarBurn.h (see page 662)

Description

This is type PSTARBURN_DELETEFILE_CALLBACK.

2.3.6 PSTARBURN_FILE_OBJECT Type

C++

```
typedef struct _STARBURN_FILE_OBJECT * PSTARBURN_FILE_OBJECT;
```

File

StarBurn.h (see page 662)

Description

Structure that represents StarBurn file object used internally with file I/O

2.3.7 PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK Type

C++

```
typedef BOOL (__stdcall * PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK)(HANDLE hFile, PDWORD
pdwError);
```

File

StarBurn.h (see page 662)

Description

This is type PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK.

2.3.8 PSTARBURN_ISO2_COLLISION_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_ISO2_COLLISION_CALLBACK)(char *p__PCHAR__NameBuffer,
unsigned long p__ULONG__BufferSize, unsigned char p__UCHAR__IsDirectory, long
p__LONG__CollisionError, const void *p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PCHAR__NameBuffer	Pointer to the file name (zero-terminated ANSI string)
p__ULONG__BufferSize	Size of Name buffer in UCHARs
p__UCHAR__IsDirectory	Non-zero if it is a directory, zero if it is a file
p__LONG__CollisionError	Size of Name buffer in UCHARs
p__PVOID__Context	Passed callback context

Description

StarBurn ISO name collision handling callback

Remarks

Check StarBurn_ISO2_DirectoryProcess (see page 252)(...) API call

2.3.9 PSTARBURN_ISO2_PROGRESS_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_ISO2_PROGRESS_CALLBACK)(const char *p__PCHAR__Name,
const unsigned short *p__PUSHORT__UnicodePathName, long p__LONG__Result, const void
*p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PCHAR__Name	Pointer to the file name added
p__PUSHORT__UnicodePathName	Pointer to the full path & name of the file added (Unicode)
p__LONG__Result	result code of file adding
p__PVOID__Context	Passed callback context

Description

StarBurn ISO progress callback

Remarks

Check StarBurn_ISO2_DirectoryProcess (see page 252)(...) API call

2.3.10 PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK)(unsigned short
*p__PUSHORT__NameBuffer, unsigned long p__ULONG__BufferSize, unsigned char
p__UCHAR__IsDirectory, long p__LONG__CollisionError, const void *p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PUSHORT__NameBuffer	Pointer to the file name (zero-terminated Unicode string)
p__ULONG__BufferSize	Size of Name buffer in UCHARs
p__UCHAR__IsDirectory	Non-zero if it is a directory, zero if it is a file
p__LONG__CollisionError	Size of Name buffer in UCHARs
p__PVOID__Context	Passed callback context

Description

StarBurn ISO Unicode name collision handling callback

Remarks

Check StarBurn_ISO2_DirectoryProcess (see page 252)(...) API call

2.3.11 PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK)(const unsigned short
*p__PUSHORT__UnicodeName, const unsigned short *p__PUSHORT__UnicodePathName, long
p__LONG__Result, const void *p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PUSHORT__UnicodeName	Pointer to the UNICODE file name added
p__PUSHORT__UnicodePathName	Pointer to the UNICODE full path & name of the file added
p__LONG__Result	result code of file adding
p__PVOID__Context	Passed callback context

Description

StarBurn ISO UNICODE progress callback

Remarks

Check StarBurn_ISO2_DirectoryUnicodeProcess (see page 254)(...) API call

2.3.12 PSTARBURN_ISO9660_READ_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_ISO9660_READ_CALLBACK)(LONG p__LONG__LBA, PVOID
p__PVOID_Buffer, ULONG p__ULONG__BufferSizeInUCHARs, PVOID p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__LONG__LBA	Logical block address to read content from

p__PVOID__Buffer	Pointer to buffer to store read content to
p__ULONG__BufferSizeInUCHARs	Buffer size in unsigned chars
p__PVOID__Context	Passed callback context

Description

StarBurn ISO2 read callback (for import)

Remarks

Check StarBurn_ISO2_ImpVolumeImport (see page 261)(...) API call

2.3.13 PSTARBURN_READFILE_CALLBACK Type

C++

```
typedef BOOL (__stdcall * PSTARBURN_READFILE_CALLBACK)(HANDLE hFile, LPVOID lpBuffer, DWORD
nNumberOfBytesToRead, LPDWORD lpNumberOfBytesRead, LPOVERLAPPED lpOverlapped, PDWORD
pdwError);
```

File

StarBurn.h (see page 662)

Description

This is type PSTARBURN_READFILE_CALLBACK.

2.3.14 PSTARBURN_SET_FILE_POINTER_CALLBACK Type

C++

```
typedef DWORD (__stdcall * PSTARBURN_SET_FILE_POINTER_CALLBACK)(HANDLE hFile, LONG
lDistanceToMove, PLONG lpDistanceToMoveHigh, DWORD dwMoveMethod, PDWORD pdwError);
```

File

StarBurn.h (see page 662)

Description

This is type PSTARBURN_SET_FILE_POINTER_CALLBACK.

2.3.15 PSTARBURN_STARWAVE_CALLBACK Type

C++

```
typedef ULONG (__stdcall * PSTARBURN_STARWAVE_CALLBACK)(IN ULONG p__ULONG__Reason, IN ULONG
p__ULONG__Parameter, IN PVOID p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__ULONG__Reason	Callback reason (see STARBURN_STARWAVE_CALLBACK_REASON (see page 558) enumeration)
p__ULONG__Parameter	Callback parameter

p__PVOID__Context	Passed callback context

Description

StarWave callback passed to I/O functions. Return anything except zero from it to cancel I/O operation

Remarks

Callback reason	Callback parameter
STARBURN_STARWAVE_CALLBACK_REASON_PROGRESS	Completion percent

2.3.16 PSTARBURN_UDF2_COLLISION_CALLBACK_EX Type

C++

```
typedef LONG (__stdcall * PSTARBURN_UDF2_COLLISION_CALLBACK_EX)(char *NameBuffer, unsigned long BufferSize, unsigned char IsDirectory, long CollisionError, const void *Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PUSHORT__NameBuffer	Pointer to the file name (zero-terminated ansi string)
p__ULONG__BufferSize	Size of Name buffer in UCHARs
p__UCHAR__IsDirectory	Non-zero if it is a directory, zero if it is a file
p__LONG__CollisionError	Size of Name buffer in UCHARs
p__PVOID__Context	Passed callback context

Description

StarBurn UDF ansi name collision handling callback

Remarks

Check StarBurn_UDF2_DirectoryContentsUnicodeProcessEx (see page 389)(...) API call

2.3.17 PSTARBURN_UDF2_IMPORT_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_UDF2_IMPORT_CALLBACK)(LONG p__LONG__LBA, PVOID p__PVOID_Buffer, ULONG p__ULONG__BufferSizeInUCHARs, PVOID p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__LONG__LBA	Logical block address to read content from
p__PVOID__Buffer	Pointer to buffer to store read content to
p__ULONG__BufferSizeInUCHARs	Buffer size in unsigned chars
p__PVOID__Context	Passed callback context

Description

StarBurn UDF2 import callback

Remarks

Check StarBurn_UDF2_ImpVolumeImport (see page 404)(...) API call

2.3.18 PSTARBURN_UDF2_PROGRESS_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_UDF2_PROGRESS_CALLBACK)(PCHAR p__PCHAR__Name, PCHAR
p__PCHAR__PathName, PVOID p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PCHAR__Name	Pointer to the file name added
p__PCHAR__PathName	Pointer to the full path & name of the file added
p__PVOID__Context	Passed callback context

Description

StarBurn UDF2 progress callback

Remarks

Check StarBurn_UDF2_DirectoryProcessEx (see page 391)(...) API call

2.3.19 PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX Type

C++

```
typedef LONG (__stdcall * PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX)(unsigned short*
p__PUSHORT__NameBuffer, unsigned long p__ULONG__BufferSize, unsigned char
p__UCHAR__IsDirectory, long p__LONG__CollisionError, const void* p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PUSHORT__NameBuffer	Pointer to the file name (zero-terminated Unicode string)
p__ULONG__BufferSize	Size of Name buffer in UCHARs
p__UCHAR__IsDirectory	Non-zero if it is a directory, zero if it is a file
p__LONG__CollisionError	Size of Name buffer in UCHARs
p__PVOID__Context	Passed callback context

Description

StarBurn UDF Unicode name collision handling callback

Remarks

Check StarBurn_UDF2_DirectoryContentsUnicodeProcessEx (see page 389)(...) API call

2.3.20 PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK Type

C++

```
typedef LONG (__stdcall * PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK)(PUSHORT
p__PUSHORT__UnicodeName, PUSHORT p__PUSHORT__UnicodePathName, PVOID p__PVOID__Context);
```

File

StarBurn.h (see page 662)

Parameters

Parameters	Description
p__PUSHORT__UnicodeName	Pointer to the UNICODE file name added
p__PUSHORT__UnicodePathName	Pointer to the UNICODE full path & name of the file added
p__PVOID__Context	Passed callback context

Description

StarBurn UDF2 UNICODE progress callback

Remarks

Check StarBurn_UDF2_DirectoryUnicodeProcessEx (see page 394)(...) API call

2.3.21 PSTARBURN_WRITEFILE_CALLBACK Type

C++

```
typedef BOOL (__stdcall * PSTARBURN_WRITEFILE_CALLBACK)(HANDLE hFile, LPCVOID lpBuffer,
DWORD nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten, LPOVERLAPPED lpOverlapped,
PDWORD pdwError);
```

File

StarBurn.h (see page 662)

Description

This is type PSTARBURN_WRITEFILE_CALLBACK.

2.4 Macros

The following table lists macros in this documentation.

Macros

Name	Description
__STARBURN_H__ (see page 596)	Define StarBurn.h (see page 662) to avoid including it more then once
ASPI_SAFE_BUFFER_SIZE_IN_UCHARS (see page 596)	ASPI safe buffer size in UCHARs. ASPI is assumed to support at least 64KB large transfer per time. 64KB - PAGE_SIZE_IN_UCHARS (see page 620) was old default value. Now we'll use DVD_ECC_BLOCK_SIZE_IN_UCHARS (see page 605) as it's smaller and produce better results when dealing with DVD media
AUDIO_LB_SIZE_IN_UCHARS (see page 597)	Audio LB (logical block) size in UCHARs

BLURAY_SPEED_IN_KBPS_1X (see page 597)	Actually 4495.5
BLURAY_SPEED_IN_KBPS_2X (see page 597)	2X Blu-Ray speed constant (BD-R and BD-RE)
BUFFER_STATUS_REPORT_DELAY_IN_SECONDS (see page 597)	Default buffer status report delay in seconds
CACHE_SIZE_IN_MBS (see page 598)	64
CD_SPEED_IN_KBPS_10X (see page 598)	10X CD speed constant
CD_SPEED_IN_KBPS_12X (see page 598)	12X CD speed constant
CD_SPEED_IN_KBPS_16X (see page 598)	16X CD speed constant
CD_SPEED_IN_KBPS_1X (see page 599)	1X CD speed constant
CD_SPEED_IN_KBPS_20X (see page 599)	20X CD speed constant
CD_SPEED_IN_KBPS_24X (see page 599)	24X CD speed constant
CD_SPEED_IN_KBPS_2P2X (see page 599)	2.2X CD speed constant
CD_SPEED_IN_KBPS_2X (see page 600)	2X CD speed constant
CD_SPEED_IN_KBPS_32X (see page 600)	32X CD speed constant
CD_SPEED_IN_KBPS_36X (see page 600)	36X CD speed constant
CD_SPEED_IN_KBPS_3X (see page 600)	3X CD speed constant
CD_SPEED_IN_KBPS_40X (see page 601)	40X CD speed constant
CD_SPEED_IN_KBPS_44X (see page 601)	44X CD speed constant
CD_SPEED_IN_KBPS_48X (see page 601)	48X CD speed constant
CD_SPEED_IN_KBPS_4X (see page 601)	4X CD speed constant
CD_SPEED_IN_KBPS_52X (see page 602)	52X CD speed constant
CD_SPEED_IN_KBPS_56X (see page 602)	56X CD speed
CD_SPEED_IN_KBPS_6X (see page 602)	6X CD speed constant
CD_SPEED_IN_KBPS_72X (see page 602)	72X CD speed
CD_SPEED_IN_KBPS_8X (see page 603)	8X CD speed constant
CDVD_SPEED_IS_KBPS_MAXIMUM (see page 603)	Top supported CD/DVD/Blu-Ray/HD-DVD speed constant (wildcard)
DEFAULT_NUMBER_OF_VERIFY_READ_RETRIES (see page 603)	Default number of read retries during verification process
DEVICES_PER_BUS (see page 603)	Top number of SCSI devices per logical SCSI bus (was 15 originally)
DISC_STATUS_COMPLETE (see page 604)	Disc is complete
DISC_STATUS_EMPTY (see page 604)	Disc is empty
DISC_STATUS_INCOMPLETE (see page 604)	Disc is incomplete
DUAL_LAYER_DVD_CAPACITY_SIZE_IN_LBS (see page 604)	Doubled...
DVD_ECC_BLOCK_SIZE_IN_LBS (see page 605)	DVD ECC block size in logical blocks (2048 UCHARs each)
DVD_ECC_BLOCK_SIZE_IN_UCHARS (see page 605)	DVD ECC block size in UCHARs. This is true "hardware" logical block size on DVD. It's recommended to have transfers of this size and this size aligned
DVD_PROTECTION_CPRM (see page 605)	DVD with CPRM (Copy-Protected Removable Media) protection
DVD_PROTECTION_CSS (see page 605)	DVD with CSS (Content Scrambling System) protection
DVD_PROTECTION_NONE (see page 606)	DVD w/o any protection system
DVD_SPEED_IN_KBPS_10X (see page 606)	10X DVD speed constant (DVD-R, DVD-R DL, DVD+R, DVD+R DL and DVD+RW)
DVD_SPEED_IN_KBPS_12X (see page 606)	12X DVD speed constant (DVD-R, DVD+R)
DVD_SPEED_IN_KBPS_16X (see page 606)	16X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_18X (see page 607)	18X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_1X (see page 607)	1X DVD speed constant (DVD-R, DVD-RW and DVD-RAM)
DVD_SPEED_IN_KBPS_20X (see page 607)	20X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_22X (see page 607)	22X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_24X (see page 608)	24X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_2DOT4X (see page 608)	2.4X DVD speed constant (DVD+R, DVD+R DL and DVD+RW)
DVD_SPEED_IN_KBPS_2X (see page 608)	2X DVD speed constant (DVD-R, DVD-RW and DVD-RAM)
DVD_SPEED_IN_KBPS_32X (see page 608)	32X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_3X (see page 609)	3X DVD speed constant (DVD-RAM)
DVD_SPEED_IN_KBPS_40X (see page 609)	40X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_48X (see page 609)	48X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_4X (see page 609)	4X DVD speed constant (DVD-R, DVD-R DL, DVD-RW, DVD+R, DVD+R DL, DVD+RW and DVD-RAM)
DVD_SPEED_IN_KBPS_50X (see page 610)	50X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_5X (see page 610)	5X DVD speed constant (DVD-RAM)
DVD_SPEED_IN_KBPS_6X (see page 610)	6X DVD speed constant (DVD-RW)
DVD_SPEED_IN_KBPS_8X (see page 610)	8X DVD speed constant (DVD-R, DVD-R DL, DVD+R, DVD+R DL and DVD+RW)
ELTORITO_BOOTABLE (see page 611)	Bootable disc ID

ELTORITO_BOOTSYSTEM_ID (see page 611)	Bootable specification signature
ELTORITO_DEF_LOAD_SEGMENT (see page 611)	Real mode code load segment
ELTORITO_NON_BOOTABLE (see page 611)	Non-bootable disc ID
ERROR_FIELD_C2_AND_BLOCK_ERROR (see page 612)	C2 and block error information
ERROR_FIELD_C2_ERROR (see page 612)	C2 error information
ERROR_FIELD_NO_ERROR (see page 612)	No error information
ERROR_FIELD_RESERVED (see page 612)	Reserved
EXPECTED_LB_TYPE_ANY (see page 613)	Accept all LBs
EXPECTED_LB_TYPE_CDDA (see page 613)	Accept only CDDA (CD digital audio) LBs
EXPECTED_LB_TYPE_MODE1 (see page 613)	Accept only MODE1 LBs
EXPECTED_LB_TYPE_MODE2 (see page 613)	Accept only MODE2 LBs (both Form1 and Form2)
EXPECTED_LB_TYPE_MODE2_FORM1 (see page 614)	Accept only MODE2 Form1 LBs
EXPECTED_LB_TYPE_MODE2_FORM2 (see page 614)	Accept only MODE2 Form2 LBs
HARD_DISK_LB_SIZE_IN_UCHARS (see page 614)	Hard disk LB (logical block) size in UCHARs
HEADER_CODE_ALL_HEADERS (see page 614)	Both header and subheader
HEADER_CODE_HEADER_ONLY (see page 615)	4-byte header
HEADER_CODE_NONE (see page 615)	No header
HEADER_CODE_SUBHEADER_ONLY (see page 615)	MODE2 Form1 or Form2 subheader
ISO9660_DATE_SIZE_IN_UCHARS (see page 615)	ISO9660 date size in UCHARs
ISO9660_FILE_SIZE_IN_UCHARS (see page 616)	ISO9660 file size in UCHARs
ISO9660_NAME_SIZE_IN_UCHARS (see page 616)	ISO9660 name size in UCHARs
ISO9660_SYSTEM_VOLUME_ID_SIZE_IN_UCHARS (see page 616)	ISO9660 volume descriptor system and volume ID size in UCHARs
ISO9660_TREE_LEVEL (see page 616)	ISO9660 top supported file tree level
LAST_SESSION_COMPLETE (see page 617)	Last session is complete
LAST_SESSION_EMPTY (see page 617)	Last session is empty
LAST_SESSION_INCOMPLETE (see page 617)	Last session is incomplete
MODE1_LB_SIZE_IN_UCHARS (see page 617)	MODE1 LB (logical block) size in UCHARs
MODE2_FORM1_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Form1 LB (logical block) size in UCHARs
MODE2_FORM2_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Form2 LB (logical block) size in UCHARs
MODE2_FORM2_SUB_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Form2 + 8 UCHARs of subheader LB (logical block) size in UCHARs
MODE2_FORMLESS_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Formless LB (logical block) size in UCHARS
NUMBER_OF_RAW_TRACKS (see page 619)	Top number of raw tracks (including maintenance tracks) on CD/DVD/Blu-Ray/HD-DVD disc. This is calculated based on the worst CD case when all tracks will be in a separate sessions. We need this approximate value to avoid allocating TOC_INFORMATION (see page 573) structure dynamically.
NUMBER_OF_READ_RETRIES (see page 619)	Default number of retries on read operations (if bad block was possibly hit, was 2 originally)
NUMBER_OF_SUBCHANNELS (see page 619)	sub-channels from P to W
NUMBER_OF_TRACKS (see page 619)	Top number of tracks (or border-in/border-out zones) on CD/DVD/Blu-Ray/HD-DVD disc
PAGE_SIZE_IN_UCHARS (see page 620)	Small page size in UCHARs on x86 machines (default allocatable storage)
RAW_LB_PQ_SUB_SIZE_IN_UCHARS (see page 620)	RAW LB (logical block) with PQ sub-channel size in UCHARs
RAW_LB_RAW_PW_SUB_SIZE_IN_UCHARS (see page 620)	RAW LB (logical block) with RAW P-W sub-channel size in UCHARs
RAW_LB_SIZE_IN_UCHARS (see page 620)	RAW LB (logical block) size in UCHARs
READ_REPORT_DELAY_IN_SECONDS (see page 621)	Default read report delay in seconds
SCSI_DEVICE_DIRECT_ACCESS (see page 621)	SCSI disk device
SCSI_DEVICE_MAGNETO_OPTICAL (see page 621)	SCSI MO (Magneto-Optical) disk device
SCSI_DEVICE_MEDIUM_CHANGER (see page 621)	SCSI jukebox (disc changer) device
SCSI_DEVICE_NETWORK (see page 622)	SCSI network device
SCSI_DEVICE_PRINTER (see page 622)	SCSI printer device
SCSI_DEVICE_PROCESSOR (see page 622)	SCSI processor device
SCSI_DEVICE_RO_DIRECT_ACCESS (see page 622)	SCSI CD/DVD/Blu-Ray/HD-DVD device
SCSI_DEVICE_SCANNER (see page 623)	SCSI scanner device
SCSI_DEVICE_SEQUENTIAL_ACCESS (see page 623)	SCSI tape device
SCSI_DEVICE_UNPRESENT (see page 623)	SCSI device... Is gone!
SCSI_DEVICE_WILDCARD (see page 623)	SCSI device any (wildcard for StarBurn_FindDevice (see page 231)() function)
SCSI_DEVICE_WORM (see page 624)	SCSI WORM (Write-Once-Read-Multiple) device
SHORT_RAW_LB_SIZE_IN_UCHARS (see page 624)	Short (w/o sync and header) RAW LB (logical block) size in UCHARs
SINGLE_LAYER_DVD_CAPACITY_SIZE_IN_LBS (see page 624)	Number of logical blocks on single layer DVD

SMALLEST_CACHE_SIZE_IN_MBS (see page 624)	Smallest cache size in MBs that toolkit will allow to use to buffer "write" operations
SMALLEST_CACHE_SIZE_IN_UCHARS (see page 625)	Smallest cache size in UCHARs that toolkit will allow to use to buffer "write" operations
STARBURN_BDRE_FORMAT_CERTIFICATION_FULL (see page 625)	This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_FULL.
STARBURN_BDRE_FORMAT_CERTIFICATION_NO (see page 625)	BD-RE format certification types
STARBURN_BDRE_FORMAT_CERTIFICATION_QUICK (see page 625)	This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_QUICK.
STARBURN_BDRE_FORMAT_CERTIFICATION_RESERVED (see page 626)	This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_RESERVED.
STARBURN_BDRE_FORMAT_DEFAULT (see page 626)	BD-RE format types
STARBURN_BDRE_FORMAT_SPARE_AREA_EXPANSION (see page 626)	This is macro STARBURN_BDRE_FORMAT_SPARE_AREA_EXPANSION.
STARBURN_BDRE_FORMAT_WITH_SPARE_AREA (see page 627)	This is macro STARBURN_BDRE_FORMAT_WITH_SPARE_AREA.
STARBURN_BDRE_FORMAT_WITHOUT_SPARE_AREA (see page 627)	This is macro STARBURN_BDRE_FORMAT_WITHOUT_SPARE_AREA.
STARBURN_CACHE_SIZE_READ_ONLY (see page 627)	Default cache size for I/O operations
STARBURN_CACHE_SIZE_READ_WRITE (see page 627)	This is macro STARBURN_CACHE_SIZE_READ_WRITE.
STARBURN_CDFS2_FILE_ATTRIBUTE_DELETED (see page 628)	This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_DELETED.
STARBURN_CDFS2_FILE_ATTRIBUTE_DIRECTORY (see page 628)	File attributes (flags for Attributes member of STARBURN_UDF2_FILE_INFO (see page 567) & STARBURN_ISO2_FILE_INFO)
STARBURN_CDFS2_FILE_ATTRIBUTE_HIDDEN (see page 628)	This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_HIDDEN.
STARBURN_CDFS2_FILE_ATTRIBUTE_IMPORTED (see page 629)	This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_IMPORTED.
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileT (see page 629)	This is macro StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileT.
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileT (see page 629)	This is macro StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileT.
StarBurn_CdvdBurnerGrabber_GetDeviceInformationT (see page 630)	This is macro StarBurn_CdvdBurnerGrabber_GetDeviceInformationT.
StarBurn_CdvdBurnerGrabber_GrabTrackT (see page 630)	This is macro StarBurn_CdvdBurnerGrabber_GrabTrackT.
StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWT (see page 630)	This is macro StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWT.
StarBurn_CdvdBurnerGrabber_SessionAtOnceT (see page 631)	This is macro StarBurn_CdvdBurnerGrabber_SessionAtOnceT.
StarBurn_CdvdBurnerGrabber_SuperVideoCDExExT (see page 631)	This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDExExT.
StarBurn_CdvdBurnerGrabber_SuperVideoCDExT (see page 631)	This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDExT.
StarBurn_CdvdBurnerGrabber_SuperVideoCDT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDT.
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileT.
StarBurn_CdvdBurnerGrabber_VerifyFileExT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_VerifyFileExT.
StarBurn_CdvdBurnerGrabber_VerifyFileT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_VerifyFileT.
StarBurn_CdvdBurnerGrabber_VideoCDExExT (see page 633)	This is macro StarBurn_CdvdBurnerGrabber_VideoCDExExT.
StarBurn_CdvdBurnerGrabber_VideoCDExT (see page 633)	This is macro StarBurn_CdvdBurnerGrabber_VideoCDExT.
StarBurn_CdvdBurnerGrabber_VideoCDT (see page 633)	This is macro StarBurn_CdvdBurnerGrabber_VideoCDT.
STARBURN_DEBUG_FACILITY_ALLMESSAGES (see page 634)	Debug facility: Permit All Messages
STARBURN_DEBUG_FACILITY_ALLTARGETS (see page 634)	Debug facility: All targets
STARBURN_DEBUG_FACILITY_CUSTOM (see page 634)	Debug facility: Custom
STARBURN_DEBUG_FACILITY_DEBUG (see page 634)	Debug facility: Permit Debug Messages
STARBURN_DEBUG_FACILITY_DEBUG_OUTPUT (see page 635)	Debug facility: Debug output
STARBURN_DEBUG_FACILITY_ERROR (see page 635)	Debug facility: Permit Error Messages
STARBURN_DEBUG_FACILITY_LOG_FILE (see page 635)	Debug facility: Log file
STARBURN_DEBUG_FACILITY_NONE (see page 635)	Debug facility: None
STARBURN_DEBUG_FACILITY_SYSTEM_CONSOLE (see page 636)	Debug facility: System console
STARBURN_DEBUG_FACILITY_TRACE (see page 636)	Debug facility: Permit Trace Messages
STARBURN_DEBUG_FACILITY_WARNING (see page 636)	Debug facility: Permit Warning Messages
STARBURN_DISC_TYPE_CDDA (see page 636)	Disc type is CDDA (CD-ROM or Digital Audio)
STARBURN_DISC_TYPE_CDI (see page 637)	Disc type is CDI
STARBURN_DISC_TYPE_UNDEFINED (see page 637)	Disc type is undefined
STARBURN_DISC_TYPE_XA (see page 637)	Disc type is CD-ROM XA
STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES (see page 637)	DVD-Video maximum number of titles
StarBurn_DVDVideo_CreateT (see page 638)	This is macro StarBurn_DVDVideo_CreateT.
StarBurn_ElTorito_BootCatalogAddSectionT (see page 638)	This is macro StarBurn_ElTorito_BootCatalogAddSectionT.
StarBurn_ElTorito_CreateBootCatalogT (see page 638)	This is macro StarBurn_ElTorito_CreateBootCatalogT.
StarBurn_GetAudioFileStreamSizeInUCHARs_FastT (see page 638)	This is macro StarBurn_GetAudioFileStreamSizeInUCHARs_FastT.
StarBurn_GetAudioFileStreamSizeInUCHARsT (see page 639)	This is macro StarBurn_GetAudioFileStreamSizeInUCHARsT.
StarBurn_GetDeviceLetterT (see page 639)	TCHAR functions
STARBURN_IMPEX_API (see page 639)	declspec(dllexport)

StarBurn_IsAudioFileSupportedT (see page 639)	This is macro StarBurn_IsAudioFileSupportedT.
StarBurn_ISO2_VolumeCreateBootElToritoT (see page 640)	This is macro StarBurn_ISO2_VolumeCreateBootElToritoT.
STARBURN_ISO9660_FILE_FLAGS_ASSOCIATED_FILE (see page 640)	This is macro STARBURN_ISO9660_FILE_FLAGS_ASSOCIATED_FILE.
STARBURN_ISO9660_FILE_FLAGS_DIRECTORY (see page 640)	This is macro STARBURN_ISO9660_FILE_FLAGS_DIRECTORY.
STARBURN_ISO9660_FILE_FLAGS_EXISTENCE (see page 641)	This is macro STARBURN_ISO9660_FILE_FLAGS_EXISTENCE.
STARBURN_ISO9660_FILE_FLAGS_NONE (see page 641)	ISO9660 file attributes flags
STARBURN_ISO9660_FILE_FLAGS_RECORD (see page 641)	This is macro STARBURN_ISO9660_FILE_FLAGS_RECORD.
STARBURN_ISO9660_FILE_MAX_SIZE_IN_UCHARS (see page 641)	(4GB - 1)
STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS (see page 642)	This is macro STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS.
STARBURN_ISO9660_JOLIET_RELAXED_NAME_SIZE_IN_WCHARS (see page 642)	
STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS (see page 642)	ISO9660 constants (begin)
STARBURN_ISO9660_TYPE_1999 (see page 643)	This is macro STARBURN_ISO9660_TYPE_1999.
STARBURN_ISO9660_TYPE_JOLIET (see page 643)	This is macro STARBURN_ISO9660_TYPE_JOLIET.
STARBURN_ISO9660_TYPE_JOLIET_RELAXED (see page 643)	This is macro STARBURN_ISO9660_TYPE_JOLIET_RELAXED.
STARBURN_ISO9660_TYPE_LEVEL1 (see page 643)	ISO9660 file system type
STARBURN_ISO9660_TYPE_LEVEL2 (see page 644)	This is macro STARBURN_ISO9660_TYPE_LEVEL2.
STARBURN_LOADING_MECHANISM_TYPE_CADDY (see page 644)	Loading mechanism type constants
STARBURN_LOADING_MECHANISM_TYPE_CARTRIDGE (see page 644)	This is macro STARBURN_LOADING_MECHANISM_TYPE_CARTRIDGE.
STARBURN_LOADING_MECHANISM_TYPE_CHANGER (see page 644)	This is macro STARBURN_LOADING_MECHANISM_TYPE_CHANGER.
STARBURN_LOADING_MECHANISM_TYPE_POP_UP (see page 645)	This is macro STARBURN_LOADING_MECHANISM_TYPE_POP_UP.
STARBURN_LOADING_MECHANISM_TYPE_RESERVED1 (see page 645)	This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED1.
STARBURN_LOADING_MECHANISM_TYPE_RESERVED2 (see page 645)	This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED2.
STARBURN_LOADING_MECHANISM_TYPE_RESERVED3 (see page 646)	This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED3.
STARBURN_LOADING_MECHANISM_TYPE_TRAY (see page 646)	This is macro STARBURN_LOADING_MECHANISM_TYPE_TRAY.
STARBURN_PIPE_SIZE_IN_UCHARS (see page 646)	60, 64 or 32 KB I/O packets
STARBURN_SMALLEST_SPLIT_FILE_SIZE_IN_MBS (see page 646)	StarBurn smallest split file size in megabytes
StarBurn_StarWave_CompressedFileReaderObjectCreateT (see page 647)	This is macro StarBurn_StarWave_CompressedFileReaderObjectCreateT.
StarBurn_StarWave_CompressedFileRecompressT (see page 647)	This is macro StarBurn_StarWave_CompressedFileRecompressT.
StarBurn_StarWave_CompressedFileUncompressT (see page 647)	This is macro StarBurn_StarWave_CompressedFileUncompressT.
StarBurn_StarWave_CompressedFileWriterObjectCreateT (see page 648)	This is macro StarBurn_StarWave_CompressedFileWriterObjectCreateT.
STARBURN_STARWAVE_IO_BUFFER_SIZE_IN_UCHARS (see page 648)	StarWave recommended I/O buffer size in UCHARs. It's *STRONGLY* recommended to use either it or multiplications of STARWAVE_IO_BUFFER_SIZE_IN_UCHARS as internal calculations and indexations are integer rather then float based
StarBurn_StarWave_UncompressedFileCompressT (see page 648)	This is macro StarBurn_StarWave_UncompressedFileCompressT.
StarBurn_StarWave_UncompressedFileSupportedIsT (see page 649)	This is macro StarBurn_StarWave_UncompressedFileSupportedIsT.
StarBurn_StarWave2_ConvertExT (see page 649)	This is macro StarBurn_StarWave2_ConvertExT.
StarBurn_StarWave2_EncodeMP3OGGFromWAVT (see page 649)	This is macro StarBurn_StarWave2_EncodeMP3OGGFromWAVT.
StarBurn_StarWave2_GetFileReaderAndFactoryT (see page 649)	This is macro StarBurn_StarWave2_GetFileReaderAndFactoryT.
StarBurn_UDF_AddT (see page 650)	This is macro StarBurn_UDF_AddT.
StarBurn_UDF_CreateExT (see page 650)	This is macro StarBurn_UDF_CreateExT.
StarBurn_UDF_CreateT (see page 650)	This is macro StarBurn_UDF_CreateT.
StarBurn_UDF_FormatTreeItemAsDirectoryT (see page 650)	This is macro StarBurn_UDF_FormatTreeItemAsDirectoryT.
StarBurn_UDF_FormatTreeItemAsFileT (see page 651)	This is macro StarBurn_UDF_FormatTreeItemAsFileT.
STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS (see page 651)	Name constants
STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS (see page 651)	This is macro STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS.
StarBurn_UDF2_VolumeCreateBootElToritoT (see page 652)	This is macro StarBurn_UDF2_VolumeCreateBootElToritoT.
StarBurn_UDFBridge_CreateExT (see page 652)	This is macro StarBurn_UDFBridge_CreateExT.
STARPORT_DEVICE_NAME_SIZE_IN_UCHARS (see page 652)	StarPort device name size in UCHARs
STARWAVEFACTORY_ID_DSHOW (see page 652)	This is macro STARWAVEFACTORY_ID_DSHOW.
STARWAVEFACTORY_ID_MP3 (see page 653)	This is macro STARWAVEFACTORY_ID_MP3.
STARWAVEFACTORY_ID_OGG (see page 653)	This is macro STARWAVEFACTORY_ID_OGG.
STARWAVEFACTORY_ID_WAV (see page 653)	StarWave2 factory ids
STARWAVEFACTORY_ID_WMA (see page 653)	This is macro STARWAVEFACTORY_ID_WMA.
SUBCHANNEL_NO (see page 654)	No sub-channel
SUBCHANNEL_PQ (see page 654)	PQ sub-channel
SUBCHANNEL_PQ_SIZE_IN_UCHARS (see page 654)	PQ sub-channel size in UCHARs

SUBCHANNEL_RAW_PW (see page 654)	RAW P-W sub-channel
SUBCHANNEL_RAW_PW_SIZE_IN_UCHARS (see page 655)	Raw P-W sub-channel size in UCHARs
SUBCHANNEL_RESERVED (see page 655)	Reserved
SUBCHANNEL_RW (see page 655)	R-W sub-channel
SUBCHANNEL_SIZE_IN_UCHARS (see page 655)	Sub-channel size in UCHARs
TRACK_NUMBER_INVISIBLE (see page 656)	Special "invisible" track number
TYPE_CODE_LAST_CHANCE (see page 656)	Logical Unit region is set. Additional restrictions required to make a change
TYPE_CODE_NONE (see page 656)	No Logical Unit region setting
TYPE_CODE_PERM (see page 656)	Logical Unit region has been set permanently, but may be reset by vendor if necessery
TYPE_CODE_SET (see page 657)	Logical Unit region is set
UDF_FILE_SET_DESCRIPTOR_LBA (see page 657)	UDF file set descriptor logical block address
UDF_FILE_SET_DESCRIPTOR_TERMINATOR_LBA (see page 657)	UDF file set descriptor terminator logical block address
UDF_FIRST_ANCHOR_LBA (see page 657)	First anchor volume descriptor pointer LBA
UDF_HEAD_SIZE_IN_LOGICAL_BLOCKS (see page 658)	UDF head size in logical blocks
UDF_ISO9660_FILE_SYSTEM_LBA (see page 658)	UDF/ISO9660 bridge file system logical block address
UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS (see page 658)	define UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS 6
UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS (see page 658)	UDF logical block size in UCHARs
UDF_NAME_SIZE_IN_UCHARS (see page 659)	UDF longest name size in UCHARs
UDF_NAME_SIZE_IN_UCHARS_ACTUAL (see page 659)	This is macro UDF_NAME_SIZE_IN_UCHARS_ACTUAL.
UDF_NSR_DESCRIPTOR_LBA (see page 659)	UDF NSR descriptor pointer LBA
UDF_TAIL_SIZE_IN_LOGICAL_BLOCKS (see page 659)	UDF tail size in logical blocks
WAVE_FILE_ALIGNMENT (see page 660)	WAVE file data alignment (4 UCHARs, 1 ULONG)
WAVE_FILE_BITS_PER_SAMPLE (see page 660)	WAVE file bits per sample
WAVE_FILE_CHANNELS (see page 660)	WAVE file number of channels
WAVE_FILE_DATA_SIGNATURE (see page 660)	WAVE file DATA signature
WAVE_FILE_RIFF_SIGNATURE (see page 661)	WAVE file RIFF signature
WAVE_FILE_SAMPLES_PER_SECOND (see page 661)	WAVE file samples per second
WAVE_FILE_TAG_SIGNATURE (see page 661)	WAVE file TAG signature
WAVE_FILE_WAVE_FORMAT (see page 661)	WAVE file WAVE format
WAVE_FILE_WAVE_SIGNATURE (see page 662)	WAVE file WAVE signature
WRITE_REPORT_DELAY_IN_SECONDS (see page 662)	Default write report delay in seconds

2.4.1 __STARBURN_H__ Macro

C++

```
#define __STARBURN_H__
```

File

StarBurn.h (see page 662)

Description

Define StarBurn.h (see page 662) to avoid including it more then once

2.4.2 ASPI_SAFE_BUFFER_SIZE_IN_UCHARS Macro

C++

```
#define ASPI_SAFE_BUFFER_SIZE_IN_UCHARS 65536
```

File

StarBurn.h (see page 662)

Description

ASPI safe buffer size in UCHARs. ASPI is assumed to support at least 64KB large transfer per time. 64KB - PAGE_SIZE_IN_UCHARS (see page 620) was old default value. Now we'll use DVD_ECC_BLOCK_SIZE_IN_UCHARS (see page 605) as it's smaller and produce better results when dealing with DVD media

2.4.3 AUDIO_LB_SIZE_IN_UCHARS Macro

C++

```
#define AUDIO_LB_SIZE_IN_UCHARS 2352
```

File

StarBurn.h (see page 662)

Description

Audio LB (logical block) size in UCHARs

2.4.4 BLURAY_SPEED_IN_KBPS_1X Macro

C++

```
#define BLURAY_SPEED_IN_KBPS_1X 4495 // Actually 4495.5
```

File

StarBurn.h (see page 662)

Description

Actually 4495.5

2.4.5 BLURAY_SPEED_IN_KBPS_2X Macro

C++

```
#define BLURAY_SPEED_IN_KBPS_2X 8991
```

File

StarBurn.h (see page 662)

Description

2X Blu-Ray speed constant (BD-R and BD-RE)

2.4.6 BUFFER_STATUS_REPORT_DELAY_IN_SECONDS Macro

C++

```
#define BUFFER_STATUS_REPORT_DELAY_IN_SECONDS 5
```

File

StarBurn.h (see page 662)

Description

Default buffer status report delay in seconds

2.4.7 CACHE_SIZE_IN_MBS Macro

C++

```
#define CACHE_SIZE_IN_MBS 160 // 64
```

File

StarBurn.h (see page 662)

Description

64

2.4.8 CD_SPEED_IN_KBPS_10X Macro

C++

```
#define CD_SPEED_IN_KBPS_10X 1760
```

File

StarBurn.h (see page 662)

Description

10X CD speed constant

2.4.9 CD_SPEED_IN_KBPS_12X Macro

C++

```
#define CD_SPEED_IN_KBPS_12X 2112
```

File

StarBurn.h (see page 662)

Description

12X CD speed constant

2.4.10 CD_SPEED_IN_KBPS_16X Macro

C++

```
#define CD_SPEED_IN_KBPS_16X 2800
```

File

StarBurn.h (see page 662)

Description

16X CD speed constant

2.4.11 CD_SPEED_IN_KBPS_1X Macro

C++

```
#define CD_SPEED_IN_KBPS_1X 176
```

File

StarBurn.h (see page 662)

Description

1X CD speed constant

2.4.12 CD_SPEED_IN_KBPS_20X Macro

C++

```
#define CD_SPEED_IN_KBPS_20X 3250
```

File

StarBurn.h (see page 662)

Description

20X CD speed constant

2.4.13 CD_SPEED_IN_KBPS_24X Macro

C++

```
#define CD_SPEED_IN_KBPS_24X 4224
```

File

StarBurn.h (see page 662)

Description

24X CD speed constant

2.4.14 CD_SPEED_IN_KBPS_2P2X Macro

C++

```
#define CD_SPEED_IN_KBPS_2P2X 387
```

File

StarBurn.h (see page 662)

Description

2.2X CD speed constant

2.4.15 CD_SPEED_IN_KBPS_2X Macro

C++

```
#define CD_SPEED_IN_KBPS_2X 353
```

File

StarBurn.h (see page 662)

Description

2X CD speed constant

2.4.16 CD_SPEED_IN_KBPS_32X Macro

C++

```
#define CD_SPEED_IN_KBPS_32X 5632
```

File

StarBurn.h (see page 662)

Description

32X CD speed constant

2.4.17 CD_SPEED_IN_KBPS_36X Macro

C++

```
#define CD_SPEED_IN_KBPS_36X 6336
```

File

StarBurn.h (see page 662)

Description

36X CD speed constant

2.4.18 CD_SPEED_IN_KBPS_3X Macro

C++

```
#define CD_SPEED_IN_KBPS_3X 528
```

File

StarBurn.h (see page 662)

Description

3X CD speed constant

2.4.19 CD_SPEED_IN_KBPS_40X Macro

C++

```
#define CD_SPEED_IN_KBPS_40X 7040
```

File

StarBurn.h (see page 662)

Description

40X CD speed constant

2.4.20 CD_SPEED_IN_KBPS_44X Macro

C++

```
#define CD_SPEED_IN_KBPS_44X 7744
```

File

StarBurn.h (see page 662)

Description

44X CD speed constant

2.4.21 CD_SPEED_IN_KBPS_48X Macro

C++

```
#define CD_SPEED_IN_KBPS_48X 8448
```

File

StarBurn.h (see page 662)

Description

48X CD speed constant

2.4.22 CD_SPEED_IN_KBPS_4X Macro

C++

```
#define CD_SPEED_IN_KBPS_4X 706
```

File

StarBurn.h (see page 662)

Description

4X CD speed constant

2.4.23 CD_SPEED_IN_KBPS_52X Macro

C++

```
#define CD_SPEED_IN_KBPS_52X 9152
```

File

StarBurn.h (see page 662)

Description

52X CD speed constant

2.4.24 CD_SPEED_IN_KBPS_56X Macro

C++

```
#define CD_SPEED_IN_KBPS_56X 9600
```

File

StarBurn.h (see page 662)

Description

56X CD speed

2.4.25 CD_SPEED_IN_KBPS_6X Macro

C++

```
#define CD_SPEED_IN_KBPS_6X 1056
```

File

StarBurn.h (see page 662)

Description

6X CD speed constant

2.4.26 CD_SPEED_IN_KBPS_72X Macro

C++

```
#define CD_SPEED_IN_KBPS_72X 11300
```

File

StarBurn.h (see page 662)

Description

72X CD speed

2.4.27 CD_SPEED_IN_KBPS_8X Macro

C++

```
#define CD_SPEED_IN_KBPS_8X 1400
```

File

StarBurn.h (see page 662)

Description

8X CD speed constant

2.4.28 CDVD_SPEED_IS_KBPS_MAXIMUM Macro

C++

```
#define CDVD_SPEED_IS_KBPS_MAXIMUM 0xFFFF
```

File

StarBurn.h (see page 662)

Description

Top supported CD/DVD/Blu-Ray/HD-DVD speed constant (wildcard)

2.4.29 DEFAULT_NUMBER_OF_VERIFY_READ_RETRIES Macro

C++

```
#define DEFAULT_NUMBER_OF_VERIFY_READ_RETRIES 10
```

File

StarBurn.h (see page 662)

Description

Default number of read retries during verification process

2.4.30 DEVICES_PER_BUS Macro

C++

```
#define DEVICES_PER_BUS 63
```

File

StarBurn.h (see page 662)

Description

Top number of SCSI devices per logical SCSI bus (was 15 originally)

2.4.31 DISC_STATUS_COMPLETE Macro

C++

```
#define DISC_STATUS_COMPLETE 0x02
```

File

StarBurn.h (see page 662)

Description

Disc is complete

2.4.32 DISC_STATUS_EMPTY Macro

C++

```
#define DISC_STATUS_EMPTY 0x00
```

File

StarBurn.h (see page 662)

Description

Disc is empty

2.4.33 DISC_STATUS_INCOMPLETE Macro

C++

```
#define DISC_STATUS_INCOMPLETE 0x01
```

File

StarBurn.h (see page 662)

Description

Disc is incomplete

2.4.34 DUAL_LAYER_DVD_CAPACITY_SIZE_IN_LBS Macro

C++

```
#define DUAL_LAYER_DVD_CAPACITY_SIZE_IN_LBS ( SINGLE_LAYER_DVD_CAPACITY_SIZE_IN_LBS * 2 )
// Doubled...
```

File

StarBurn.h (see page 662)

Description

Doubled...

2.4.35 DVD_ECC_BLOCK_SIZE_IN_LBS Macro

C++

```
#define DVD_ECC_BLOCK_SIZE_IN_LBS 16
```

File

StarBurn.h (see page 662)

Description

DVD ECC block size in logical blocks (2048 UCHARs each)

2.4.36 DVD_ECC_BLOCK_SIZE_IN_UCHARS Macro

C++

```
#define DVD_ECC_BLOCK_SIZE_IN_UCHARS 32768
```

File

StarBurn.h (see page 662)

Description

DVD ECC block size in UCHARs. This is true "hardware" logical block size on DVD. It's recommended to have transfers of this size and this size aligned

2.4.37 DVD_PROTECTION_CPRM Macro

C++

```
#define DVD_PROTECTION_CPRM 0x02
```

File

StarBurn.h (see page 662)

Description

DVD with CPRM (Copy-Protected Removable Media) protection

2.4.38 DVD_PROTECTION_CSS Macro

C++

```
#define DVD_PROTECTION_CSS 0x01
```

File

StarBurn.h (see page 662)

Description

DVD with CSS (Content Scrambling System) protection

2.4.39 DVD_PROTECTION_NONE Macro

C++

```
#define DVD_PROTECTION_NONE 0x00
```

File

StarBurn.h (see page 662)

Description

DVD w/o any protection system

2.4.40 DVD_SPEED_IN_KBPS_10X Macro

C++

```
#define DVD_SPEED_IN_KBPS_10X 13850
```

File

StarBurn.h (see page 662)

Description

10X DVD speed constant (DVD-R, DVD-R DL, DVD+R, DVD+R DL and DVD+RW)

2.4.41 DVD_SPEED_IN_KBPS_12X Macro

C++

```
#define DVD_SPEED_IN_KBPS_12X 16620
```

File

StarBurn.h (see page 662)

Description

12X DVD speed constant (DVD-R, DVD+R)

2.4.42 DVD_SPEED_IN_KBPS_16X Macro

C++

```
#define DVD_SPEED_IN_KBPS_16X 22160
```

File

StarBurn.h (see page 662)

Description

16X DVD speed constant (DVD+R and DVD-R)

2.4.43 DVD_SPEED_IN_KBPS_18X Macro

C++

```
#define DVD_SPEED_IN_KBPS_18X 24930
```

File

StarBurn.h (see page 662)

Description

18X DVD speed constant (DVD+R and DVD-R)

2.4.44 DVD_SPEED_IN_KBPS_1X Macro

C++

```
#define DVD_SPEED_IN_KBPS_1X 1385
```

File

StarBurn.h (see page 662)

Description

1X DVD speed constant (DVD-R, DVD-RW and DVD-RAM)

2.4.45 DVD_SPEED_IN_KBPS_20X Macro

C++

```
#define DVD_SPEED_IN_KBPS_20X 27700
```

File

StarBurn.h (see page 662)

Description

20X DVD speed constant (DVD+R and DVD-R)

2.4.46 DVD_SPEED_IN_KBPS_22X Macro

C++

```
#define DVD_SPEED_IN_KBPS_22X 30470
```

File

StarBurn.h (see page 662)

Description

22X DVD speed constant (DVD+R and DVD-R)

2.4.47 DVD_SPEED_IN_KBPS_24X Macro

C++

```
#define DVD_SPEED_IN_KBPS_24X 33240
```

File

StarBurn.h (see page 662)

Description

24X DVD speed constant (DVD+R and DVD-R)

2.4.48 DVD_SPEED_IN_KBPS_2DOT4X Macro

C++

```
#define DVD_SPEED_IN_KBPS_2DOT4X 3324
```

File

StarBurn.h (see page 662)

Description

2.4X DVD speed constant (DVD+R, DVD+R DL and DVD+RW)

2.4.49 DVD_SPEED_IN_KBPS_2X Macro

C++

```
#define DVD_SPEED_IN_KBPS_2X 2770
```

File

StarBurn.h (see page 662)

Description

2X DVD speed constant (DVD-R, DVD-RW and DVD-RAM)

2.4.50 DVD_SPEED_IN_KBPS_32X Macro

C++

```
#define DVD_SPEED_IN_KBPS_32X DVD_SPEED_IN_KBPS_1X * 32
```

File

StarBurn.h (see page 662)

Description

32X DVD speed constant (DVD+R and DVD-R)

2.4.51 DVD_SPEED_IN_KBPS_3X Macro

C++

```
#define DVD_SPEED_IN_KBPS_3X 4155
```

File

StarBurn.h (see page 662)

Description

3X DVD speed constant (DVD-RAM)

2.4.52 DVD_SPEED_IN_KBPS_40X Macro

C++

```
#define DVD_SPEED_IN_KBPS_40X DVD_SPEED_IN_KBPS_1X * 40
```

File

StarBurn.h (see page 662)

Description

40X DVD speed constant (DVD+R and DVD-R)

2.4.53 DVD_SPEED_IN_KBPS_48X Macro

C++

```
#define DVD_SPEED_IN_KBPS_48X DVD_SPEED_IN_KBPS_1X * 48
```

File

StarBurn.h (see page 662)

Description

48X DVD speed constant (DVD+R and DVD-R)

2.4.54 DVD_SPEED_IN_KBPS_4X Macro

C++

```
#define DVD_SPEED_IN_KBPS_4X 5540
```

File

StarBurn.h (see page 662)

Description

4X DVD speed constant (DVD-R, DVD-R DL, DVD-RW, DVD+R, DVD+R DL, DVD+RW and DVD-RAM)

2.4.55 DVD_SPEED_IN_KBPS_50X Macro

C++

```
#define DVD_SPEED_IN_KBPS_50X DVD_SPEED_IN_KBPS_1X * 50
```

File

StarBurn.h (see page 662)

Description

50X DVD speed constant (DVD+R and DVD-R)

2.4.56 DVD_SPEED_IN_KBPS_5X Macro

C++

```
#define DVD_SPEED_IN_KBPS_5X 6925
```

File

StarBurn.h (see page 662)

Description

5X DVD speed constant (DVD-RAM)

2.4.57 DVD_SPEED_IN_KBPS_6X Macro

C++

```
#define DVD_SPEED_IN_KBPS_6X 8310
```

File

StarBurn.h (see page 662)

Description

6X DVD speed constant (DVD-RW)

2.4.58 DVD_SPEED_IN_KBPS_8X Macro

C++

```
#define DVD_SPEED_IN_KBPS_8X 11080
```

File

StarBurn.h (see page 662)

Description

8X DVD speed constant (DVD-R, DVD-R DL, DVD+R, DVD+R DL and DVD+RW)

2.4.59 ELTORITO_BOOTABLE Macro

C++

```
#define ELTORITO_BOOTABLE 0x88
```

File

StarBurn.h (see page 662)

Description

Bootable disc ID

2.4.60 ELTORITO_BOOTSYSTEM_ID Macro

C++

```
#define ELTORITO_BOOTSYSTEM_ID "EL TORITO SPECIFICATION"
```

File

StarBurn.h (see page 662)

Description

Bootable specification signature

2.4.61 ELTORITO_DEF_LOAD_SEGMENT Macro

C++

```
#define ELTORITO_DEF_LOAD_SEGMENT 0x07C0
```

File

StarBurn.h (see page 662)

Description

Real mode code load segment

2.4.62 ELTORITO_NON_BOOTABLE Macro

C++

```
#define ELTORITO_NON_BOOTABLE 0x00
```

File

StarBurn.h (see page 662)

Description

Non-bootable disc ID

2.4.63 ERROR_FIELD_C2_AND_BLOCK_ERROR Macro

C++

```
#define ERROR_FIELD_C2_AND_BLOCK_ERROR 0x02 // C2 and block error information
```

File

StarBurn.h (see page 662)

Description

C2 and block error information

2.4.64 ERROR_FIELD_C2_ERROR Macro

C++

```
#define ERROR_FIELD_C2_ERROR 0x01 // C2 error information
```

File

StarBurn.h (see page 662)

Description

C2 error information

2.4.65 ERROR_FIELD_NO_ERROR Macro

C++

```
#define ERROR_FIELD_NO_ERROR 0x00 // No error information
```

File

StarBurn.h (see page 662)

Description

No error information

2.4.66 ERROR_FIELD_RESERVED Macro

C++

```
#define ERROR_FIELD_RESERVED 0x03 // Reserved
```

File

StarBurn.h (see page 662)

Description

Reserved

2.4.67 EXPECTED_LB_TYPE_ANY Macro

C++

```
#define EXPECTED_LB_TYPE_ANY 0x00 // Accept all LBs
```

File

StarBurn.h (see page 662)

Description

Accept all LBs

2.4.68 EXPECTED_LB_TYPE_CDDA Macro

C++

```
#define EXPECTED_LB_TYPE_CDDA 0x01 // Accept only CDDA (CD digital audio) LBs
```

File

StarBurn.h (see page 662)

Description

Accept only CDDA (CD digital audio) LBs

2.4.69 EXPECTED_LB_TYPE_MODE1 Macro

C++

```
#define EXPECTED_LB_TYPE_MODE1 0x02 // Accept only MODE1 LBs
```

File

StarBurn.h (see page 662)

Description

Accept only MODE1 LBs

2.4.70 EXPECTED_LB_TYPE_MODE2 Macro

C++

```
#define EXPECTED_LB_TYPE_MODE2 0x03 // Accept only MODE2 LBs (both Form1 and Form2)
```

File

StarBurn.h (see page 662)

Description

Accept only MODE2 LBs (both Form1 and Form2)

2.4.71 EXPECTED_LB_TYPE_MODE2_FORM1 Macro

C++

```
#define EXPECTED_LB_TYPE_MODE2_FORM1 0x04 // Accept only MODE2 Form1 LBs
```

File

StarBurn.h (see page 662)

Description

Accept only MODE2 Form1 LBs

2.4.72 EXPECTED_LB_TYPE_MODE2_FORM2 Macro

C++

```
#define EXPECTED_LB_TYPE_MODE2_FORM2 0x05 // Accept only MODE2 Form2 LBs
```

File

StarBurn.h (see page 662)

Description

Accept only MODE2 Form2 LBs

2.4.73 HARD_DISK_LB_SIZE_IN_UCHARS Macro

C++

```
#define HARD_DISK_LB_SIZE_IN_UCHARS 512
```

File

StarBurn.h (see page 662)

Description

Hard disk LB (logical block) size in UCHARs

2.4.74 HEADER_CODE_ALL_HEADERS Macro

C++

```
#define HEADER_CODE_ALL_HEADERS 0x03 // Both header and subheader
```

File

StarBurn.h (see page 662)

Description

Both header and subheader

2.4.75 HEADER_CODE_HEADER_ONLY Macro

C++

```
#define HEADER_CODE_HEADER_ONLY 0x01 // 4-byte header
```

File

StarBurn.h (see page 662)

Description

4-byte header

2.4.76 HEADER_CODE_NONE Macro

C++

```
#define HEADER_CODE_NONE 0x00 // No header
```

File

StarBurn.h (see page 662)

Description

No header

2.4.77 HEADER_CODE_SUBHEADER_ONLY Macro

C++

```
#define HEADER_CODE_SUBHEADER_ONLY 0x02 // MODE2 Form1 or Form2 subheader
```

File

StarBurn.h (see page 662)

Description

MODE2 Form1 or Form2 subheader

2.4.78 ISO9660_DATE_SIZE_IN_UCHARS Macro

C++

```
#define ISO9660_DATE_SIZE_IN_UCHARS 17
```

File

StarBurn.h (see page 662)

Description

ISO9660 date size in UCHARs

2.4.79 ISO9660_FILE_SIZE_IN_UCHARS Macro

C++

```
#define ISO9660_FILE_SIZE_IN_UCHARS 37
```

File

StarBurn.h (see page 662)

Description

ISO9660 file size in UCHARs

2.4.80 ISO9660_NAME_SIZE_IN_UCHARS Macro

C++

```
#define ISO9660_NAME_SIZE_IN_UCHARS 128
```

File

StarBurn.h (see page 662)

Description

ISO9660 name size in UCHARs

2.4.81 ISO9660_SYSTEM_VOLUME_ID_SIZE_IN_UCHARS Macro

C++

```
#define ISO9660_SYSTEM_VOLUME_ID_SIZE_IN_UCHARS 32
```

File

StarBurn.h (see page 662)

Description

ISO9660 volume descriptor system and volume ID size in UCHARs

2.4.82 ISO9660_TREE_LEVEL Macro

C++

```
#define ISO9660_TREE_LEVEL 8
```

File

StarBurn.h (see page 662)

Description

ISO9660 top supported file tree level

2.4.83 LAST_SESSION_COMPLETE Macro

C++

```
#define LAST_SESSION_COMPLETE 0x03
```

File

StarBurn.h (see page 662)

Description

Last session is complete

2.4.84 LAST_SESSION_EMPTY Macro

C++

```
#define LAST_SESSION_EMPTY 0x00
```

File

StarBurn.h (see page 662)

Description

Last session is empty

2.4.85 LAST_SESSION_INCOMPLETE Macro

C++

```
#define LAST_SESSION_INCOMPLETE 0x01
```

File

StarBurn.h (see page 662)

Description

Last session is incomplete

2.4.86 MODE1_LB_SIZE_IN_UCHARS Macro

C++

```
#define MODE1_LB_SIZE_IN_UCHARS 2048
```

File

StarBurn.h (see page 662)

Description

MODE1 LB (logical block) size in UCHARs

2.4.87 MODE2_FORM1_LB_SIZE_IN_UCHARS Macro

C++

```
#define MODE2_FORM1_LB_SIZE_IN_UCHARS 2048
```

File

StarBurn.h (see page 662)

Description

MODE2 Form1 LB (logical block) size in UCHARs

2.4.88 MODE2_FORM2_LB_SIZE_IN_UCHARS Macro

C++

```
#define MODE2_FORM2_LB_SIZE_IN_UCHARS 2324
```

File

StarBurn.h (see page 662)

Description

MODE2 Form2 LB (logical block) size in UCHARs

2.4.89 MODE2_FORM2_SUB_LB_SIZE_IN_UCHARS Macro

C++

```
#define MODE2_FORM2_SUB_LB_SIZE_IN_UCHARS 2332
```

File

StarBurn.h (see page 662)

Description

MODE2 Form2 + 8 UCHARs of subheader LB (logical block) size in UCHARs

2.4.90 MODE2_FORMLESS_LB_SIZE_IN_UCHARS Macro

C++

```
#define MODE2_FORMLESS_LB_SIZE_IN_UCHARS 2336
```

File

StarBurn.h (see page 662)

Description

MODE2 Formless LB (logical block) size in UCHARS

2.4.91 NUMBER_OF_RAW_TRACKS Macro

C++

```
#define NUMBER_OF_RAW_TRACKS ( NUMBER_OF_TRACKS * 4 + 1 )
```

File

StarBurn.h (see page 662)

Description

Top number of raw tracks (including maintenance tracks) on CD/DVD/Blu-Ray/HD-DVD disc.

This is calculated based on the worst CD case when all tracks will be in a separate sessions. We need this approximate value to avoid allocating TOC_INFORMATION (see page 573) structure dynamically.

2.4.92 NUMBER_OF_READ_RETRIES Macro

C++

```
#define NUMBER_OF_READ_RETRIES 1
```

File

StarBurn.h (see page 662)

Description

Default number of retries on read operations (if bad block was possibly hit, was 2 originally)

2.4.93 NUMBER_OF_SUBCHANNELS Macro

C++

```
#define NUMBER_OF_SUBCHANNELS 8 // sub-channels from P to W
```

File

StarBurn.h (see page 662)

Description

sub-channels from P to W

2.4.94 NUMBER_OF_TRACKS Macro

C++

```
#define NUMBER_OF_TRACKS 100
```

File

StarBurn.h (see page 662)

Description

Top number of tracks (or border-in/border-out zones) on CD/DVD/Blu-Ray/HD-DVD disc

2.4.95 PAGE_SIZE_IN_UCHARS Macro

C++

```
#define PAGE_SIZE_IN_UCHARS 4096
```

File

StarBurn.h (see page 662)

Description

Small page size in UCHARs on x86 machines (default allocatable storage)

2.4.96 RAW_LB_PQ_SUB_SIZE_IN_UCHARS Macro

C++

```
#define RAW_LB_PQ_SUB_SIZE_IN_UCHARS 2368
```

File

StarBurn.h (see page 662)

Description

RAW LB (logical block) with PQ sub-channel size in UCHARs

2.4.97 RAW_LB_RAW_PW_SUB_SIZE_IN_UCHARS Macro

C++

```
#define RAW_LB_RAW_PW_SUB_SIZE_IN_UCHARS 2448
```

File

StarBurn.h (see page 662)

Description

RAW LB (logical block) with RAW P-W sub-channel size in UCHARs

2.4.98 RAW_LB_SIZE_IN_UCHARS Macro

C++

```
#define RAW_LB_SIZE_IN_UCHARS 2352
```

File

StarBurn.h (see page 662)

Description

RAW LB (logical block) size in UCHARs

2.4.99 READ_REPORT_DELAY_IN_SECONDS Macro

C++

```
#define READ_REPORT_DELAY_IN_SECONDS 1
```

File

StarBurn.h (see page 662)

Description

Default read report delay in seconds

2.4.100 SCSI_DEVICE_DIRECT_ACCESS Macro

C++

```
#define SCSI_DEVICE_DIRECT_ACCESS 0x00
```

File

StarBurn.h (see page 662)

Description

SCSI disk device

2.4.101 SCSI_DEVICE_MAGNETO_OPTICAL Macro

C++

```
#define SCSI_DEVICE_MAGNETO_OPTICAL 0x07
```

File

StarBurn.h (see page 662)

Description

SCSI MO (Magneto-Optical) disk device

2.4.102 SCSI_DEVICE_MEDIUM_CHANGER Macro

C++

```
#define SCSI_DEVICE_MEDIUM_CHANGER 0x08
```

File

StarBurn.h (see page 662)

Description

SCSI jukebox (disc changer) device

2.4.103 SCSI_DEVICE_NETWORK Macro

C++

```
#define SCSI_DEVICE_NETWORK 0x09
```

File

StarBurn.h (see page 662)

Description

SCSI network device

2.4.104 SCSI_DEVICE_PRINTER Macro

C++

```
#define SCSI_DEVICE_PRINTER 0x02
```

File

StarBurn.h (see page 662)

Description

SCSI printer device

2.4.105 SCSI_DEVICE_PROCESSOR Macro

C++

```
#define SCSI_DEVICE_PROCESSOR 0x03
```

File

StarBurn.h (see page 662)

Description

SCSI processor device

2.4.106 SCSI_DEVICE_RO_DIRECT_ACCESS Macro

C++

```
#define SCSI_DEVICE_RO_DIRECT_ACCESS 0x05
```

File

StarBurn.h (see page 662)

Description

SCSI CD/DVD/Blu-Ray/HD-DVD device

2.4.107 SCSI_DEVICE_SCANNER Macro

C++

```
#define SCSI_DEVICE_SCANNER 0x06
```

File

StarBurn.h (see page 662)

Description

SCSI scanner device

2.4.108 SCSI_DEVICE_SEQUENTIAL_ACCESS Macro

C++

```
#define SCSI_DEVICE_SEQUENTIAL_ACCESS 0x01
```

File

StarBurn.h (see page 662)

Description

SCSI tape device

2.4.109 SCSI_DEVICE_UNPRESENT Macro

C++

```
#define SCSI_DEVICE_UNPRESENT 0x7F
```

File

StarBurn.h (see page 662)

Description

SCSI device... Is gone!

2.4.110 SCSI_DEVICE_WILDCARD Macro

C++

```
#define SCSI_DEVICE_WILDCARD 0xFF
```

File

StarBurn.h (see page 662)

Description

SCSI device any (wildcard for StarBurn_FindDevice (see page 231)() function)

2.4.111 SCSI_DEVICE_WORM Macro

C++

```
#define SCSI_DEVICE_WORM 0x04
```

File

StarBurn.h (see page 662)

Description

SCSI WORM (Write-Once-Read-Multiple) device

2.4.112 SHORT_RAW_LB_SIZE_IN_UCHARS Macro

C++

```
#define SHORT_RAW_LB_SIZE_IN_UCHARS 2336
```

File

StarBurn.h (see page 662)

Description

Short (w/o sync and header) RAW LB (logical block) size in UCHARs

2.4.113 SINGLE_LAYER_DVD_CAPACITY_SIZE_IN_LBS Macro

C++

```
#define SINGLE_LAYER_DVD_CAPACITY_SIZE_IN_LBS 2298495 // Number of logical blocks on single layer DVD
```

File

StarBurn.h (see page 662)

Description

Number of logical blocks on single layer DVD

2.4.114 SMALLEST_CACHE_SIZE_IN_MBS Macro

C++

```
#define SMALLEST_CACHE_SIZE_IN_MBS 64
```

File

StarBurn.h (see page 662)

Description

Smallest cache size in MBs that toolkit will allow to use to buffer "write" operations

2.4.115 SMALLEST_CACHE_SIZE_IN_UCHARS Macro

C++

```
#define SMALLEST_CACHE_SIZE_IN_UCHARS ( SMALLEST_CACHE_SIZE_IN_MBS * 1024 * 1024 )
```

File

StarBurn.h (see page 662)

Description

Smallest cache size in UCHARs that toolkit will allow to use to buffer "write" operations

2.4.116 STARBURN_BDRE_FORMAT_CERTIFICATION_FULL Macro

C++

```
#define STARBURN_BDRE_FORMAT_CERTIFICATION_FULL 0x01
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_FULL.

2.4.117 STARBURN_BDRE_FORMAT_CERTIFICATION_NO Macro

C++

```
#define STARBURN_BDRE_FORMAT_CERTIFICATION_NO 0x00
```

File

StarBurn.h (see page 662)

Description

BD-RE format certification types

2.4.118 STARBURN_BDRE_FORMAT_CERTIFICATION_QUICK Macro

C++

```
#define STARBURN_BDRE_FORMAT_CERTIFICATION_QUICK 0x02
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_QUICK.

2.4.119 STARBURN_BDRE_FORMAT_CERTIFICATION_RESERVED Macro

C++

```
#define STARBURN_BDRE_FORMAT_CERTIFICATION_RESERVED 0x03
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_RESERVED.

2.4.120 STARBURN_BDRE_FORMAT_DEFAULT Macro

C++

```
#define STARBURN_BDRE_FORMAT_DEFAULT 0x00
```

File

StarBurn.h (see page 662)

Description

BD-RE format types

2.4.121 STARBURN_BDRE_FORMAT_SPARE_AREA_EXPANSION Macro

C++

```
#define STARBURN_BDRE_FORMAT_SPARE_AREA_EXPANSION 0x01
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_BDRE_FORMAT_SPARE_AREA_EXPANSION.

2.4.122 STARBURN_BDRE_FORMAT_WITH_SPARE_AREA Macro

C++

```
#define STARBURN_BDRE_FORMAT_WITH_SPARE_AREA 0x30
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_BDRE_FORMAT_WITH_SPARE_AREA.

2.4.123 STARBURN_BDRE_FORMAT_WITHOUT_SPARE_AREA Macro

C++

```
#define STARBURN_BDRE_FORMAT_WITHOUT_SPARE_AREA 0x31
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_BDRE_FORMAT_WITHOUT_SPARE_AREA.

2.4.124 STARBURN_CACHE_SIZE_READ_ONLY Macro

C++

```
#define STARBURN_CACHE_SIZE_READ_ONLY 1
```

File

StarBurn.h (see page 662)

Description

Default cache size for I/O operations

2.4.125 STARBURN_CACHE_SIZE_READ_WRITE Macro

C++

```
#define STARBURN_CACHE_SIZE_READ_WRITE 0
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_CACHE_SIZE_READ_WRITE.

2.4.126 STARBURN_CDFS2_FILE_ATTRIBUTE_DELETED Macro

C++

```
#define STARBURN_CDFS2_FILE_ATTRIBUTE_DELETED 0x00000008
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_DELETED.

2.4.127 STARBURN_CDFS2_FILE_ATTRIBUTE_DIRECTORY Macro

C++

```
#define STARBURN_CDFS2_FILE_ATTRIBUTE_DIRECTORY 0x00000001
```

File

StarBurn.h (see page 662)

Description

File attributes (flags for Attributes member of STARBURN_UDF2_FILE_INFO (see page 567) & STARBURN_ISO2_FILE_INFO)

2.4.128 STARBURN_CDFS2_FILE_ATTRIBUTE_HIDDEN Macro

C++

```
#define STARBURN_CDFS2_FILE_ATTRIBUTE_HIDDEN 0x00000004
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_HIDDEN.

2.4.129 STARBURN_CDFS2_FILE_ATTRIBUTE_IMPORTED Macro

C++

```
#define STARBURN_CDFS2_FILE_ATTRIBUTE_IMPORTED 0x00000002
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_IMPORTED.

2.4.130 StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileT
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileT.

2.4.131 StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileT
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileT.

2.4.132 StarBurn_CdvdBurnerGrabber_GetDeviceInformationT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_GetDeviceInformationT
StarBurn_CdvdBurnerGrabber_GetDeviceInformation
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_GetDeviceInformationT.

2.4.133 StarBurn_CdvdBurnerGrabber_GrabTrackT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_GrabTrackT StarBurn_CdvdBurnerGrabber_GrabTrack
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_GrabTrackT.

2.4.134 StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWT
StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWT.

2.4.135 StarBurn_CdvdBurnerGrabber_SessionAtOnceT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_SessionAtOnceT StarBurn_CdvdBurnerGrabber_SessionAtOnce
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_SessionAtOnceT.

2.4.136 StarBurn_CdvdBurnerGrabber_SuperVideoCDExExT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_SuperVideoCDExExT
StarBurn_CdvdBurnerGrabber_SuperVideoCDExEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDExExT.

2.4.137 StarBurn_CdvdBurnerGrabber_SuperVideoCDExT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_SuperVideoCDExT StarBurn_CdvdBurnerGrabber_SuperVideoCDEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDExT.

2.4.138 StarBurn_CdvdBurnerGrabber_SuperVideoCDT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_SuperVideoCDT StarBurn_CdvdBurnerGrabber_SuperVideoCD
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDT.

2.4.139 StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileT
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileT.

2.4.140 StarBurn_CdvdBurnerGrabber_VerifyFileExT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_VerifyFileExT StarBurn_CdvdBurnerGrabber_VerifyFileEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_VerifyFileExT.

2.4.141 StarBurn_CdvdBurnerGrabber_VerifyFileT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_VerifyFileT StarBurn_CdvdBurnerGrabber_VerifyFile
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_VerifyFileT.

2.4.142 StarBurn_CdvdBurnerGrabber_VideoCDExExT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_VideoCDExExT StarBurn_CdvdBurnerGrabber_VideoCDExEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_VideoCDExExT.

2.4.143 StarBurn_CdvdBurnerGrabber_VideoCDExT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_VideoCDExT StarBurn_CdvdBurnerGrabber_VideoCDEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_VideoCDExT.

2.4.144 StarBurn_CdvdBurnerGrabber_VideoCDT Macro

C++

```
#define StarBurn_CdvdBurnerGrabber_VideoCDT StarBurn_CdvdBurnerGrabber_VideoCD
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_CdvdBurnerGrabber_VideoCDT.

2.4.145 STARBURN_DEBUG_FACILITY_ALLMESSAGES Macro

C++

```
#define STARBURN_DEBUG_FACILITY_ALLMESSAGES 0xFFF00000
```

File

StarBurn.h (see page 662)

Description

Debug facility: Permit All Messages

2.4.146 STARBURN_DEBUG_FACILITY_ALLTARGETS Macro

C++

```
#define STARBURN_DEBUG_FACILITY_ALLTARGETS 0x00000FFF
```

File

StarBurn.h (see page 662)

Description

Debug facility: All targets

2.4.147 STARBURN_DEBUG_FACILITY_CUSTOM Macro

C++

```
#define STARBURN_DEBUG_FACILITY_CUSTOM 0x00000008
```

File

StarBurn.h (see page 662)

Description

Debug facility: Custom

2.4.148 STARBURN_DEBUG_FACILITY_DEBUG Macro

C++

```
#define STARBURN_DEBUG_FACILITY_DEBUG 0x00200000
```

File

StarBurn.h (see page 662)

Description

Debug facility: Permit Debug Messages

2.4.149 STARBURN_DEBUG_FACILITY_DEBUG_OUTPUT Macro

C++

```
#define STARBURN_DEBUG_FACILITY_DEBUG_OUTPUT 0x00000001
```

File

StarBurn.h (see page 662)

Description

Debug facility: Debug output

2.4.150 STARBURN_DEBUG_FACILITY_ERROR Macro

C++

```
#define STARBURN_DEBUG_FACILITY_ERROR 0x00800000
```

File

StarBurn.h (see page 662)

Description

Debug facility: Permit Error Messages

2.4.151 STARBURN_DEBUG_FACILITY_LOG_FILE Macro

C++

```
#define STARBURN_DEBUG_FACILITY_LOG_FILE 0x00000002
```

File

StarBurn.h (see page 662)

Description

Debug facility: Log file

2.4.152 STARBURN_DEBUG_FACILITY_NONE Macro

C++

```
#define STARBURN_DEBUG_FACILITY_NONE 0x00000000
```

File

StarBurn.h (see page 662)

Description

Debug facility: None

2.4.153 STARBURN_DEBUG_FACILITY_SYSTEM_CONSOLE Macro

C++

```
#define STARBURN_DEBUG_FACILITY_SYSTEM_CONSOLE 0x00000004
```

File

StarBurn.h (see page 662)

Description

Debug facility: System console

2.4.154 STARBURN_DEBUG_FACILITY_TRACE Macro

C++

```
#define STARBURN_DEBUG_FACILITY_TRACE 0x00100000
```

File

StarBurn.h (see page 662)

Description

Debug facility: Permit Trace Messages

2.4.155 STARBURN_DEBUG_FACILITY_WARNING Macro

C++

```
#define STARBURN_DEBUG_FACILITY_WARNING 0x00400000
```

File

StarBurn.h (see page 662)

Description

Debug facility: Permit Warning Messages

2.4.156 STARBURN_DISC_TYPE_CDDA Macro

C++

```
#define STARBURN_DISC_TYPE_CDDA 0x01
```

File

StarBurn.h (see page 662)

Description

Disc type is CDDA (CD-ROM or Digital Audio)

2.4.157 STARBURN_DISC_TYPE_CDI Macro

C++

```
#define STARBURN_DISC_TYPE_CDI 0x10
```

File

StarBurn.h (see page 662)

Description

Disc type is CDI

2.4.158 STARBURN_DISC_TYPE_UNDEFINED Macro

C++

```
#define STARBURN_DISC_TYPE_UNDEFINED 0xFF
```

File

StarBurn.h (see page 662)

Description

Disc type is undefined

2.4.159 STARBURN_DISC_TYPE_XA Macro

C++

```
#define STARBURN_DISC_TYPE_XA 0x20
```

File

StarBurn.h (see page 662)

Description

Disc type is CD-ROM XA

2.4.160 STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES Macro

C++

```
#define STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES 99
```

File

StarBurn.h (see page 662)

Description

DVD-Video maximum number of titles

2.4.161 StarBurn_DVDVideo_CreateT Macro

C++

```
#define StarBurn_DVDVideo_CreateT StarBurn_DVDVideo_Create
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_DVDVideo_CreateT.

2.4.162 StarBurn_ElTorito_BootCatalogAddSectionT Macro

C++

```
#define StarBurn_ElTorito_BootCatalogAddSectionT StarBurn_ElTorito_BootCatalogAddSection
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_ElTorito_BootCatalogAddSectionT.

2.4.163 StarBurn_ElTorito_CreateBootCatalogT Macro

C++

```
#define StarBurn_ElTorito_CreateBootCatalogT StarBurn_ElTorito_CreateBootCatalog
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_ElTorito_CreateBootCatalogT.

2.4.164 StarBurn_GetAudioFileStreamSizeInUCHARs_FastT Macro

C++

```
#define StarBurn_GetAudioFileStreamSizeInUCHARs_FastT
StarBurn_GetAudioFileStreamSizeInUCHARs_Fast
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_GetAudioFileStreamSizeInUCHARs_FastT.

2.4.165 StarBurn_GetAudioFileStreamSizeInUCHARsT Macro

C++

```
#define StarBurn_GetAudioFileStreamSizeInUCHARsT StarBurn_GetAudioFileStreamSizeInUCHARs
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_GetAudioFileStreamSizeInUCHARsT.

2.4.166 StarBurn_GetDeviceLetterT Macro

C++

```
#define StarBurn_GetDeviceLetterT StarBurn_GetDeviceLetter
```

File

StarBurn.h (see page 662)

Description

TCHAR functions

2.4.167 STARBURN_IMPEX_API Macro

C++

```
#define STARBURN_IMPEX_API
```

File

StarBurn.h (see page 662)

Description

declspec(dllexport)

2.4.168 StarBurn_IsAudioFileSupportedT Macro

C++

```
#define StarBurn_IsAudioFileSupportedT StarBurn_IsAudioFileSupported
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_IsAudioFileSupportedT.

2.4.169 StarBurn_ISO2_VolumeCreateBootElToritoT Macro

C++

```
#define StarBurn_ISO2_VolumeCreateBootElToritoT StarBurn_ISO2_VolumeCreateBootElTorito
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_ISO2_VolumeCreateBootElToritoT.

2.4.170 STARBURN_ISO9660_FILE_FLAGS_ASSOCIATED_FILE Macro

C++

```
#define STARBURN_ISO9660_FILE_FLAGS_ASSOCIATED_FILE 0x04
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_FILE_FLAGS_ASSOCIATED_FILE.

2.4.171 STARBURN_ISO9660_FILE_FLAGS_DIRECTORY Macro

C++

```
#define STARBURN_ISO9660_FILE_FLAGS_DIRECTORY 0x02
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_FILE_FLAGS_DIRECTORY.

2.4.172 STARBURN_ISO9660_FILE_FLAGS_EXISTENCE Macro

C++

```
#define STARBURN_ISO9660_FILE_FLAGS_EXISTENCE 0x01
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_FILE_FLAGS_EXISTENCE.

2.4.173 STARBURN_ISO9660_FILE_FLAGS_NONE Macro

C++

```
#define STARBURN_ISO9660_FILE_FLAGS_NONE 0x00
```

File

StarBurn.h (see page 662)

Description

ISO9660 file attributes flags

2.4.174 STARBURN_ISO9660_FILE_FLAGS_RECORD Macro

C++

```
#define STARBURN_ISO9660_FILE_FLAGS_RECORD 0x08
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_FILE_FLAGS_RECORD.

2.4.175 STARBURN_ISO9660_FILE_MAX_SIZE_IN_UCHARS Macro

C++

```
#define STARBURN_ISO9660_FILE_MAX_SIZE_IN_UCHARS ( LONGLONG )( 0xFFFFFFFF ) // ( 4GB - 1 )
```

File

StarBurn.h (see page 662)

Description

(4GB - 1)

2.4.176 STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS Macro

C++

```
#define STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS 128
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS.

2.4.177 STARBURN_ISO9660_JOLIET_RELAXED_NAME_SIZE_IN_WCHARS Macro

C++

```
#define STARBURN_ISO9660_JOLIET_RELAXED_NAME_SIZE_IN_WCHARS 108 // TODO: Check it!!!
```

File

StarBurn.h (see page 662)

Todo

Check it!!!

2.4.178 STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS Macro

C++

```
#define STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS 256
```

File

StarBurn.h (see page 662)

Description

ISO9660 constants (begin)

2.4.179 STARBURN_ISO9660_TYPE_1999 Macro

C++

```
#define STARBURN_ISO9660_TYPE_1999 2
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_TYPE_1999.

2.4.180 STARBURN_ISO9660_TYPE_JOLIET Macro

C++

```
#define STARBURN_ISO9660_TYPE_JOLIET 4
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_TYPE_JOLIET.

2.4.181 STARBURN_ISO9660_TYPE_JOLIET_RELAXED Macro

C++

```
#define STARBURN_ISO9660_TYPE_JOLIET_RELAXED 12
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_TYPE_JOLIET_RELAXED.

2.4.182 STARBURN_ISO9660_TYPE_LEVEL1 Macro

C++

```
#define STARBURN_ISO9660_TYPE_LEVEL1 0
```

File

StarBurn.h (see page 662)

Description

ISO9660 file system type

2.4.183 STARBURN_ISO9660_TYPE_LEVEL2 Macro

C++

```
#define STARBURN_ISO9660_TYPE_LEVEL2 1
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_ISO9660_TYPE_LEVEL2.

2.4.184 STARBURN_LOADING_MECHANISM_TYPE_CADDY Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_CADDY 0x00
```

File

StarBurn.h (see page 662)

Description

Loading mechanism type constants

2.4.185 STARBURN_LOADING_MECHANISM_TYPE_CARTRIDGE Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_CARTRIDGE 0x05
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_LOADING_MECHANISM_TYPE_CARTRIDGE.

2.4.186 STARBURN_LOADING_MECHANISM_TYPE_CHANGER Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_CHANGER 0x04
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_LOADING_MECHANISM_TYPE_CHANGER.

2.4.187 STARBURN_LOADING_MECHANISM_TYPE_POP_UP Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_POP_UP 0x02
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_LOADING_MECHANISM_TYPE_POP_UP.

2.4.188 STARBURN_LOADING_MECHANISM_TYPE_RESERVED1 Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_RESERVED1 0x03
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED1.

2.4.189 STARBURN_LOADING_MECHANISM_TYPE_RESERVED2 Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_RESERVED2 0x06
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED2.

2.4.190 STARBURN_LOADING_MECHANISM_TYPE_RESERVED3 Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_RESERVED3 0x07
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED3.

2.4.191 STARBURN_LOADING_MECHANISM_TYPE_TRAY Macro

C++

```
#define STARBURN_LOADING_MECHANISM_TYPE_TRAY 0x01
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_LOADING_MECHANISM_TYPE_TRAY.

2.4.192 STARBURN_PIPE_SIZE_IN_UCHARS Macro

C++

```
#define STARBURN_PIPE_SIZE_IN_UCHARS ( 60 * 64 * 1024 ) // 60, 64 or 32 KB I/O packets
```

File

StarBurn.h (see page 662)

Description

60, 64 or 32 KB I/O packets

2.4.193 STARBURN_SMALLEST_SPLIT_FILE_SIZE_IN_MBS Macro

C++

```
#define STARBURN_SMALLEST_SPLIT_FILE_SIZE_IN_MBS 200
```

File

StarBurn.h (see page 662)

Description

StarBurn smallest split file size in megabytes

2.4.194 StarBurn_StarWave_CompressedFileReaderObjectCreateT Macro

C++

```
#define StarBurn_StarWave_CompressedFileReaderObjectCreateT
StarBurn_StarWave_CompressedFileReaderObjectCreate
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave_CompressedFileReaderObjectCreateT.

2.4.195 StarBurn_StarWave_CompressedFileRecompressT Macro

C++

```
#define StarBurn_StarWave_CompressedFileRecompressT
StarBurn_StarWave_CompressedFileRecompress
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave_CompressedFileRecompressT.

2.4.196 StarBurn_StarWave_CompressedFileUncompressT Macro

C++

```
#define StarBurn_StarWave_CompressedFileUncompressT
StarBurn_StarWave_CompressedFileUncompress
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave_CompressedFileUncompressT.

2.4.197 StarBurn_StarWave_CompressedFileWriterObjectCreateT Macro

C++

```
#define StarBurn_StarWave_CompressedFileWriterObjectCreateT
StarBurn_StarWave_CompressedFileWriterObjectCreate
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave_CompressedFileWriterObjectCreateT.

2.4.198 STARBURN_STARWAVE_IO_BUFFER_SIZE_IN_UCHARS Macro

C++

```
#define STARBURN_STARWAVE_IO_BUFFER_SIZE_IN_UCHARS ( 44100 * 2 * 2 )
```

File

StarBurn.h (see page 662)

Description

StarWave recommended I/O buffer size in UCHARs. It's *STRONGLY* recommended to use either it or multiplications of STARWAVE_IO_BUFFER_SIZE_IN_UCHARS as internal calculations and indexations are integer rather then float based

2.4.199 StarBurn_StarWave_UncompressedFileCompressT Macro

C++

```
#define StarBurn_StarWave_UncompressedFileCompressT
StarBurn_StarWave_UncompressedFileCompress
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave_UncompressedFileCompressT.

2.4.200 StarBurn_StarWave_UncompressedFileSupportedIsT Macro

C++

```
#define StarBurn_StarWave_UncompressedFileSupportedIsT
StarBurn_StarWave_UncompressedFileSupportedIs
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave_UncompressedFileSupportedIsT.

2.4.201 StarBurn_StarWave2_ConvertExT Macro

C++

```
#define StarBurn_StarWave2_ConvertExT StarBurn_StarWave2_ConvertEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave2_ConvertExT.

2.4.202 StarBurn_StarWave2_EncodeMP3OGGFromWAVT Macro

C++

```
#define StarBurn_StarWave2_EncodeMP3OGGFromWAVT StarBurn_StarWave2_EncodeMP3OGGFromWAV
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave2_EncodeMP3OGGFromWAVT.

2.4.203 StarBurn_StarWave2_GetFileReaderAndFactoryT Macro

C++

```
#define StarBurn_StarWave2_GetFileReaderAndFactoryT
StarBurn_StarWave2_GetFileReaderAndFactory
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_StarWave2_GetFileReaderAndFactoryT.

2.4.204 StarBurn_UDF_AddT Macro

C++

```
#define StarBurn_UDF_AddT StarBurn_UDF_Add
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_UDF_AddT.

2.4.205 StarBurn_UDF_CreateExT Macro

C++

```
#define StarBurn_UDF_CreateExT StarBurn_UDF_CreateEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_UDF_CreateExT.

2.4.206 StarBurn_UDF_CreateT Macro

C++

```
#define StarBurn_UDF_CreateT StarBurn_UDF_Create
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_UDF_CreateT.

2.4.207 StarBurn_UDF_FormatTreeItemAsDirectoryT Macro

C++

```
#define StarBurn_UDF_FormatTreeItemAsDirectoryT StarBurn_UDF_FormatTreeItemAsDirectory
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_UDF_FormatTreeItemAsDirectoryT.

2.4.208 StarBurn_UDF_FormatTreeItemAsFileT Macro

C++

```
#define StarBurn_UDF_FormatTreeItemAsFileT StarBurn_UDF_FormatTreeItemAsFile
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_UDF_FormatTreeItemAsFileT.

2.4.209 STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS Macro

C++

```
#define STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS 255
```

File

StarBurn.h (see page 662)

Description

Name constants

2.4.210 STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS Macro

C++

```
#define STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS 2048
```

File

StarBurn.h (see page 662)

Description

This is macro STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS.

2.4.211 StarBurn_UDF2_VolumeCreateBootElToritoT Macro

C++

```
#define StarBurn_UDF2_VolumeCreateBootElToritoT StarBurn_UDF2_VolumeCreateBootElTorito
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_UDF2_VolumeCreateBootElToritoT.

2.4.212 StarBurn_UDFBridge_CreateExT Macro

C++

```
#define StarBurn_UDFBridge_CreateExT StarBurn_UDFBridge_CreateEx
```

File

StarBurn.h (see page 662)

Description

This is macro StarBurn_UDFBridge_CreateExT.

2.4.213 STARPORT_DEVICE_NAME_SIZE_IN_UCHARS Macro

C++

```
#define STARPORT_DEVICE_NAME_SIZE_IN_UCHARS 1024
```

File

StarBurn.h (see page 662)

Description

StarPort device name size in UCHARs

2.4.214 STARWAVEFACTORY_ID_DSHOW Macro

C++

```
#define STARWAVEFACTORY_ID_DSHOW "9674798D-F70F-4034-920E-54520F76B3CE"
```

File

StarBurn.h (see page 662)

Description

This is macro STARWAVEFACTORY_ID_DSHOW.

2.4.215 STARWAVEFACTORY_ID_MP3 Macro

C++

```
#define STARWAVEFACTORY_ID_MP3 "C72EBA33-7D5E-42d8-8017-C497103B34B5"
```

File

StarBurn.h (see page 662)

Description

This is macro STARWAVEFACTORY_ID_MP3.

2.4.216 STARWAVEFACTORY_ID_OGG Macro

C++

```
#define STARWAVEFACTORY_ID_OGG "6DE5E9B0-7D81-480b-94F4-F3C96C172C38"
```

File

StarBurn.h (see page 662)

Description

This is macro STARWAVEFACTORY_ID_OGG.

2.4.217 STARWAVEFACTORY_ID_WAV Macro

C++

```
#define STARWAVEFACTORY_ID_WAV "242C65F5-9B18-4a9b-BA0F-FF498A2E7AB7"
```

File

StarBurn.h (see page 662)

Description

StarWave2 factory ids

2.4.218 STARWAVEFACTORY_ID_WMA Macro

C++

```
#define STARWAVEFACTORY_ID_WMA "DFE900CD-DC98-493a-A0C5-EC3F7A8A6724"
```

File

StarBurn.h (see page 662)

Description

This is macro STARWAVEFACTORY_ID_WMA.

2.4.219 SUBCHANNEL_NO Macro

C++

```
#define SUBCHANNEL_NO 0x00 // No sub-channel
```

File

StarBurn.h (see page 662)

Description

No sub-channel

2.4.220 SUBCHANNEL_PQ Macro

C++

```
#define SUBCHANNEL_PQ 0x02 // PQ sub-channel
```

File

StarBurn.h (see page 662)

Description

PQ sub-channel

2.4.221 SUBCHANNEL_PQ_SIZE_IN_UCHARS Macro

C++

```
#define SUBCHANNEL_PQ_SIZE_IN_UCHARS 16 // PQ sub-channel size in UCHARs
```

File

StarBurn.h (see page 662)

Description

PQ sub-channel size in UCHARs

2.4.222 SUBCHANNEL_RAW_PW Macro

C++

```
#define SUBCHANNEL_RAW_PW 0x01 // RAW P-W sub-channel
```

File

StarBurn.h (see page 662)

Description

RAW P-W sub-channel

2.4.223 SUBCHANNEL_RAW_PW_SIZE_IN_UCHARS Macro

C++

```
#define SUBCHANNEL_RAW_PW_SIZE_IN_UCHARS 96 // Raw P-W sub-channel size in UCHARs
```

File

StarBurn.h (see page 662)

Description

Raw P-W sub-channel size in UCHARs

2.4.224 SUBCHANNEL_RESERVED Macro

C++

```
#define SUBCHANNEL_RESERVED 0x03 // Reserved
```

File

StarBurn.h (see page 662)

Description

Reserved

2.4.225 SUBCHANNEL_RW Macro

C++

```
#define SUBCHANNEL_RW 0x04 // R-W sub-channel
```

File

StarBurn.h (see page 662)

Description

R-W sub-channel

2.4.226 SUBCHANNEL_SIZE_IN_UCHARS Macro

C++

```
#define SUBCHANNEL_SIZE_IN_UCHARS 12 // Sub-channel size in UCHARs
```

File

StarBurn.h (see page 662)

Description

Sub-channel size in UCHARs

2.4.227 TRACK_NUMBER_INVISIBLE Macro

C++

```
#define TRACK_NUMBER_INVISIBLE 0xFF
```

File

StarBurn.h (see page 662)

Description

Special "invisible" track number

2.4.228 TYPE_CODE_LAST_CHANCE Macro

C++

```
#define TYPE_CODE_LAST_CHANCE 0x02 // Logical Unit region is set. Additional restrictions required to make a change
```

File

StarBurn.h (see page 662)

Description

Logical Unit region is set. Additional restrictions required to make a change

2.4.229 TYPE_CODE_NONE Macro

C++

```
#define TYPE_CODE_NONE 0x00 // No Logical Unit region setting
```

File

StarBurn.h (see page 662)

Description

No Logical Unit region setting

2.4.230 TYPE_CODE_PERM Macro

C++

```
#define TYPE_CODE_PERM 0x03 // Logical Unit region has been set permanently, but may be reset by vendor if necessery
```

File

StarBurn.h (see page 662)

Description

Logical Unit region has been set permanently, but may be reset by vendor if necessery

2.4.231 TYPE_CODE_SET Macro

C++

```
#define TYPE_CODE_SET 0x01 // Logical Unit region is set
```

File

StarBurn.h (see page 662)

Description

Logical Unit region is set

2.4.232 UDF_FILE_SET_DESCRIPTOR_LBA Macro

C++

```
#define UDF_FILE_SET_DESCRIPTOR_LBA ( UDF_ISO9660_FILE_SYSTEM_LBA +
UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS )
```

File

StarBurn.h (see page 662)

Description

UDF file set descriptor logical block address

2.4.233 UDF_FILE_SET_DESCRIPTOR_TERMINATOR_LBA Macro

C++

```
#define UDF_FILE_SET_DESCRIPTOR_TERMINATOR_LBA ( 1 + UDF_FILE_SET_DESCRIPTOR_LBA )
```

File

StarBurn.h (see page 662)

Description

UDF file set descriptor terminator logical block address

2.4.234 UDF_FIRST_ANCHOR_LBA Macro

C++

```
#define UDF_FIRST_ANCHOR_LBA 256
```

File

StarBurn.h (see page 662)

Description

First anchor volume descriptor pointer LBA

2.4.235 UDF_HEAD_SIZE_IN_LOGICAL_BLOCKS Macro

C++

```
#define UDF_HEAD_SIZE_IN_LOGICAL_BLOCKS ( UDF_FILE_SET_DESCRIPTOR_TERMINATOR_LBA + 1 )
```

File

StarBurn.h (see page 662)

Description

UDF head size in logical blocks

2.4.236 UDF_ISO9660_FILE_SYSTEM_LBA Macro

C++

```
#define UDF_ISO9660_FILE_SYSTEM_LBA 257
```

File

StarBurn.h (see page 662)

Description

UDF/ISO9660 bridge file system logical block address

2.4.237 UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS Macro

C++

```
#define UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS 0
```

File

StarBurn.h (see page 662)

Description

define UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS 6

2.4.238 UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS Macro

C++

```
#define UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS 2048
```

File

StarBurn.h (see page 662)

Description

UDF logical block size in UCHARs

2.4.239 UDF_NAME_SIZE_IN_UCHARS Macro

C++

```
#define UDF_NAME_SIZE_IN_UCHARS 255
```

File

StarBurn.h (see page 662)

Description

UDF longest name size in UCHARs

2.4.240 UDF_NAME_SIZE_IN_UCHARS_ACTUAL Macro

C++

```
#define UDF_NAME_SIZE_IN_UCHARS_ACTUAL 127
```

File

StarBurn.h (see page 662)

Description

This is macro UDF_NAME_SIZE_IN_UCHARS_ACTUAL.

2.4.241 UDF_NSR_DESCRIPTOR_LBA Macro

C++

```
#define UDF_NSR_DESCRIPTOR_LBA 17
```

File

StarBurn.h (see page 662)

Description

UDF NSR descriptor pointer LBA

2.4.242 UDF_TAIL_SIZE_IN_LOGICAL_BLOCKS Macro

C++

```
#define UDF_TAIL_SIZE_IN_LOGICAL_BLOCKS 256
```

File

StarBurn.h (see page 662)

Description

UDF tail size in logical blocks

2.4.243 WAVE_FILE_ALIGNMENT Macro

C++

```
#define WAVE_FILE_ALIGNMENT 4
```

File

StarBurn.h (see page 662)

Description

WAVE file data alignment (4 UCHARs, 1 ULONG)

2.4.244 WAVE_FILE_BITS_PER_SAMPLE Macro

C++

```
#define WAVE_FILE_BITS_PER_SAMPLE 16
```

File

StarBurn.h (see page 662)

Description

WAVE file bits per sample

2.4.245 WAVE_FILE_CHANNELS Macro

C++

```
#define WAVE_FILE_CHANNELS 2
```

File

StarBurn.h (see page 662)

Description

WAVE file number of channels

2.4.246 WAVE_FILE_DATA_SIGNATURE Macro

C++

```
#define WAVE_FILE_DATA_SIGNATURE ( ULONG )( 'atad' )
```

File

StarBurn.h (see page 662)

Description

WAVE file DATA signature

2.4.247 WAVE_FILE_RIFF_SIGNATURE Macro

C++

```
#define WAVE_FILE_RIFF_SIGNATURE ( ULONG )( 'FFIR' )
```

File

StarBurn.h (see page 662)

Description

WAVE file RIFF signature

2.4.248 WAVE_FILE_SAMPLES_PER_SECOND Macro

C++

```
#define WAVE_FILE_SAMPLES_PER_SECOND 44100
```

File

StarBurn.h (see page 662)

Description

WAVE file samples per second

2.4.249 WAVE_FILE_TAG_SIGNATURE Macro

C++

```
#define WAVE_FILE_TAG_SIGNATURE ( ULONG )( ' tmf' )
```

File

StarBurn.h (see page 662)

Description

WAVE file TAG signature

2.4.250 WAVE_FILE_WAVE_FORMAT Macro

C++

```
#define WAVE_FILE_WAVE_FORMAT 1
```

File

StarBurn.h (see page 662)

Description

WAVE file WAVE format

2.4.251 WAVE_FILE_WAVE_SIGNATURE Macro

C++

```
#define WAVE_FILE_WAVE_SIGNATURE ( ULONG )( 'EVAW' )
```

File

StarBurn.h (see page 662)

Description

WAVE file WAVE signature

2.4.252 WRITE_REPORT_DELAY_IN_SECONDS Macro

C++

```
#define WRITE_REPORT_DELAY_IN_SECONDS 1
```

File

StarBurn.h (see page 662)

Description

Default write report delay in seconds

2.5 Files

The following table lists files in this documentation.

Files

Name	Description
StarBurn.h (see page 662)	This is file StarBurn.h.

2.5.1 StarBurn.h

This is file StarBurn.h.

Enumerations

	Name	Description
	_CALLBACK_NUMBER (see page 424)	Enum that represents callback number
	_DISC_TYPE (see page 432)	Enum that represents inserted disc type
	_ERASE_TYPE (see page 434)	Enum that represents erase type
	_EXCEPTION_NUMBER (see page 435)	Enum that represents exception number
	_FILE_TIME (see page 438)	Enum that represents file time that will be included in the ISO9660/Joliet image
	_FILE_TREE (see page 438)	Enum that represents file tree
	_ISO9660_KIDTYPE (see page 441)	Enum that represent type of ISO9660 tree node
	_NAME_COLLISION_SOLUTION (see page 442)	Enum that represents the solutions of name collisions
	_READ_MODE (see page 444)	Enum that represents raw read modes
	_STARBURN_CD_MODE (see page 449)	Enum that represents CD modes type

	_STARBURN_ELTORITO_MEDIA (see page 452)	Media emulation types
	_STARBURN_ELTORITO_PLATFORM (see page 452)	ElTorito platform types
	_STARBURN_STARWAVE_CALLBACK_REASON (see page 454)	StarWave callback reasons we'll be using
	_STARBURN_STARWAVE_COMPRESSION (see page 454)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	_STARBURN_STARWAVE2_COMPRESS_TYPE (see page 456)	Compress settings
	_STARBURN_STARWAVE2_CONVERSION_MODE (see page 456)	Structure that represents conversion mode
	_STARBURN_STARWAVE2_QUALITY_MODE (see page 458)	Enum that represents MP3 quality for conversion
	_STARBURN_UDF_BRIDGE_TYPE (see page 461)	Restore original structure packing
	_STARBURN_UDF2_FILE_DATE_TIME_TYPE (see page 462)	Enum that represents StarBurn date/time type for files in UDF
	_STARPORT_DEVICE_TYPE (see page 464)	Enum that represents StarPort device type
	_STARWAVE2_COMPRESSION (see page 465)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	_UDF_OS_CLASS (see page 472)	UDF OS class
	_UDF_VERSION (see page 474)	Enum that represents UDF version
	_WRITE_MODE (see page 476)	Enum that represents write modes
	CALLBACK_NUMBER (see page 476)	Enum that represents callback number
	DISC_TYPE (see page 484)	Enum that represents inserted disc type
	ERASE_TYPE (see page 486)	Enum that represents erase type
	EXCEPTION_NUMBER (see page 487)	Enum that represents exception number
	FILE_TIME (see page 490)	Enum that represents file time that will be included in the ISO9660/Joliet image
	FILE_TREE (see page 490)	Enum that represents file tree
	ISO9660_KIDTYPE (see page 493)	Enum that represent type of ISO9660 tree node
	NAME_COLLISION_SOLUTION (see page 494)	Enum that represents the solutions of name collisions
	PCALLBACK_NUMBER (see page 495)	Enum that represents callback number
	PDISC_TYPE (see page 502)	Enum that represents inserted disc type
	PERASE_TYPE (see page 504)	Enum that represents erase type
	PEXCEPTION_NUMBER (see page 505)	Enum that represents exception number
	PFILE_TIME (see page 508)	Enum that represents file time that will be included in the ISO9660/Joliet image
	PFILE_TREE (see page 508)	Enum that represents file tree
	PISO9660_KIDTYPE (see page 511)	Enum that represent type of ISO9660 tree node
	PNAME_COLLISION_SOLUTION (see page 512)	Enum that represents the solutions of name collisions
	PPSTARBURN_STARWAVE_CALLBACK_REASON (see page 515)	StarWave callback reasons we'll be using
	PPSTARBURN_STARWAVE_COMPRESSION (see page 515)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PPSTARPORT_DEVICE_TYPE (see page 518)	Enum that represents StarPort device type
	PPSTARWAVE2_COMPRESSION (see page 518)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PREAD_MODE (see page 521)	Enum that represents raw read modes
	PSTARBURN_CD_MODE (see page 526)	Enum that represents CD modes type
	PSTARBURN_STARWAVE_CALLBACK_REASON (see page 529)	StarWave callback reasons we'll be using
	PSTARBURN_STARWAVE_COMPRESSION (see page 530)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PSTARBURN_STARWAVE2_CONVERSION_MODE (see page 531)	Structure that represents conversion mode
	PSTARBURN_STARWAVE2_QUALITY_MODE (see page 533)	Enum that represents MP3 quality for conversion
	PSTARPORT_DEVICE_TYPE (see page 537)	Enum that represents StarPort device type
	PSTARWAVE2_COMPRESSION (see page 538)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	PWRITE_MODE (see page 548)	Enum that represents write modes
	READ_MODE (see page 549)	Enum that represents raw read modes

	STARBURN_CD_MODE (see page 553)	Enum that represents CD modes type
	STARBURN_ELTORITO_MEDIA (see page 557)	Media emulation types
	STARBURN_ELTORITO_PLATFORM (see page 557)	ElTorito platform types
	STARBURN_STARWAVE_CALLBACK_REASON (see page 558)	StarWave callback reasons we'll be using
	STARBURN_STARWAVE_COMPRESSION (see page 559)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates, all of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	STARBURN_STARWAVE2_COMPRESS_TYPE (see page 560)	Compress settings
	STARBURN_STARWAVE2_CONVERSION_MODE (see page 561)	Structure that represents conversion mode
	STARBURN_STARWAVE2_QUALITY_MODE (see page 562)	Enum that represents MP3 quality for conversion
	STARBURN_UDF_BRIDGE_TYPE (see page 565)	Restore original structure packing
	STARBURN_UDF2_FILE_DATE_TIME_TYPE (see page 567)	Enum that represents StarBurn date/time type for files in UDF
	STARPORT_DEVICE_TYPE (see page 569)	Enum that represents StarPort device type
	STARWAVE2_COMPRESSION (see page 570)	Compression templates we'll be using, pointer to compression templates and pointer to pointer to compression templates All of the compression templates are 44 kHz, stereo, 16-bit, CBR or VBR
	UDF_OS_CLASS (see page 577)	UDF OS class
	UDF_VERSION (see page 579)	Enum that represents UDF version
	WRITE_MODE (see page 581)	Enum that represents write modes

Functions

	Name	Description
	StarBurn_CdvdBurnerGrabber_AuthorizeDVD (see page 17)	This function authorizes DVD that has CSS protection system.
	StarBurn_CdvdBurnerGrabber_AuthorizeDVDEx (see page 19)	This function authorizes DVD that has CSS protection system and keeps bus key for disc key.
	StarBurn_CdvdBurnerGrabber_Blank (see page 21)	This function erases the rewritable media inserted into CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_BluRayFormat (see page 23)	This function formats Blu-Ray rewritable media (BD-RE) inserted into the Blu-Ray burner device.
	StarBurn_CdvdBurnerGrabber_BluRayFormatQuery (see page 25)	This function queries available Blu-Ray formats for the currently inserted Blu-Ray rewritable media (BD-RE).
	StarBurn_CdvdBurnerGrabber_Cancel (see page 27)	This function cancels current read or write process on the CD/DVD/Blu-Ray/HD-DVD burner.
	StarBurn_CdvdBurnerGrabber_CloseSession (see page 29)	This function closes session on on CD/DVD/Blu-Ray/HD-DVD burner device after something was written using StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189), StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184) or StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172).
	StarBurn_CdvdBurnerGrabber_Create (see page 31)	This function creates CD/DVD/Blu-Ray/HD-DVD burner device object. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD releated actions.
	StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)	This function creates CD/DVD/Blu-Ray/HD-DVD burner device object using SPTI (SCSI Pass Through Interface) transport. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD related actions.
	StarBurn_CdvdBurnerGrabber_CreateExEx (see page 35)	This function creates CD/DVD/Blu-Ray/HD-DVD burner device object using SPTD (SCSI Pass Through Direct) transport. This object will be used later to perform CD/DVD/Blu-Ray/HD-DVD related actions.
	StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFile (see page 37)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be one sound file or full set of sound files if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once PQ mode.
	StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileUnicode (see page 39)	This is function StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileUnicode.
	StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromTree (see page 40)	This function records ISO9660 or Joliet file system image located in file tree object with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once PQ mode.

	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFile (see page 42)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be one sound file or full set of sound files if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once raw P-W mode.
	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileAudioUnicode (see page 45)	This API accepts track names in Unicode format and does AUDIO only DAO burn.
	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileUnicode (see page 46)	This is function StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileUnicode.
	StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromTree (see page 46)	This function records ISO9660 or Joliet file system image located in file tree object with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Disc-At-Once raw P-W mode.
	StarBurn_CdvdBurnerGrabber_Eject (see page 49)	This function ejects the media on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_ExecuteGeneric (see page 50)	This function executes custom CDB (Command Descriptor Block) on CD/DVD/Blu-Ray/HD-DVD burner device object. This could be used to deal with the propriatry hardware or execute some special code StarBurn does not support out-of-box.
	StarBurn_CdvdBurnerGrabber_GetAdvancedSupportedMediaFormats (see page 53)	This function returns CD/DVD/Blu-Ray/HD-DVD burner device object extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetBUP (see page 55)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device supports BUP (Buffer Underrun Protection) or not and what is BUP current status (if supported). This information can be used later to set BUP status before starting write operations.
	StarBurn_CdvdBurnerGrabber_GetDeviceInformation (see page 57)	This function returns CD/DVD/Blu-Ray/HD-DVD burner device object information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetDeviceInformationUnicode (see page 59)	This is function StarBurn_CdvdBurnerGrabber_GetDeviceInformationUnicode.
	StarBurn_CdvdBurnerGrabber_GetDiscFileSystem (see page 59)	This function detects recorded optical disc file system.
	StarBurn_CdvdBurnerGrabber_GetDiscFreeSpace (see page 60)	This function gets disc free space of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetDiscInformation (see page 62)	This function gets disc information of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetDiscUsedSpace (see page 64)	This function gets disc used space of the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetDVDProtectionSystem (see page 66)	This function gets DVD protection system of inserted DVD media into DVD device.
	StarBurn_CdvdBurnerGrabber_GetDVDRegionMask (see page 68)	This function gets DVD region mask from DVD media inserted to CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.
	StarBurn_CdvdBurnerGrabber_GetInsertedDiscType (see page 70)	This function returns inserted to CD/DVD/Blu-Ray/HD-DVD burner device disc type. This data can be used to report it to the user and to select the stream type to record.
	StarBurn_CdvdBurnerGrabber_GetLastAddress (see page 71)	This function gets last recorded address from CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object. Such an address could be used for already recorded session import with the UDF mastering.
	StarBurn_CdvdBurnerGrabber_GetLastTrack (see page 73)	This function gets last recorded track from CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object. Such a track number could be used for already recorded session import.
	StarBurn_CdvdBurnerGrabber_GetMechanicalOptions (see page 75)	This function gets mechanical options (is device capable of programmatically load/eject and lock/unlock etc) from CD/DVD/Blu-Ray/HD-DVD burner device object.
	StarBurn_CdvdBurnerGrabber_GetMediaTrayStatus (see page 77)	This function gets media (inserted or not) and tray (opened or closed) statuses from CD/DVD/Blu-Ray/HD-DVD burner device object.
	StarBurn_CdvdBurnerGrabber_GetNumberOfSystemDescriptors (see page 79)	This function gets number of system descriptors in system structures stream. This value is used for updating head of the image when creating signle track session import code.

	StarBurn_CdvdBurnerGrabber_GetRPC (see page 81)	This function gets RPC (region play code) and some additional DVD region management information from "DVD part" of the CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.
	StarBurn_CdvdBurnerGrabber_GetSpeeds (see page 83)	This function gets current read and write and top supported read and write speeds on CD/DVD/Blu-Ray/HD-DVD burner device. This speeds later will be used to set current speeds during all read and write operations.
	StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormats (see page 85)	This function returns CD/DVD/Blu-Ray/HD-DVD burner device object extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsEx (see page 88)	This function returns DVD+R(W) burner device object extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetSupportedMediaFormatsExEx (see page 90)	This function returns DVD+R(W) burner device object extended extended information. This data can be used to report the capabilities of the device to the user.
	StarBurn_CdvdBurnerGrabber_GetTOCInformation (see page 92)	This function gets TOC information of the currently inserted disc in the CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetTrackInformation (see page 94)	This function gets track information of the asked track number on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GetTrackInformationEx (see page 96)	This function gets track extended information of the asked track number on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GrabCD (see page 98)	This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format.
	StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)	This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files.
	StarBurn_CdvdBurnerGrabber_GrabDVDEx (see page 102)	This function grabs disc image (extended) from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files. StarBurn_CdvdBurnerGrabber_GrabDVDEx function for all I/O files operations use user's callback functions.
	StarBurn_CdvdBurnerGrabber_GrabDVDFast (see page 105)	This function grabs disc image from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device and store it in MDS format, disc image will be stored into the series of files. Works faster than StarBurn_CdvdBurnerGrabber_GrabDVD (see page 100)
	StarBurn_CdvdBurnerGrabber_GrabRange (see page 108)	This function grabs range of logical blocks from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GrabTrack (see page 110)	This function grabs track from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_GrabTrackEx (see page 112)	This function grabs track (extended) from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device. StarBurn_CdvdBurnerGrabber_GrabTrackEx function for all I/O files operations use user's callback functions.
	StarBurn_CdvdBurnerGrabber_GrabTrackUnicode (see page 115)	This is function StarBurn_CdvdBurnerGrabber_GrabTrackUnicode.
	StarBurn_CdvdBurnerGrabber_IsDiscBlank (see page 115)	This function detect blank state of currently inserted disc in CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_Load (see page 117)	This function loads the media on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_LoadEx (see page 119)	This function loads the media on CD/DVD/Blu-Ray/HD-DVD burner device. Unlike StarBurn_CdvdBurnerGrabber_Load (see page 117)(...) this API call allows to control should it return immediately or wait until the drive would become into ready state.
	StarBurn_CdvdBurnerGrabber_Lock (see page 121)	This function locks the media inside the CD/DVD/Blu-Ray/HD-DVD device.
	StarBurn_CdvdBurnerGrabber_MSFToLBA (see page 123)	This function converts MSF (minute:second:frame) address into the LBA (logical block address).
	StarBurn_CdvdBurnerGrabber_ProbeSupportedReadModes (see page 123)	This function probes supported read modes on the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_ProbeSupportedWriteModes (see page 126)	This function probes supported read modes on the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.

	StarBurn_CdvdBurnerGrabber_ReadATIP (see page 128)	This function reads ATIP information from the media currently inserted into CD/DVD/Blu-Ray/HD-DVD burner device object created with the call to the one of the StarBurn_CdvdBurnerGrabber_Create (see page 31) or StarBurn_CdvdBurnerGrabber_CreateEx (see page 33) API calls.
	StarBurn_CdvdBurnerGrabber_ReadCooked (see page 130)	This function reads one or more logical block(s) from the currently inserted CD/DVD/Blu-Ray/HD-DVD in cooked format.
	StarBurn_CdvdBurnerGrabber_ReadLB (see page 132)	This function reads one logical block from the currently inserted disc on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_ReadRaw (see page 134)	This function reads one or more logical block(s) from the currently inserted CD in RAW format.
	StarBurn_CdvdBurnerGrabber_Release (see page 137)	This function releases the media inside the CD/DVD/Blu-Ray/HD-DVD device after the media was locked before.
	StarBurn_CdvdBurnerGrabber_SendOPC (see page 139)	This function sends OPC (Optimum Power Calibration) for current write speed on CD/DVD/Blu-Ray/HD-DVD burner device and current CD/DVD/Blu-Ray/HD-DVD media. The set laser level will be used later during all write operations.
	StarBurn_CdvdBurnerGrabber_SessionAtOnce (see page 141)	This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) or ISO image with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Session-At-Once mode.
	StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPW (see page 143)	This function records raw image with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in raw Session-At-Once mode.
	StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWUnicode (see page 146)	This is function StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWUnicode.
	StarBurn_CdvdBurnerGrabber_SessionAtOnceUnicode (see page 146)	This is function StarBurn_CdvdBurnerGrabber_SessionAtOnceUnicode.
	StarBurn_CdvdBurnerGrabber_SetBUP (see page 147)	This function sets BUP (Buffer Underrun Protection) on CD/DVD/Blu-Ray/HD-DVD burner device that supports BUP. BUP must be enabled for successful completion of write operations.
	StarBurn_CdvdBurnerGrabber_SetCDTextItem (see page 149)	Sets CD-Text item for the specified CD/DVD/Blu-Ray/HD-DVD device.
	StarBurn_CdvdBurnerGrabber_SetReadSpeed (see page 151)	This function sets read speed for CD/DVD/Blu-Ray/HD-DVD media currently inserted to CD/DVD/Blu-Ray/HD-DVD burner device object.
	StarBurn_CdvdBurnerGrabber_SetSpeeds (see page 153)	This function sets current read and write speeds on CD/DVD/Blu-Ray/HD-DVD burner device This speeds later will be used during all read and write operations.
	StarBurn_CdvdBurnerGrabber_StopPlayScan (see page 155)	This function resets media state inside CD/DVD/Blu-Ray/HD-DVD burner device object. This actions allow to update disc content w/o need to eject the disc. Could be used for multiple burnings w/o disc eject or manual disc data verification.
	StarBurn_CdvdBurnerGrabber_SuperVideoCD (see page 157)	This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDEx (see page 159)	This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDExEx (see page 162)	This function records SVCD/MPEG2 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDExExUnicode (see page 164)	This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDExExUnicode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDExUnicode (see page 165)	This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDExUnicode.
	StarBurn_CdvdBurnerGrabber_SuperVideoCDUnicode (see page 165)	This is function StarBurn_CdvdBurnerGrabber_SuperVideoCDUnicode.
	StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations.

	StarBurn_CdvdBurnerGrabber_TestUnitReadyEx (see page 168)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations. The only difference between this and simple StarBurn_CdvdBurnerGrabber_TestUnitReady (see page 166)() is that this code passed parameter to wait for unit to become ready. And "simple" code uses default timeout of 20 seconds.
	StarBurn_CdvdBurnerGrabber_TestUnitReadyExEx (see page 170)	This function tests is CD/DVD/Blu-Ray/HD-DVD burner device object unit ready or not. This information can be used later to start I/O operations. This "double extended" variant of the call also can wait for particular amount of seconds and do "fast exit" if there's no disc in drive.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFile (see page 172)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be sound file if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicode (see page 174)	This API call deals with audio tracks with Unicode names only.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileAudioUnicodeEx (see page 175)	This API call deals with audio tracks with Unicode names only. Also it allows to set track-to-track pause length.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileEx (see page 176)	This function records ISO9660 or Joliet file system image located in a file on the hard disk (also it can be sound file if an audio CD is mastered) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileUnicode (see page 178)	This is function StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileUnicode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemory (see page 179)	This function records ISO9660 or Joliet file system image or audio stream located in a memory with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromMemoryEx (see page 181)	This function records ISO9660 or Joliet file system image or audio stream located in a memory with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode. If audio content is recorded this API call can change default Track-At-Once track-to-track pause length from 150 logical blocks (2 seconds) to any passed one.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipe (see page 184)	This function records ISO9660 or Joliet file system image located in a pipe with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromPipeEx (see page 186)	This function records ISO9660 or Joliet file system image or audio stream located in a pipe with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTree (see page 189)	This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_TrackAtOnceFromTreeEx (see page 191)	This function records ISO9660 or Joliet file system image located in a virtual file tree created with StarBurn_ISO9660JolietFileTree_Create (see page 288) with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192)	This function parses the UDF file system on disc and calls a callback function for each found file. The information about LBAs that file resides on is passed to callback function.
	StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193)	This function parses the UDF file system on disc and calls a callback function for each found file and directory. The information about LBAs that file resides on is passed to callback function.
	StarBurn_CdvdBurnerGrabber_VerifyFile (see page 194)	This function verifies recorded file system image on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular image recording (or maybe other image). This information can be used to be sure recorded disc is readable and really contains recorded information.
	StarBurn_CdvdBurnerGrabber_VerifyFileEx (see page 196)	This function verifies recorded file system image on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular image recording (or maybe other image). This information can be used to be sure recorded disc is readable and really contains recorded information.

	StarBurn_CdvdBurnerGrabber_VerifyFileExUnicode (see page 198)	This is function StarBurn_CdvdBurnerGrabber_VerifyFileExUnicode.
	StarBurn_CdvdBurnerGrabber_VerifyFileUnicode (see page 198)	This is function StarBurn_CdvdBurnerGrabber_VerifyFileUnicode.
	StarBurn_CdvdBurnerGrabber_VerifyTree (see page 199)	This function verifies recorded file system tree on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular tree recording. This information can be used to be sure recorded disc is readable and really contains recorded information.
	StarBurn_CdvdBurnerGrabber_VerifyTreeEx (see page 201)	This function verifies recorded file system tree on CD/DVD/Blu-Ray/HD-DVD burner device object used for this particular tree recording. This information can be used to be sure recorded disc is readable and really contains recorded information.
	StarBurn_CdvdBurnerGrabber_VideoCD (see page 203)	This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in Track-At-Once mode.
	StarBurn_CdvdBurnerGrabber_VideoCDEx (see page 205)	This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_VideoCDExEx (see page 208)	This function records VCD/MPEG1 image located in a file on the hard disk with current write speed on CD/DVD/Blu-Ray/HD-DVD burner device in selected write mode.
	StarBurn_CdvdBurnerGrabber_VideoCDExExUnicode (see page 210)	This is function StarBurn_CdvdBurnerGrabber_VideoCDExExUnicode.
	StarBurn_CdvdBurnerGrabber_VideoCDExUnicode (see page 211)	This is function StarBurn_CdvdBurnerGrabber_VideoCDExUnicode.
	StarBurn_CdvdBurnerGrabber_VideoCDUnicode (see page 211)	This is function StarBurn_CdvdBurnerGrabber_VideoCDUnicode.
	StarBurn_CorrectISO9660Name_Default (see page 212)	This function generates next ISO9660 file name to make it unique.
	StarBurn_CorrectJolietName_Default (see page 212)	This function generates next Joliet file name to make it unique.
	StarBurn_CreatePipe (see page 213)	This function creates pipe for all of the StarBurn pipe operations and returns both "read" and "write" handles for it.
	StarBurn_Destroy (see page 214)	This function frees allocated memory in the way of destroying the object that toolkit created internally. This is universal call, it frees all the objects and does not care about object type.
	StarBurn_DestroyPipe (see page 215)	This function closes both "read" and "write" pipe handles.
	StarBurn_DownShut (see page 216)	This function uninitialize burning toolkit. It's expected to be called after very last function call before exiting from StarBurn client application. It's a good idea to gather the trash before exiting. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used.
	StarBurn_DVDVideo_Create (see page 216)	This function creates DVD-Video file system object from passed pointer to VIDEO_TS directory with DVD corresponding files.
	StarBurn_DVDVideo_CreateUnicode (see page 218)	This is function StarBurn_DVDVideo_CreateUnicode.
	StarBurn_DVDVideo_Destroy (see page 218)	This function destroys DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.
	StarBurn_DVDVideo_GetSizeInUCHARs (see page 219)	This function gets file system content size in UCHARs from DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.
	StarBurn_DVDVideo_GetTreePointer (see page 219)	This function returns pointer to ISO9660 file tree object embedded to DVD-Video file system object created with StarBurn_DVDVideo_Create (see page 216) API call.
	StarBurn_DVDVideo_PatchHeader (see page 220)	This function patches IFO/BUF file header RBA for BUP.
	StarBurn_DVDVideo_PatchTable (see page 221)	This function patches IFO/BUF file table RBAs.
	StarBurn_DVDVideo_Read (see page 221)	This function reads requested number of UCHARs from DVD-Video file system object (created with StarBurn_DVDVideo_Create (see page 216) API call) from it's current read position.
	StarBurn_DVDVideo_SeekToBegin (see page 222)	This function moves DVD-Video file system object (created with StarBurn_DVDVideo_Create (see page 216) API call) current read position to the very beginning.
	StarBurn_ElTorito_BootCatalogAddSection (see page 223)	This function creates ElTorito boot catalog (ANSI names).

	StarBurn_ElTorito_BootCatalogAddSectionUnicode (see page 223)	This function creates ElTorito boot catalog (UNICODE names).
	StarBurn_ElTorito_CreateBootCatalog (see page 224)	This function creates ElTorito boot catalog (ANSI names).
	StarBurn_ElTorito_CreateBootCatalogUnicode (see page 225)	This function creates ElTorito boot catalog (UNICODE names).
	StarBurn_FileClose (see page 225)	This fucntion closes file handle embedded to file object.
	StarBurn_FileCreate (see page 226)	This function creates new file from the passed file path and name.
	StarBurn_FileInitialize (see page 226)	This function initializes file handle embedded to file object with bad file handle value.
	StarBurn_FileOpen (see page 226)	This function opens existing file from the passed file path and name.
	StarBurn_FileOpenEx (see page 227)	This function opens existing file from the passed file path and name.
	StarBurn_FileRead (see page 227)	This function reads asked amount of unsigned chars from file at current seek position.
	StarBurn_FileReWind (see page 228)	This function rewinds file position in unsigned chars to the very beginning for the file.
	StarBurn_FileSeek (see page 228)	This function sets file position in unsigned chars to required one.
	StarBurn_FileSeekRead (see page 229)	This function sets file position in unsigned chars to required one and reads requested amounts unsigned chars.
	StarBurn_FileSizeInUCHARsGet (see page 229)	This function gets file size in unsigned chars.
	StarBurn_FileUnicodeCreate (see page 230)	This function creates new file from the passed UNICODE file path and name.
	StarBurn_FileUnicodeOpen (see page 230)	This function opens existing file from the passed UNICODE file path and name.
	StarBurn_FileUnicodeOpenEx (see page 230)	This function opens existing file from the passed UNICODE file path and name.
	StarBurn_FileWrite (see page 231)	This function writes asked amount of unsigned chars to file at current seek position
	StarBurn_FindDevice (see page 231)	This function finds device of the specified device type.
	StarBurn_GetAudioFileStreamSizeInUCHARs (see page 233)	This function returns audio stream size in UCHARs for passed supported as source audio file.
	StarBurn_GetAudioFileStreamSizeInUCHARs_Fast (see page 234)	This function returns audio stream size in UCHARs for passed supported as source audio file.
	StarBurn_GetAudioFileStreamSizeInUCHARs_UnicodeFast (see page 235)	This is function StarBurn_GetAudioFileStreamSizeInUCHARs_UnicodeFast.
	StarBurn_GetAudioFileStreamSizeInUCHARsUnicode (see page 236)	This is function StarBurn_GetAudioFileStreamSizeInUCHARsUnicode.
	StarBurn_GetBufferUnderrunTimeOutInMs (see page 236)	This function returns buffer underrun timeout in milliseconds. It's amount of time StarBurn core would wait before resubmitting each command to the drive if burning was faster then reading and StarBurn had exhausted software cache it uses for buffering. Modern hardware has BUP (Buffer Underrun Protection) and would perfectly survive in such a condition and keep burning disc still usable.
	StarBurn_GetDeviceLetter (see page 237)	This function returns device letter (device root path) for a device name, like the one used in StarBurn_CdvdBurnerGrabber_CreateEx (see page 33)(...) API call.
	StarBurn_GetDeviceLetterUnicode (see page 237)	This is function StarBurn_GetDeviceLetterUnicode.
	StarBurn_GetDeviceNameByDeviceAddress (see page 238)	This function gets device name by device address.
	StarBurn_GetDeviceTimeOutByDeviceAddress (see page 239)	This function gets device timeout in seconds by device SCSI address.
	StarBurn_GetDVDPadding (see page 239)	This function returns is DVD padding mode (when at least 1GB of the data would be recorded to DVD media - required for DVD-Video compilations to work properly on standalone DVD players) enabled (TRUE) or disabled (FALSE).
	StarBurn_GetDVDPLUSRDLCompatibleMode (see page 240)	This function returns is DVD+R DL (Dual Layer) so-called "compatible mode" (when layer would be switched exactly at 1/2 of the data payload recorded on the disc - required for DVD-Video compilations to work properly) enabled (TRUE) or disabled (FALSE).
	StarBurn_GetEjectAfterFail (see page 241)	This function returns is so-called "eject after fail" (if during burning process error would happen - should StarBurn eject disc or not) enabled (TRUE) or disabled (FALSE).

	StarBurn_GetFastReadTOC (see page 241)	This function returns is so-called "fast read TOC" (TOC information is not 100% accurate - to get EXACT track length StarBurn trys to read beginning and end of the track to find sub-channel index switch indicating EXACT track beginning or EXACT track end) mode enabled (TRUE) or disabled (FALSE).
	StarBurn_GetId (see page 242)	This function gets ASPI layer ID strings.
	StarBurn_GetIs64KBIO (see page 242)	This function returns is so-called "64KB I/O" (when StarBurn issues 64KB I/O packets instead of the 32KB ones) currently enabled (TRUE) or disabled (FALSE).
	StarBurn_GetIsCollisionDetectionDisabled (see page 243)	This function returns is collision detection disabled (TRUE) or enabled (FALSE). If collision detection is enabled - every new file added to ISO9660 or Joliet file tree would be checked to have unique name. If such a file already exist - special callback (collision detection one) would be called so user would be able to either replace or rename new or old file. However if collision detection is disabled - no comparison and no actions would be performed. File would be just added to the destination file system image AS IS, having it's original name.
	StarBurn_GetIsSafeGrabbingEnabled (see page 243)	This function returns is so-called "safe grabbing" (when StarBurn would mix read commands with checking for device to become ready - required to workaround broken ATAPI-to-USB bridges) enabled (TRUE) or disabled (FALSE).
	StarBurn_GetVersion (see page 244)	This function returns a 32-bit unsigned long value that contains packed numbers that represent the year, the month and the date of the toolkit build. So 10th of March, year of 1978 will look like 0x19780310.
	StarBurn_GetVolumeIDs (see page 245)	This function gets volume ID from ISO9660/Joliet volume name.
	StarBurn_ImageFile_UDFFileSystemLookup (see page 245)	This function parses the UDF file system inside image file and calls a callback function for each found file. The information about LBAs that file resides on is passed to callback function.
	StarBurn_ImageFile_UDFFileSystemLookupEx (see page 246)	This function parses the UDF file system inside image file and calls a callback function for each found file and directory. The information about LBAs that file resides on is passed to callback function.
	StarBurn_IsAudioFileSupported (see page 246)	This function returns is currently passed file supported as source audio file.
	StarBurn_IsAudioFileSupportedUnicode (see page 247)	This is function StarBurn_IsAudioFileSupportedUnicode.
	StarBurn_IsLocalDateTimeUsed (see page 248)	This is function StarBurn_IsLocalDateTimeUsed.
	StarBurn_ISO2_Check1999Name (see page 248)	This function checks if given string valid ISO9660:1999 file (or directory) name
	StarBurn_ISO2_CheckJolietName (see page 248)	This function checks if given string valid ISO9660 Joliet file (or directory) name
	StarBurn_ISO2_CheckLevel1Name (see page 249)	This function checks if given string valid ISO9660 Level 1 file (or directory) name
	StarBurn_ISO2_CheckLevel2Name (see page 249)	This function checks if given string valid ISO9660 Level 2 file (or directory) name
	StarBurn_ISO2_CheckRelaxedJolietName (see page 250)	This function checks if given string valid ISO9660 Joliet file (or directory) name with less restrictions: names may contain more than one dot.
	StarBurn_ISO2_CreateNewName (see page 250)	This function creates new ANSI name by adding ~XX (where XX is a number (Seed)) at the end of file name.
	StarBurn_ISO2_DirectoryBrowse (see page 250)	This function browses ISO9660 directory content.
	StarBurn_ISO2_DirectoryCreate (see page 251)	This function creates ISO9660 directory.
	StarBurn_ISO2_DirectoryDestroy (see page 251)	This function destroys created ISO9660 directory object.
	StarBurn_ISO2_DirectoryProcess (see page 252)	This function processes single ISO9660 directory.
	StarBurn_ISO2_DirectoryQuery (see page 252)	This function do query for ISO9660 file properties.
	StarBurn_ISO2_DirectoryRootCreate (see page 253)	This function creates root ISO9660 directory.
	StarBurn_ISO2_DirectoryUnicodeCreate (see page 253)	This function creates UNICODE ISO9660 directory.
	StarBurn_ISO2_DirectoryUnicodeProcess (see page 254)	This function processes single UNICODE ISO9660 directory.
	StarBurn_ISO2_FileCreate (see page 255)	This function creates ISO9660 file.
	StarBurn_ISO2_FileDestroy (see page 255)	This function destroys created ISO9660 file object.
	StarBurn_ISO2_FileDirectoryDateTimeGet (see page 256)	This function gets date and time from created ISO9660 file or directory object.
	StarBurn_ISO2_FileDirectoryDateTimeSet (see page 256)	This function sets date and time on created ISO9660 file or directory object.

	StarBurn_ISO2_FileDirectoryNameSet (see page 256)	This function set new name for ISO9660 file or directory object.
	StarBurn_ISO2_FileDirectoryUnicodeNameSet (see page 257)	This function set new UNICODE name for ISO9660 file or directory object.
	StarBurn_ISO2_FileMemoryCreate (see page 257)	This function creates ISO9660 memory file.
	StarBurn_ISO2_FileMemoryUnicodeCreate (see page 258)	This function creates Unicode ISO9660 memory file.
	StarBurn_ISO2_FileQuery (see page 258)	This function do query for ISO9660 file properties.
	StarBurn_ISO2_FileSetAttributes (see page 259)	This function sets attributes for ISO9660 file.
	StarBurn_ISO2_FileUnicodeCreate (see page 259)	This function creates UNICODE ISO9660 file.
	StarBurn_ISO2_ImpVolumeCreate (see page 260)	This function processes contents of single UNICODE ISO9660 directory. __stdcall StarBurn_ISO2_DirectoryContentsUnicodeProcess(const unsigned short *UnicodePathName, unsigned long *GUID, void *Root, PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK (see page 587) Callback, const void *Context, STARBURN_ISO2_FILE_DATE_TIME *DateTime); This function creates ISO9660 import volume.
	StarBurn_ISO2_ImpVolumeDestroy (see page 261)	This function destroys created ISO9660 import volume object.
	StarBurn_ISO2_ImpVolumeImport (see page 261)	This function imports ISO9660 import volume content.
	StarBurn_ISO2_ISO9660FileTreeCreate (see page 262)	This function create ISO image from already created ISO9660 volume.
	StarBurn_ISO2_VolumeCreate (see page 262)	This function creates ISO9660 volume.
	StarBurn_ISO2_VolumeCreateBoot (see page 263)	This function creates ISO9660 bootable volume (ANSI names).
	StarBurn_ISO2_VolumeCreateBootElTorito (see page 264)	This function creates ISO9660 bootable volume (ANSI names).
	StarBurn_ISO2_VolumeCreateEx (see page 265)	This function creates ISO9660 volume (with both Unicode and ANSI names).
	StarBurn_ISO2_VolumeCreateExBoot (see page 266)	This function creates ISO9660 bootable volume (with both Unicode and ANSI names).
	StarBurn_ISO2_VolumeCreateUnicode (see page 267)	This function creates ISO9660 volume.
	StarBurn_ISO2_VolumeCreateUnicodeBoot (see page 268)	This function creates ISO9660 bootable volume (Unicode names).
	StarBurn_ISO2_VolumeCreateUnicodeBootElTorito (see page 269)	This function creates ISO9660 bootable volume (Unicode names).
	StarBurn_ISO2_VolumeDescriptorsBufferGet (see page 270)	This function get ISO9660 volume descriptors buffer.
	StarBurn_ISO2_VolumeDestroy (see page 270)	This function destroys created ISO9660 volume object.
	StarBurn_ISO2_VolumeSeekRead (see page 271)	This function seeks to passed offset and reads requested amount of data.
	StarBurn_ISO2_VolumeSizeInUCHARsGet (see page 271)	This function get ISO9660 volume size in unsigned chars.
	StarBurn_ISO9660JolietFileTree_Add (see page 272)	This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.
	StarBurn_ISO9660JolietFileTree_AddEx (see page 274)	This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.
	StarBurn_ISO9660JolietFileTree_AddMemory (see page 277)	This function adds the node created from the memory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.

	StarBurn_ISO9660JolietFileTree_AddW (see page 279)	This function adds the tree created from the directory to already created ISO9660 or Joliet file tree from passed directory. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations. Unlike other tree management calls this one deals with Unicode names (wide char variant of basic call).
	StarBurn_ISO9660JolietFileTree_AddZeroFile (see page 282)	This function adds the new "filled with zeros" file to already created ISO9660 or Joliet file tree. Later this tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be burn directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate operations.
	StarBurn_ISO9660JolietFileTree_BuildImage (see page 283)	This function builds ISO9660 or Joliet file system image from ISO9660 or Joliet file tree (assign all the logical block offsets and allocates and initializes all file system internal structures). Later this tree can be used to store the "virtual" ISO9660 or Joliet image to the file on the disk or to burn this ISO9660 or Joliet image directly to the CD/DVD/Blu-Ray/HD-DVD media.
	StarBurn_ISO9660JolietFileTree_BuildImageEx (see page 285)	This function builds ISO9660 or Joliet file system image from ISO9660 or Joliet file tree (assign all the logical block offsets and allocates and initializes all file system internal structures). Later this tree can be used to store the "virtual" ISO9660 or Joliet image to the file on the disk or to burn this ISO9660 or Joliet image directly to the CD/DVD/Blu-Ray/HD-DVD media. It's identical to StarBurn_ISO9660JolietFileTree_BuildImage (see page 283) except has some extra parameter to put into file system descriptor.
	StarBurn_ISO9660JolietFileTree_Cancel (see page 287)	This function cancel ISO9660 or Joliet file tree creation process.
	StarBurn_ISO9660JolietFileTree_Create (see page 288)	This function allocates in the memory and initializes ISO9660 or Joliet file tree. Later this file tree can be used to build "virtual" ISO96600 or Joliet file system image. Resulting "virtual" image can be either stored in the file on the disk or be recorder directly to the CD/DVD/Blu-Ray/HD-DVD media w/o any other intermediate buffering operations on the hard disk.
	StarBurn_ISO9660JolietFileTree_GetAttributes (see page 290)	This function returns ISO9660 or Joliet file tree node attributes.
	StarBurn_ISO9660JolietFileTree_GetAutoSorting (see page 292)	This function returns ISO9660 or Joliet file tree node sorting mode.
	StarBurn_ISO9660JolietFileTree_GetFileSizeInUCHARs (see page 292)	This function returns ISO9660 or Joliet file tree file size in UCHARs.
	StarBurn_ISO9660JolietFileTree_GetFirstKid (see page 294)	This function returns ISO9660 or Joliet file tree node first kid pointer.
	StarBurn_ISO9660JolietFileTree_GetFullPath (see page 296)	This function returns full path to ISO9660 or Joliet file tree node.
	StarBurn_ISO9660JolietFileTree_GetKidType (see page 297)	This function returns type of ISO9660 or Joliet file tree node.
	StarBurn_ISO9660JolietFileTree_GetLevel (see page 297)	This function returns ISO9660 or Joliet file tree level. The number of levels in the created tree.
	StarBurn_ISO9660JolietFileTree_GetNames (see page 299)	This function returns ISO9660 or Joliet file tree node names (long, short and absolute).
	StarBurn_ISO9660JolietFileTree_GetNamesEx (see page 300)	This function returns ISO9660 or Joliet file tree node names (long, short and absolute). Also it returns file system dependent names in ANSI and Unicode.
	StarBurn_ISO9660JolietFileTree_GetNamesW (see page 302)	StarBurn_ISO9660JolietFileTree_GetNamesW function is not documented
	StarBurn_ISO9660JolietFileTree_GetNextKid (see page 303)	This function returns ISO9660 or Joliet file tree root node next kid pointer.
	StarBurn_ISO9660JolietFileTree_GetNodeISO9660DateTime (see page 305)	This function returns ISO9660 or Joliet file tree node ISO9660 time stamp (date and time).
	StarBurn_ISO9660JolietFileTree_GetNodePowerInUCHARs (see page 306)	This function returns ISO9660 or Joliet file tree node power (file size) in UCHARs.
	StarBurn_ISO9660JolietFileTree_GetNodeStartingLBA (see page 307)	This function returns ISO9660 or Joliet file tree node starting LBA after the image was built.
	StarBurn_ISO9660JolietFileTree_GetNodeSystemTime (see page 308)	This function returns ISO9660 or Joliet file tree node SYSTEMTIME.
	StarBurn_ISO9660JolietFileTree_GetParent (see page 309)	This function returns ISO9660 or Joliet file tree node parent pointer.

	StarBurn_ISO9660JolietFileTree_GetRoot (see page 311)	This function returns ISO9660 or Joliet file tree root node pointer.
	StarBurn_ISO9660JolietFileTree_GetSizeInUCHARs (see page 312)	This function returns ISO9660 or Joliet built file system image size in UCHARs.
	StarBurn_ISO9660JolietFileTree_ImportFile (see page 314)	This function imports file for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_ImportTrack (see page 316)	This function imports track for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_Read (see page 318)	This function reads ISO9660 or Joliet file system image from current offset into the user's data buffer. ISO9660 or Joliet file tree must have assigned all the logical block offsets and all file system internal structures allocated and initialized. The data that is read can be either stored in the file on the hard disk or directly written to CD/DVD/Blu-Ray/HD-DVD media.
	StarBurn_ISO9660JolietFileTree_Remove (see page 320)	This function deletes ISO9660 or Joliet file tree node and all it's kids.
	StarBurn_ISO9660JolietFileTree_SeekToBegin (see page 322)	This function seeks the ISO9660 or Joliet file system image from current offset to the very beginning so read operation will start from the 0 offset of the file system image. ISO9660 or Joliet file tree must have assigned all the logical block offsets and all file system internal structures allocated and initialized.
	StarBurn_ISO9660JolietFileTree_SetAttributes (see page 324)	This function sets ISO9660 or Joliet file tree node attributes.
	StarBurn_ISO9660JolietFileTree_SetAutoSorting (see page 325)	This function sets ISO9660 or Joliet file tree node auto sorting mode.
	StarBurn_ISO9660JolietFileTree_SetBootImage (see page 326)	This function sets boot image for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_SetBootImageEx (see page 328)	This extended function sets boot image for created ISO9660 or Joliet file tree.
	StarBurn_ISO9660JolietFileTree_SetNames (see page 330)	This function sets ISO9660 or Joliet file tree node names (long, short and absolute).
	StarBurn_SetBufferUnderrunTimeOutInMs (see page 332)	This function sets buffer underrun timeout in milliseconds. It's amount of time StarBurn core would wait before resubmitting each command to the drive if burning was faster then reading and StarBurn had exhausted software cache it uses for buffering. Modern hardware has BUP (Buffer Underrun Protection) and would perfectly survive in such a condition and keep burning disc still usable.
	StarBurn_SetDeviceTimeOutByDeviceAddress (see page 333)	This function sets device timeout in seconds by device SCSI address.
	StarBurn_SetDVDPadding (see page 333)	This function sets DVD padding mode (when at least 1GB of the data would be recorded to DVD media - required for DVD-Video compilations to work properly on standalone DVD players) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetDVDPLUSRDLCompatibleMode (see page 334)	This function sets DVD+R DL (Dual Layer) so-called "compatible mode" (when layer would be switched exactly at 1/2 of the data payload recorded on the disc - required for DVD-Video compilations to work properly) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetEjectAfterFail (see page 335)	This function sets so-called "eject after fail" (if during burning process error would happen - should StarBurn eject disc or not) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetFastReadTOC (see page 335)	This function sets so-called "fast read TOC" (TOC information is not 100% accurate - to get EXACT track length StarBurn try to read beginning and end of the track to find sub-channel index switch indicating EXACT track beginning or EXACT track end) mode to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetIs64KBIO (see page 336)	This function sets current so-called "64KB I/O" (when StarBurn issues 64KB I/O packets instead of the 32KB ones) to enabled (TRUE) or disabled (FALSE) state.
	StarBurn_SetIsCollisionDetectionDisabled (see page 336)	This function sets collision detection to disabled (TRUE) or enabled (FALSE). If collision detection is enabled - every new file added to ISO9660 or Joliet file tree would be checked to have unique name. If such a file already exist - special callback (collision detection one) would be called so user would be able to either replace or rename new or old file. However if collision detection is disabled - no comparison and no actions would be performed. File would be just added to the destination file system image AS IS, having it's original name.

	StarBurn_SetIsSafeGrabbingEnabled (see page 337)	This function sets so-called "safe grabbing" (when StarBurn would mix read commands with checking for device to become ready - required to workaround broken ATAPI-to-USB bridges) to enabled (TRUE) or disabled (FALSE).
	StarBurn_SetUsingLocalDateTime (see page 337)	This is function StarBurn_SetUsingLocalDateTime.
	StarBurn_SetUsingUTCDateTime (see page 338)	This is function StarBurn_SetUsingUTCDateTime.
	StarBurn_sprintf_s (see page 338)	This function wrap runtime sprintf_s function. In Visual Studio 2003 function sprintf_s not present.
	StarBurn_SPTD_GetVersion (see page 338)	This function checks if the SPTD (SCSI Pass Through Direct) driver is present and gets SPTD driver version.
	StarBurn_StarPort_DeviceAddLocal (see page 339)	This function adds local device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.
	StarBurn_StarPort_DeviceAddLocalEx (see page 340)	This function adds local device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Device can be added persistent.
	StarBurn_StarPort_DeviceAddRemote (see page 341)	This function adds remote device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Device can be mount persistent.
	StarBurn_StarPort_DeviceAddRemoteEx (see page 341)	This function adds remote device (RAM disk, hard disk or DVD-ROM emulation) to StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.
	StarBurn_StarPort_DeviceListQueryLocal (see page 342)	This function gets device list from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver. Unlike StarBurn_StarPort_DeviceListQueryRemote (see page 343) API call which retrieves information about REMOTE device list this call gets information about LOCAL device list.
	StarBurn_StarPort_DeviceListQueryRemote (see page 343)	This function gets device list from remote StarWind iSCSI target. StarWind exports RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) targets. Unlike StarBurn_StarPort_DeviceListQueryLocal (see page 342) API call which retreives information about LOCAL device list this call gets information about REMOTE device list.
	StarBurn_StarPort_DeviceRemove (see page 344)	This function removes device from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.
	StarBurn_StarPort_DeviceRemoveEx (see page 344)	This function removes device from StarPort virtual storage controller. StarPort acts like RAM disk, hard disk and DVD-ROM emulator, AoE (ATA-over-Ethernet) and iSCSI (SCSI-over-IP) initiator driver.
	StarBurn_StarPort_GetDeviceInformation (see page 345)	This function retrieves device type, vendor ID, product ID and revision for StarPort device by its target ID.
	StarBurn_StarPort_GetDeviceLetter (see page 346)	This function retrieves device letter (symbolic link) for StarPort device by its target ID.
	StarBurn_StarPort_GetDeviceSCSIAddress (see page 347)	This function retrieves device SCSI address for StarPort device by its target ID.
	StarBurn_StarPort_GetVersion (see page 347)	This function checks if the StarPortLite driver is installed and that its API version is equal to the one StarBurn was built with.
	StarBurn_StarWave_CompressedFileReaderObjectBeginSeek (see page 348)	This function seeks all of the internal object pointers to the very beginning of it so following read operations would start from the very beginning instead of the currently set position.
	StarBurn_StarWave_CompressedFileReaderObjectCreate (see page 349)	This function creates compressed file reader object from passed compressed audio file name.
	StarBurn_StarWave_CompressedFileReaderObjectCreateUnicode (see page 350)	This is function StarBurn_StarWave_CompressedFileReaderObjectCreateUnicode.
	StarBurn_StarWave_CompressedFileReaderObjectDestroy (see page 350)	This function destroys compressed file reader object.
	StarBurn_StarWave_CompressedFileReaderObjectRead (see page 351)	This function reads uncompressed payload data chunk from underlying compressed file reader object.

	StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet (see page 352)	This function returns uncompressed payload size in UCHARs for underlying compressed file reader object.
	StarBurn_StarWave_CompressedFileReaderObjectUncompressedSizeGet_Fast (see page 353)	This function returns uncompressed payload size in UCHARs for underlying compressed file reader object.
	StarBurn_StarWave_CompressedFileReaderObjectUnicodeCreate (see page 354)	This is function StarBurn_StarWave_CompressedFileReaderObjectUnicodeCreate.
	StarBurn_StarWave_CompressedFileRecompress (see page 354)	This function recompresse already compressed file to new one with another compression.
	StarBurn_StarWave_CompressedFileRecompressUnicode (see page 355)	This is function StarBurn_StarWave_CompressedFileRecompressUnicode.
	StarBurn_StarWave_CompressedFileSupportedIs (see page 355)	This function checks is file name actually points to supported compressed audio file.
	StarBurn_StarWave_CompressedFileUncompress (see page 356)	This function uncompresses compressed file to either WAV or RAW PCM.
	StarBurn_StarWave_CompressedFileUncompressUnicode (see page 357)	This is function StarBurn_StarWave_CompressedFileUncompressUnicode.
	StarBurn_StarWave_CompressedFileWriterObjectCreate (see page 358)	This function creates compressed file writer object.
	StarBurn_StarWave_CompressedFileWriterObjectCreateUnicode (see page 359)	This is function StarBurn_StarWave_CompressedFileWriterObjectCreateUnicode.
	StarBurn_StarWave_CompressedFileWriterObjectDestroy (see page 359)	This function destroys compressed file writer object.
	StarBurn_StarWave_CompressedFileWriterObjectWrite (see page 360)	This function writes uncompressed data chunk to compressed file writer object.
	StarBurn_StarWave_UncompressedFileCompress (see page 361)	This function compresses uncompressed file to new one with asked compression.
	StarBurn_StarWave_UncompressedFileCompressUnicode (see page 362)	This is function StarBurn_StarWave_UncompressedFileCompressUnicode.
	StarBurn_StarWave_UncompressedFileSupportedIs (see page 362)	This function checks is file name actually points to supported uncompressed audio file.
	StarBurn_StarWave_UncompressedFileSupportedUnicodeIs (see page 363)	This function checks is file name actually points to supported uncompressed audio file.
	StarBurn_StarWave_UncompressedFileSupportedUnicodeIsEx (see page 364)	This function checks is file name actually points to supported uncompressed audio file.
	StarBurn_StarWave_VersionGet (see page 364)	This function returns StarWave library version.
	StarBurn_StarWave2_Convert (see page 365)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_ConvertEx (see page 366)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_ConvertExUnicode (see page 367)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_ConvertUnicode (see page 367)	This function convert audio files from any supported format to any other. Supported formats are: WAV (44100Hz, 16, stereo), MP3, WMA, OGG, FLAC, Windows Media formats (ASF, WMV etc.) (for reading)
	StarBurn_StarWave2_EncodeMP3OGGFromWAV (see page 368)	This function compress audio file from WAV (44100Hz, 16, stereo) format to MP3 and OGG format.
	StarBurn_StarWave2_EncodeMP3OGGFromWAVUnicode (see page 369)	This is function StarBurn_StarWave2_EncodeMP3OGGFromWAVUnicode.
	StarBurn_StarWave2_IsFileSupported (see page 369)	This function determines audio file by StarWave2 library supported is.
	StarBurn_StarWave2_IsFileSupportedUnicode (see page 370)	This function determines audio file by StarWave2 library supported is.
	StarBurn_strcpy_s (see page 370)	This function wrap runtime strcpy_s function. In Visual Studio 2003 function strcpy_s not present.
	StarBurn_swprintf_s (see page 371)	This function wrap runtime swprintf_s function. In Visual Studio 2003 oriinal swprintf_s not present.
	StarBurn_UDF_Add (see page 371)	Creates the UDF Tree Node
	StarBurn_UDF_AddUnicode (see page 372)	This is function StarBurn_UDF_AddUnicode.
	StarBurn_UDF_CleanUp (see page 372)	This function removes created UDF tree from the pointed UDF tree node sequentially. Processes file formatted UDF nodes and either closes file handles or frees memory (non-cached vs. cached content).

	StarBurn_UDF_Create (see page 374)	StarBurn_UDF_Create() function is obsolete. Use StarBurn_UDF_CreatEx() instead of it
	StarBurn_UDF_CreateEx (see page 374)	This function creates UDF tree from the allocated data buffers.
	StarBurn_UDF_CreateExUnicode (see page 377)	This is function StarBurn_UDF_CreateExUnicode.
	StarBurn_UDF_CreateUnicode (see page 377)	This is function StarBurn_UDF_CreateUnicode.
	StarBurn_UDF_Destroy (see page 377)	This function devastates created UDF tree from the pointed root recursively.
	StarBurn_UDF_DestroyNodeAndKids (see page 379)	Destroys UDF Tree Node and all its kids
	StarBurn_UDF_FormatTreeItemAsDirectory (see page 379)	This function formats UDF tree item as directory.
	StarBurn_UDF_FormatTreeItemAsDirectoryUnicode (see page 382)	This is function StarBurn_UDF_FormatTreeItemAsDirectoryUnicode.
	StarBurn_UDF_FormatTreeItemAsFile (see page 382)	This function formats UDF tree item as file.
	StarBurn_UDF_FormatTreeItemAsFileUnicode (see page 384)	This is function StarBurn_UDF_FormatTreeItemAsFileUnicode.
	StarBurn_UDF_GetFirstChild (see page 385)	This function returns the pointer to first child node
	StarBurn_UDF_GetNextSibling (see page 385)	This function returns the pointer to next sibling node
	StarBurn_UDF_GetNodeObject (see page 385)	Returns the object to current Tree Node
	StarBurn_UDF_GetParent (see page 386)	This function returns the pointer to next parent node
	StarBurn_UDF_GetPreviousSibling (see page 386)	This function returns the pointer to previous sibling node
	StarBurn_UDF2_DirectoryBrowse (see page 387)	This function browses UDF directory content.
	StarBurn_UDF2_DirectoryContentsProcess (see page 387)	This function processes contents of single UDF directory.
	StarBurn_UDF2_DirectoryContentsProcessEx (see page 388)	This function processes contents of single UDF directory.
	StarBurn_UDF2_DirectoryContentsUnicodeProcess (see page 388)	This function processes contents of single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryContentsUnicodeProcessEx (see page 389)	This function processes contents of single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryCreate (see page 390)	This function creates UDF directory.
	StarBurn_UDF2_DirectoryDestroy (see page 390)	This function destroys created UDF directory object.
	StarBurn_UDF2_DirectoryProcess (see page 390)	This function processes single UDF directory.
	StarBurn_UDF2_DirectoryProcessEx (see page 391)	This function processes single UDF directory.
	StarBurn_UDF2_DirectoryProcessExEx (see page 392)	This function processes single UDF directory.
	StarBurn_UDF2_DirectoryQuery (see page 392)	This function does query for UDF directory properties.
	StarBurn_UDF2_DirectoryRootCreate (see page 393)	This function creates root UDF directory.
	StarBurn_UDF2_DirectoryUnicodeCreate (see page 393)	This function creates UNICODE UDF directory.
	StarBurn_UDF2_DirectoryUnicodeProcess (see page 394)	This function processes single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryUnicodeProcessEx (see page 394)	This function processes single UNICODE UDF directory.
	StarBurn_UDF2_DirectoryUnicodeProcessExEx (see page 395)	This function processes single UNICODE UDF directory.
	StarBurn_UDF2_FileBootCreate (see page 395)	This function creates UDF boot file.
	StarBurn_UDF2_FileBootUnicodeCreate (see page 396)	This function creates UNICODE UDF boot file.
	StarBurn_UDF2_FileCreate (see page 396)	This function creates UDF file.
	StarBurn_UDF2_FileDestroy (see page 397)	This function destroys created UDF file object.
	StarBurn_UDF2_FileDirectoryDateTimeGet (see page 397)	This function gets date and time from created UDF file or directory object.
	StarBurn_UDF2_FileDirectoryDateTimeGetEx (see page 398)	This function gets date and time from created UDF file or directory object. It may get creation, last access, or last write date and time.
	StarBurn_UDF2_FileDirectoryDateTimeSet (see page 398)	This function sets date and time on created UDF file or directory object.
	StarBurn_UDF2_FileDirectoryDateTimeSetEx (see page 399)	This function sets date and time on created UDF file or directory object. It may set creation, last access, or last write date and time.
	StarBurn_UDF2_FileDirectoryNameSet (see page 399)	This function set new name for UDF file or directory object.
	StarBurn_UDF2_FileDirectoryUnicodeNameSet (see page 400)	This function set new UNICODE name for UDF file or directory object.
	StarBurn_UDF2_FileMemoryCreate (see page 400)	This function creates UDF memory file.
	StarBurn_UDF2_FileMemoryUnicodeCreate (see page 401)	This function creates UNICODE UDF memory file.
	StarBurn_UDF2_FileQuery (see page 401)	This function do query for UDF file properties.
	StarBurn_UDF2_FileSetAttributes (see page 402)	This function sets attributes for UDF file.
	StarBurn_UDF2_FileSystemGetSizes (see page 402)	This function calculates UDF file system sizes in Bytes and LBS on the fly (without build volume).
	StarBurn_UDF2_FileUnicodeCreate (see page 403)	This function creates UNICODE UDF file.
	StarBurn_UDF2_ImpVolumeCreate (see page 403)	This function creates UDF import volume.

	StarBurn_UDF2_ImpVolumeDestroy (see page 404)	This function destroys created UDF import volume object.
	StarBurn_UDF2_ImpVolumeImport (see page 404)	This function imports UDF import volume content.
	StarBurn_UDF2_ISO9660FileTreeCreate (see page 404)	This function create ISO image from already created UDF volume.
	StarBurn_UDF2_ISO9660FileTreeCreateEx (see page 405)	This function create ISO image from already created UDF volume.
	StarBurn_UDF2_VolumeAnchorGet (see page 405)	This function gets first anchor volume descriptor pointer we'll have to replace on rewritable media.
	StarBurn_UDF2_VolumeCreate (see page 406)	This function creates UDF 1.02 volume.
	StarBurn_UDF2_VolumeCreateBoot (see page 406)	This function creates UDF Bootable volume.
	StarBurn_UDF2_VolumeCreateBootElTorito (see page 408)	This function creates UDF Bootable volume.
	StarBurn_UDF2_VolumeCreateEx (see page 409)	This function creates UDF volume.
	StarBurn_UDF2_VolumeCreateUnicodeBootElTorito (see page 410)	This function creates UDF Bootable volume.
	StarBurn_UDF2_VolumeDestroy (see page 411)	This function destroys created UDF volume object.
	StarBurn_UDF2_VolumeInitialSizeInLBsGet (see page 411)	This function get UDF volume size in unsigned chars.
	StarBurn_UDF2_VolumeSeekRead (see page 412)	This function seeks to passed offset and reads requested amount of data.
	StarBurn_UDF2_VolumeSizeInLBsGet (see page 412)	This function get UDF volume size in unsigned chars.
	StarBurn_UDF2_VolumeSizeInUCHARsGet (see page 413)	This function get UDF volume size in unsigned chars.
	StarBurn_UDF2_VolumeStore (see page 413)	This function stores UDF volume to file.
	StarBurn_UDF2_VolumeUnicodeCreate (see page 413)	This function creates UDF 1.02 UNICODE volume.
	StarBurn_UDF2_VolumeUnicodeCreateBoot (see page 414)	This function creates UDF Unicode Bootable volume.
	StarBurn_UDF2_VolumeUnicodeCreateEx (see page 415)	This function creates UDF UNICODE volume.
	StarBurn_UDFBridge_CreateEx (see page 416)	This is function StarBurn_UDFBridge_CreateEx.
	StarBurn_UDFBridge_CreateExUnicode (see page 417)	This is function StarBurn_UDFBridge_CreateExUnicode.
	StarBurn_UpStart (see page 417)	This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used. License file StarBurn.key is assumed to be located near main application executable. Since StarBurn SDK V12 it's possible to use this call again. It will just enable using embedded evaluational StarBurn SDK key (file shipped with StarBurn SDK... more (see page 417)
	StarBurn_UpStartEx (see page 418)	This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Starting from build 4.2.6 it's *REQUIRED* to call this function, it does not matter what build (static Vs. dynamic) of StarBurn is used. The difference between this call and call to StarBurn_UpStart (see page 417)(...) is that this particular call assumes StarBurn key is being embedded to the application binary and StarBurn_UpStart (see page 417)(...) always looks for StarBurn.key near application executable. Use this API... more (see page 418)
	StarBurn_UpStartExEx (see page 419)	This function initializes burning toolkit. It's expected to be called as the very first function call before calling any other StarBurn exported code. Some of StarBurn functions would work with not initialized core, some would fail with EN_REGISTRATION_FAILED error. Unlike StarBurn_UpStartEx (see page 418)(...) API call this one allows to use custom debug output control. See StarBurn_UpStartEx (see page 418)(...) API call for more information about this topic.

Macros

Name	Description
__STARBURN_H__ (see page 596)	Define StarBurn.h to avoid including it more then once
ASPI_SAFE_BUFFER_SIZE_IN_UCHARS (see page 596)	ASPI safe buffer size in UCHARs. ASPI is assumed to support at least 64KB large transfer per time. 64KB - PAGE_SIZE_IN_UCHARS (see page 620) was old default value. Now we'll use DVD_ECC_BLOCK_SIZE_IN_UCHARS (see page 605) as it's smaller and produce better results when dealing with DVD media
AUDIO_LB_SIZE_IN_UCHARS (see page 597)	Audio LB (logical block) size in UCHARs
BLURAY_SPEED_IN_KBPS_1X (see page 597)	Actually 4495.5

BLURAY_SPEED_IN_KBPS_2X (see page 597)	2X Blu-Ray speed constant (BD-R and BD-RE)
BUFFER_STATUS_REPORT_DELAY_IN_SECONDS (see page 597)	Default buffer status report delay in seconds
CACHE_SIZE_IN_MBS (see page 598)	64
CD_SPEED_IN_KBPS_10X (see page 598)	10X CD speed constant
CD_SPEED_IN_KBPS_12X (see page 598)	12X CD speed constant
CD_SPEED_IN_KBPS_16X (see page 598)	16X CD speed constant
CD_SPEED_IN_KBPS_1X (see page 599)	1X CD speed constant
CD_SPEED_IN_KBPS_20X (see page 599)	20X CD speed constant
CD_SPEED_IN_KBPS_24X (see page 599)	24X CD speed constant
CD_SPEED_IN_KBPS_2P2X (see page 599)	2.2X CD speed constant
CD_SPEED_IN_KBPS_2X (see page 600)	2X CD speed constant
CD_SPEED_IN_KBPS_32X (see page 600)	32X CD speed constant
CD_SPEED_IN_KBPS_36X (see page 600)	36X CD speed constant
CD_SPEED_IN_KBPS_3X (see page 600)	3X CD speed constant
CD_SPEED_IN_KBPS_40X (see page 601)	40X CD speed constant
CD_SPEED_IN_KBPS_44X (see page 601)	44X CD speed constant
CD_SPEED_IN_KBPS_48X (see page 601)	48X CD speed constant
CD_SPEED_IN_KBPS_4X (see page 601)	4X CD speed constant
CD_SPEED_IN_KBPS_52X (see page 602)	52X CD speed constant
CD_SPEED_IN_KBPS_56X (see page 602)	56X CD speed
CD_SPEED_IN_KBPS_6X (see page 602)	6X CD speed constant
CD_SPEED_IN_KBPS_72X (see page 602)	72X CD speed
CD_SPEED_IN_KBPS_8X (see page 603)	8X CD speed constant
CDVD_SPEED_IS_KBPS_MAXIMUM (see page 603)	Top supported CD/DVD/Blu-Ray/HD-DVD speed constant (wildcard)
DEFAULT_NUMBER_OF_VERIFY_READ_RETRIES (see page 603)	Default number of read retries during verification process
DEVICES_PER_BUS (see page 603)	Top number of SCSI devices per logical SCSI bus (was 15 originally)
DISC_STATUS_COMPLETE (see page 604)	Disc is complete
DISC_STATUS_EMPTY (see page 604)	Disc is empty
DISC_STATUS_INCOMPLETE (see page 604)	Disc is incomplete
DUAL_LAYER_DVD_CAPACITY_SIZE_IN_LBS (see page 604)	Doubled...
DVD_ECC_BLOCK_SIZE_IN_LBS (see page 605)	DVD ECC block size in logical blocks (2048 UCHARs each)
DVD_ECC_BLOCK_SIZE_IN_UCHARS (see page 605)	DVD ECC block size in UCHARs. This is true "hardware" logical block size on DVD. It's recommended to have transfers of this size and this size aligned
DVD_PROTECTION_CPRM (see page 605)	DVD with CPRM (Copy-Protected Removable Media) protection
DVD_PROTECTION_CSS (see page 605)	DVD with CSS (Content Scrambling System) protection
DVD_PROTECTION_NONE (see page 606)	DVD w/o any protection system
DVD_SPEED_IN_KBPS_10X (see page 606)	10X DVD speed constant (DVD-R, DVD-R DL, DVD+R, DVD+R DL and DVD+RW)
DVD_SPEED_IN_KBPS_12X (see page 606)	12X DVD speed constant (DVD-R, DVD+R)
DVD_SPEED_IN_KBPS_16X (see page 606)	16X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_18X (see page 607)	18X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_1X (see page 607)	1X DVD speed constant (DVD-R, DVD-RW and DVD-RAM)
DVD_SPEED_IN_KBPS_20X (see page 607)	20X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_22X (see page 607)	22X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_24X (see page 608)	24X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_2DOT4X (see page 608)	2.4X DVD speed constant (DVD+R, DVD+R DL and DVD+RW)
DVD_SPEED_IN_KBPS_2X (see page 608)	2X DVD speed constant (DVD-R, DVD-RW and DVD-RAM)
DVD_SPEED_IN_KBPS_32X (see page 608)	32X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_3X (see page 609)	3X DVD speed constant (DVD-RAM)
DVD_SPEED_IN_KBPS_40X (see page 609)	40X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_48X (see page 609)	48X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_4X (see page 609)	4X DVD speed constant (DVD-R, DVD-R DL, DVD-RW, DVD+R, DVD+R DL, DVD+RW and DVD-RAM)
DVD_SPEED_IN_KBPS_50X (see page 610)	50X DVD speed constant (DVD+R and DVD-R)
DVD_SPEED_IN_KBPS_5X (see page 610)	5X DVD speed constant (DVD-RAM)
DVD_SPEED_IN_KBPS_6X (see page 610)	6X DVD speed constant (DVD-RW)
DVD_SPEED_IN_KBPS_8X (see page 610)	8X DVD speed constant (DVD-R, DVD-R DL, DVD+R, DVD+R DL and DVD+RW)
ELTORITO_BOOTABLE (see page 611)	Bootable disc ID
ELTORITO_BOOTSYSTEM_ID (see page 611)	Bootable specification signature

ELTORITO_DEF_LOAD_SEGMENT (see page 611)	Real mode code load segment
ELTORITO_NON_BOOTABLE (see page 611)	Non-bootable disc ID
ERROR_FIELD_C2_AND_BLOCK_ERROR (see page 612)	C2 and block error information
ERROR_FIELD_C2_ERROR (see page 612)	C2 error information
ERROR_FIELD_NO_ERROR (see page 612)	No error information
ERROR_FIELD_RESERVED (see page 612)	Reserved
EXPECTED_LB_TYPE_ANY (see page 613)	Accept all LBs
EXPECTED_LB_TYPE_CDDA (see page 613)	Accept only CDDA (CD digital audio) LBs
EXPECTED_LB_TYPE_MODE1 (see page 613)	Accept only MODE1 LBs
EXPECTED_LB_TYPE_MODE2 (see page 613)	Accept only MODE2 LBs (both Form1 and Form2)
EXPECTED_LB_TYPE_MODE2_FORM1 (see page 614)	Accept only MODE2 Form1 LBs
EXPECTED_LB_TYPE_MODE2_FORM2 (see page 614)	Accept only MODE2 Form2 LBs
HARD_DISK_LB_SIZE_IN_UCHARS (see page 614)	Hard disk LB (logical block) size in UCHARs
HEADER_CODE_ALL_HEADERS (see page 614)	Both header and subheader
HEADER_CODE_HEADER_ONLY (see page 615)	4-byte header
HEADER_CODE_NONE (see page 615)	No header
HEADER_CODE_SUBHEADER_ONLY (see page 615)	MODE2 Form1 or Form2 subheader
ISO9660_DATE_SIZE_IN_UCHARS (see page 615)	ISO9660 date size in UCHARs
ISO9660_FILE_SIZE_IN_UCHARS (see page 616)	ISO9660 file size in UCHARs
ISO9660_NAME_SIZE_IN_UCHARS (see page 616)	ISO9660 name size in UCHARs
ISO9660_SYSTEM_VOLUME_ID_SIZE_IN_UCHARS (see page 616)	ISO9660 volume descriptor system and volume ID size in UCHARs
ISO9660_TREE_LEVEL (see page 616)	ISO9660 top supported file tree level
LAST_SESSION_COMPLETE (see page 617)	Last session is complete
LAST_SESSION_EMPTY (see page 617)	Last session is empty
LAST_SESSION_INCOMPLETE (see page 617)	Last session is incomplete
MODE1_LB_SIZE_IN_UCHARS (see page 617)	MODE1 LB (logical block) size in UCHARs
MODE2_FORM1_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Form1 LB (logical block) size in UCHARs
MODE2_FORM2_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Form2 LB (logical block) size in UCHARs
MODE2_FORM2_SUB_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Form2 + 8 UCHARs of subheader LB (logical block) size in UCHARs
MODE2_FORMLESS_LB_SIZE_IN_UCHARS (see page 618)	MODE2 Formless LB (logical block) size in UCHARS
NUMBER_OF_RAW_TRACKS (see page 619)	Top number of raw tracks (including maintenance tracks) on CD/DVD/Blu-Ray/HD-DVD disc. This is calculated based on the worst CD case when all tracks will be in a separate sessions. We need this approximate value to avoid allocating TOC_INFORMATION (see page 573) structure dynamically.
NUMBER_OF_READ_RETRIES (see page 619)	Default number of retries on read operations (if bad block was possibly hit, was 2 originally)
NUMBER_OF_SUBCHANNELS (see page 619)	sub-channels from P to W
NUMBER_OF_TRACKS (see page 619)	Top number of tracks (or border-in/border-out zones) on CD/DVD/Blu-Ray/HD-DVD disc
PAGE_SIZE_IN_UCHARS (see page 620)	Small page size in UCHARs on x86 machines (default allocatable storage)
RAW_LB_PQ_SUB_SIZE_IN_UCHARS (see page 620)	RAW LB (logical block) with PQ sub-channel size in UCHARs
RAW_LB_RAW_PW_SUB_SIZE_IN_UCHARS (see page 620)	RAW LB (logical block) with RAW P-W sub-channel size in UCHARs
RAW_LB_SIZE_IN_UCHARS (see page 620)	RAW LB (logical block) size in UCHARs
READ_REPORT_DELAY_IN_SECONDS (see page 621)	Default read report delay in seconds
SCSI_DEVICE_DIRECT_ACCESS (see page 621)	SCSI disk device
SCSI_DEVICE_MAGNETO_OPTICAL (see page 621)	SCSI MO (Magneto-Optical) disk device
SCSI_DEVICE_MEDIUM_CHANGER (see page 621)	SCSI jukebox (disc changer) device
SCSI_DEVICE_NETWORK (see page 622)	SCSI network device
SCSI_DEVICE_PRINTER (see page 622)	SCSI printer device
SCSI_DEVICE_PROCESSOR (see page 622)	SCSI processor device
SCSI_DEVICE_RO_DIRECT_ACCESS (see page 622)	SCSI CD/DVD/Blu-Ray/HD-DVD device
SCSI_DEVICE_SCANNER (see page 623)	SCSI scanner device
SCSI_DEVICE_SEQUENTIAL_ACCESS (see page 623)	SCSI tape device
SCSI_DEVICE_UNPRESENT (see page 623)	SCSI device... Is gone!
SCSI_DEVICE_WILDCARD (see page 623)	SCSI device any (wildcard for StarBurn_FindDevice (see page 231)() function)
SCSI_DEVICE_WORM (see page 624)	SCSI WORM (Write-Once-Read-Multiple) device
SHORT_RAW_LB_SIZE_IN_UCHARS (see page 624)	Short (w/o sync and header) RAW LB (logical block) size in UCHARs
SINGLE_LAYER_DVD_CAPACITY_SIZE_IN_LBS (see page 624)	Number of logical blocks on single layer DVD

SMALLEST_CACHE_SIZE_IN_MBS (see page 624)	Smallest cache size in MBs that toolkit will allow to use to buffer "write" operations
SMALLEST_CACHE_SIZE_IN_UCHARS (see page 625)	Smallest cache size in UCHARs that toolkit will allow to use to buffer "write" operations
STARBURN_BDRE_FORMAT_CERTIFICATION_FULL (see page 625)	This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_FULL.
STARBURN_BDRE_FORMAT_CERTIFICATION_NO (see page 625)	BD-RE format certification types
STARBURN_BDRE_FORMAT_CERTIFICATION_QUICK (see page 625)	This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_QUICK.
STARBURN_BDRE_FORMAT_CERTIFICATION_RESERVED (see page 626)	This is macro STARBURN_BDRE_FORMAT_CERTIFICATION_RESERVED.
STARBURN_BDRE_FORMAT_DEFAULT (see page 626)	BD-RE format types
STARBURN_BDRE_FORMAT_SPARE_AREA_EXPANSION (see page 626)	This is macro STARBURN_BDRE_FORMAT_SPARE_AREA_EXPANSION.
STARBURN_BDRE_FORMAT_WITH_SPARE_AREA (see page 627)	This is macro STARBURN_BDRE_FORMAT_WITH_SPARE_AREA.
STARBURN_BDRE_FORMAT_WITHOUT_SPARE_AREA (see page 627)	This is macro STARBURN_BDRE_FORMAT_WITHOUT_SPARE_AREA.
STARBURN_CACHE_SIZE_READ_ONLY (see page 627)	Default cache size for I/O operations
STARBURN_CACHE_SIZE_READ_WRITE (see page 627)	This is macro STARBURN_CACHE_SIZE_READ_WRITE.
STARBURN_CDFS2_FILE_ATTRIBUTE_DELETED (see page 628)	This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_DELETED.
STARBURN_CDFS2_FILE_ATTRIBUTE_DIRECTORY (see page 628)	File attributes (flags for Attributes member of STARBURN_UDF2_FILE_INFO (see page 567) & STARBURN_ISO2_FILE_INFO)
STARBURN_CDFS2_FILE_ATTRIBUTE_HIDDEN (see page 628)	This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_HIDDEN.
STARBURN_CDFS2_FILE_ATTRIBUTE_IMPORTED (see page 629)	This is macro STARBURN_CDFS2_FILE_ATTRIBUTE_IMPORTED.
StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileT (see page 629)	This is macro StarBurn_CdvdBurnerGrabber_DiscAtOncePQFromFileT.
StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileT (see page 629)	This is macro StarBurn_CdvdBurnerGrabber_DiscAtOnceRawPWFromFileT.
StarBurn_CdvdBurnerGrabber_GetDeviceInformationT (see page 630)	This is macro StarBurn_CdvdBurnerGrabber_GetDeviceInformationT.
StarBurn_CdvdBurnerGrabber_GrabTrackT (see page 630)	This is macro StarBurn_CdvdBurnerGrabber_GrabTrackT.
StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWT (see page 630)	This is macro StarBurn_CdvdBurnerGrabber_SessionAtOnceRawRawPWT.
StarBurn_CdvdBurnerGrabber_SessionAtOnceT (see page 631)	This is macro StarBurn_CdvdBurnerGrabber_SessionAtOnceT.
StarBurn_CdvdBurnerGrabber_SuperVideoCDExExT (see page 631)	This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDExExT.
StarBurn_CdvdBurnerGrabber_SuperVideoCDExT (see page 631)	This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDExT.
StarBurn_CdvdBurnerGrabber_SuperVideoCDT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_SuperVideoCDT.
StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_TrackAtOnceFromFileT.
StarBurn_CdvdBurnerGrabber_VerifyFileExT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_VerifyFileExT.
StarBurn_CdvdBurnerGrabber_VerifyFileT (see page 632)	This is macro StarBurn_CdvdBurnerGrabber_VerifyFileT.
StarBurn_CdvdBurnerGrabber_VideoCDExExT (see page 633)	This is macro StarBurn_CdvdBurnerGrabber_VideoCDExExT.
StarBurn_CdvdBurnerGrabber_VideoCDExT (see page 633)	This is macro StarBurn_CdvdBurnerGrabber_VideoCDExT.
StarBurn_CdvdBurnerGrabber_VideoCDT (see page 633)	This is macro StarBurn_CdvdBurnerGrabber_VideoCDT.
STARBURN_DEBUG_FACILITY_ALLMESSAGES (see page 634)	Debug facility: Permit All Messages
STARBURN_DEBUG_FACILITY_ALLTARGETS (see page 634)	Debug facility: All targets
STARBURN_DEBUG_FACILITY_CUSTOM (see page 634)	Debug facility: Custom
STARBURN_DEBUG_FACILITY_DEBUG (see page 634)	Debug facility: Permit Debug Messages
STARBURN_DEBUG_FACILITY_DEBUG_OUTPUT (see page 635)	Debug facility: Debug output
STARBURN_DEBUG_FACILITY_ERROR (see page 635)	Debug facility: Permit Error Messages
STARBURN_DEBUG_FACILITY_LOG_FILE (see page 635)	Debug facility: Log file
STARBURN_DEBUG_FACILITY_NONE (see page 635)	Debug facility: None
STARBURN_DEBUG_FACILITY_SYSTEM_CONSOLE (see page 636)	Debug facility: System console
STARBURN_DEBUG_FACILITY_TRACE (see page 636)	Debug facility: Permit Trace Messages
STARBURN_DEBUG_FACILITY_WARNING (see page 636)	Debug facility: Permit Warning Messages
STARBURN_DISC_TYPE_CDDA (see page 636)	Disc type is CDDA (CD-ROM or Digital Audio)
STARBURN_DISC_TYPE_CDI (see page 637)	Disc type is CDI
STARBURN_DISC_TYPE_UNDEFINED (see page 637)	Disc type is undefined
STARBURN_DISC_TYPE_XA (see page 637)	Disc type is CD-ROM XA
STARBURN_DVD_VIDEO_MAX_NUMBER_OF_TITLES (see page 637)	DVD-Video maximum number of titles
StarBurn_DVDVideo_CreateT (see page 638)	This is macro StarBurn_DVDVideo_CreateT.
StarBurn_ElTorito_BootCatalogAddSectionT (see page 638)	This is macro StarBurn_ElTorito_BootCatalogAddSectionT.
StarBurn_ElTorito_CreateBootCatalogT (see page 638)	This is macro StarBurn_ElTorito_CreateBootCatalogT.
StarBurn_GetAudioFileStreamSizeInUCHARs_FastT (see page 638)	This is macro StarBurn_GetAudioFileStreamSizeInUCHARs_FastT.
StarBurn_GetAudioFileStreamSizeInUCHARsT (see page 639)	This is macro StarBurn_GetAudioFileStreamSizeInUCHARsT.
StarBurn_GetDeviceLetterT (see page 639)	TCHAR functions
STARBURN_IMPEX_API (see page 639)	declspec(dllexport)

StarBurn_IsAudioFileSupportedT (see page 639)	This is macro StarBurn_IsAudioFileSupportedT.
StarBurn_ISO2_VolumeCreateBootElToritoT (see page 640)	This is macro StarBurn_ISO2_VolumeCreateBootElToritoT.
STARBURN_ISO9660_FILE_FLAGS_ASSOCIATED_FILE (see page 640)	This is macro STARBURN_ISO9660_FILE_FLAGS_ASSOCIATED_FILE.
STARBURN_ISO9660_FILE_FLAGS_DIRECTORY (see page 640)	This is macro STARBURN_ISO9660_FILE_FLAGS_DIRECTORY.
STARBURN_ISO9660_FILE_FLAGS_EXISTENCE (see page 641)	This is macro STARBURN_ISO9660_FILE_FLAGS_EXISTENCE.
STARBURN_ISO9660_FILE_FLAGS_NONE (see page 641)	ISO9660 file attributes flags
STARBURN_ISO9660_FILE_FLAGS_RECORD (see page 641)	This is macro STARBURN_ISO9660_FILE_FLAGS_RECORD.
STARBURN_ISO9660_FILE_MAX_SIZE_IN_UCHARS (see page 641)	(4GB - 1)
STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS (see page 642)	This is macro STARBURN_ISO9660_JOLIET_NAME_SIZE_IN_WCHARS.
STARBURN_ISO9660_JOLIET_RELAXED_NAME_SIZE_IN_WCHARS (see page 642)	
STARBURN_ISO9660_NAME_SIZE_IN_SYMBOLS (see page 642)	ISO9660 constants (begin)
STARBURN_ISO9660_TYPE_1999 (see page 643)	This is macro STARBURN_ISO9660_TYPE_1999.
STARBURN_ISO9660_TYPE_JOLIET (see page 643)	This is macro STARBURN_ISO9660_TYPE_JOLIET.
STARBURN_ISO9660_TYPE_JOLIET_RELAXED (see page 643)	This is macro STARBURN_ISO9660_TYPE_JOLIET_RELAXED.
STARBURN_ISO9660_TYPE_LEVEL1 (see page 643)	ISO9660 file system type
STARBURN_ISO9660_TYPE_LEVEL2 (see page 644)	This is macro STARBURN_ISO9660_TYPE_LEVEL2.
STARBURN_LOADING_MECHANISM_TYPE_CADDY (see page 644)	Loading mechanism type constants
STARBURN_LOADING_MECHANISM_TYPE_CARTRIDGE (see page 644)	This is macro STARBURN_LOADING_MECHANISM_TYPE_CARTRIDGE.
STARBURN_LOADING_MECHANISM_TYPE_CHANGER (see page 644)	This is macro STARBURN_LOADING_MECHANISM_TYPE_CHANGER.
STARBURN_LOADING_MECHANISM_TYPE_POP_UP (see page 645)	This is macro STARBURN_LOADING_MECHANISM_TYPE_POP_UP.
STARBURN_LOADING_MECHANISM_TYPE_RESERVED1 (see page 645)	This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED1.
STARBURN_LOADING_MECHANISM_TYPE_RESERVED2 (see page 645)	This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED2.
STARBURN_LOADING_MECHANISM_TYPE_RESERVED3 (see page 646)	This is macro STARBURN_LOADING_MECHANISM_TYPE_RESERVED3.
STARBURN_LOADING_MECHANISM_TYPE_TRAY (see page 646)	This is macro STARBURN_LOADING_MECHANISM_TYPE_TRAY.
STARBURN_PIPE_SIZE_IN_UCHARS (see page 646)	60, 64 or 32 KB I/O packets
STARBURN_SMALLEST_SPLIT_FILE_SIZE_IN_MBS (see page 646)	StarBurn smallest split file size in megabytes
StarBurn_StarWave_CompressedFileReaderObjectCreateT (see page 647)	This is macro StarBurn_StarWave_CompressedFileReaderObjectCreateT.
StarBurn_StarWave_CompressedFileRecompressT (see page 647)	This is macro StarBurn_StarWave_CompressedFileRecompressT.
StarBurn_StarWave_CompressedFileUncompressT (see page 647)	This is macro StarBurn_StarWave_CompressedFileUncompressT.
StarBurn_StarWave_CompressedFileWriterObjectCreateT (see page 648)	This is macro StarBurn_StarWave_CompressedFileWriterObjectCreateT.
STARBURN_STARWAVE_IO_BUFFER_SIZE_IN_UCHARS (see page 648)	StarWave recommended I/O buffer size in UCHARs. It's *STRONGLY* recommended to use either it or multiplications of STARWAVE_IO_BUFFER_SIZE_IN_UCHARS as internal calculations and indexations are integer rather then float based
StarBurn_StarWave_UncompressedFileCompressT (see page 648)	This is macro StarBurn_StarWave_UncompressedFileCompressT.
StarBurn_StarWave_UncompressedFileSupportedIsT (see page 649)	This is macro StarBurn_StarWave_UncompressedFileSupportedIsT.
StarBurn_StarWave2_ConvertExT (see page 649)	This is macro StarBurn_StarWave2_ConvertExT.
StarBurn_StarWave2_EncodeMP3OGGFromWAVT (see page 649)	This is macro StarBurn_StarWave2_EncodeMP3OGGFromWAVT.
StarBurn_StarWave2_GetFileReaderAndFactoryT (see page 649)	This is macro StarBurn_StarWave2_GetFileReaderAndFactoryT.
StarBurn_UDF_AddT (see page 650)	This is macro StarBurn_UDF_AddT.
StarBurn_UDF_CreateExT (see page 650)	This is macro StarBurn_UDF_CreateExT.
StarBurn_UDF_CreateT (see page 650)	This is macro StarBurn_UDF_CreateT.
StarBurn_UDF_FormatTreeItemAsDirectoryT (see page 650)	This is macro StarBurn_UDF_FormatTreeItemAsDirectoryT.
StarBurn_UDF_FormatTreeItemAsFileT (see page 651)	This is macro StarBurn_UDF_FormatTreeItemAsFileT.
STARBURN_UDF2_NAME_SIZE_IN_SYMBOLS (see page 651)	Name constants
STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS (see page 651)	This is macro STARBURN_UDF2_PATH_SIZE_IN_SYMBOLS.
StarBurn_UDF2_VolumeCreateBootElToritoT (see page 652)	This is macro StarBurn_UDF2_VolumeCreateBootElToritoT.
StarBurn_UDFBridge_CreateExT (see page 652)	This is macro StarBurn_UDFBridge_CreateExT.
STARPORT_DEVICE_NAME_SIZE_IN_UCHARS (see page 652)	StarPort device name size in UCHARs
STARWAVEFACTORY_ID_DSHOW (see page 652)	This is macro STARWAVEFACTORY_ID_DSHOW.
STARWAVEFACTORY_ID_MP3 (see page 653)	This is macro STARWAVEFACTORY_ID_MP3.
STARWAVEFACTORY_ID_OGG (see page 653)	This is macro STARWAVEFACTORY_ID_OGG.
STARWAVEFACTORY_ID_WAV (see page 653)	StarWave2 factory ids
STARWAVEFACTORY_ID_WMA (see page 653)	This is macro STARWAVEFACTORY_ID_WMA.
SUBCHANNEL_NO (see page 654)	No sub-channel
SUBCHANNEL_PQ (see page 654)	PQ sub-channel
SUBCHANNEL_PQ_SIZE_IN_UCHARS (see page 654)	PQ sub-channel size in UCHARs

SUBCHANNEL_RAW_PW (see page 654)	RAW P-W sub-channel
SUBCHANNEL_RAW_PW_SIZE_IN_UCHARS (see page 655)	Raw P-W sub-channel size in UCHARs
SUBCHANNEL_RESERVED (see page 655)	Reserved
SUBCHANNEL_RW (see page 655)	R-W sub-channel
SUBCHANNEL_SIZE_IN_UCHARS (see page 655)	Sub-channel size in UCHARs
TRACK_NUMBER_INVISIBLE (see page 656)	Special "invisible" track number
TYPE_CODE_LAST_CHANCE (see page 656)	Logical Unit region is set. Additional restrictions required to make a change
TYPE_CODE_NONE (see page 656)	No Logical Unit region setting
TYPE_CODE_PERM (see page 656)	Logical Unit region has been set permanently, but may be reset by vendor if necessery
TYPE_CODE_SET (see page 657)	Logical Unit region is set
UDF_FILE_SET_DESCRIPTOR_LBA (see page 657)	UDF file set descriptor logical block address
UDF_FILE_SET_DESCRIPTOR_TERMINATOR_LBA (see page 657)	UDF file set descriptor terminator logical block address
UDF_FIRST_ANCHOR_LBA (see page 657)	First anchor volume descriptor pointer LBA
UDF_HEAD_SIZE_IN_LOGICAL_BLOCKS (see page 658)	UDF head size in logical blocks
UDF_ISO9660_FILE_SYSTEM_LBA (see page 658)	UDF/ISO9660 bridge file system logical block address
UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS (see page 658)	define UDF_ISO9660_FILE_SYSTEM_SIZE_IN_LBS 6
UDF_LOGICAL_BLOCK_SIZE_IN_UCHARS (see page 658)	UDF logical block size in UCHARs
UDF_NAME_SIZE_IN_UCHARS (see page 659)	UDF longest name size in UCHARs
UDF_NAME_SIZE_IN_UCHARS_ACTUAL (see page 659)	This is macro UDF_NAME_SIZE_IN_UCHARS_ACTUAL.
UDF_NSR_DESCRIPTOR_LBA (see page 659)	UDF NSR descriptor pointer LBA
UDF_TAIL_SIZE_IN_LOGICAL_BLOCKS (see page 659)	UDF tail size in logical blocks
WAVE_FILE_ALIGNMENT (see page 660)	WAVE file data alignment (4 UCHARs, 1 ULONG)
WAVE_FILE_BITS_PER_SAMPLE (see page 660)	WAVE file bits per sample
WAVE_FILE_CHANNELS (see page 660)	WAVE file number of channels
WAVE_FILE_DATA_SIGNATURE (see page 660)	WAVE file DATA signature
WAVE_FILE_RIFF_SIGNATURE (see page 661)	WAVE file RIFF signature
WAVE_FILE_SAMPLES_PER_SECOND (see page 661)	WAVE file samples per second
WAVE_FILE_TAG_SIGNATURE (see page 661)	WAVE file TAG signature
WAVE_FILE_WAVE_FORMAT (see page 661)	WAVE file WAVE format
WAVE_FILE_WAVE_SIGNATURE (see page 662)	WAVE file WAVE signature
WRITE_REPORT_DELAY_IN_SECONDS (see page 662)	Default write report delay in seconds

Structures

	Name	Description
	_CDB_FAILURE_INFORMATION (see page 428)	Structure that represents CDB failure information
	_DAO_DISC_LAYOUT (see page 428)	Structure that represents DAO disc layout
	_DAO_DISC_LAYOUT_ENTRY (see page 429)	Structure that represents DAO disc layout entry (track)
	_DISC_FILESYSTEM (see page 430)	Structure with disc file system flags
	_DISC_LAYOUT (see page 430)	Structure that represents disc layout
	_DISC_LAYOUT_ENTRY (see page 431)	Structure that represents disc layout entry (track)
	_DVD_VIDEO_CONTROL_BLOCK (see page 434)	Structure that represents DVD-Video control block and pointer to DVD-Video control block
	_FULL_TOC_ENTRY_RAW (see page 439)	Structure that represents full TOC entry
	_ISO9660_DATE_TIME (see page 440)	Structure that represents ISO9660 date and time
	_NAME_COLLISION_INFO (see page 441)	Structure that represents information about name collision
	_PQ_SUBCHANNEL (see page 443)	Structure that represents formatted PQ sub-channel
	_SCSI_DEVICE_ADDRESS (see page 445)	Structure that represents SCSI device address
	_STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS (see page 445)	Structure that represents advanced supported media formats
	_STARBURN_BDRE_FORMAT_PROFILE (see page 448)	Structure that represents format profile for BD-RE media
	_STARBURN_DISC_ATIP_INFORMATION (see page 449)	Structure that represents Disc ATIP information
	_STARBURN_DISC_INFORMATION (see page 451)	Structure that represents Disc information
	_STARBURN_ISO9660_DIRECTORY_INFO (see page 453)	This is record _STARBURN_ISO9660_DIRECTORY_INFO.
	_STARBURN_ISO9660_FILE_INFO (see page 453)	ISO9660 file information
	_STARBURN_STARWAVE2_INIT_PARAMS (see page 457)	Structure that represents initialization parameters for audio conversion
	_STARBURN_TRACK_INFORMATION (see page 458)	Structure that represents Track information
	_STARBURN_TRACK_INFORMATION_EX (see page 459)	Structure that represents Track information extended
	_STARBURN_UDF2_DIRECTORY_INFO (see page 461)	UDF directory information
	_STARBURN_UDF2_FILE_DATE_TIME (see page 461)	Structure that represents UDF2 file date and time

	_STARBURN_UDF2_FILE_INFO (see page 463)	UDF file information
	_STARPORT_DEVICE_LIST (see page 463)	Structure that represents StarPort device list
	_STARPORT_DEVICE_LIST_ENTRY (see page 464)	Structure that represents StarPort device list entry
	_STARWAVE2_COMPRESSION_PROFILE (see page 466)	
	_TOC_ENTRY (see page 466)	Structure that represents TOC entry
	_TOC_INFORMATION (see page 468)	Structure that represents TOC information
	_UDF_CONTROL_BLOCK (see page 469)	Structure that represents UDF control block
	_UDF_FILE_EXTENT (see page 469)	Structure that describes a single file extent
	_UDF_FILE_HANDLE (see page 470)	Structure that represents UDF file handle
	_UDF_FILE_LOOKUP_ENTRY (see page 470)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function
	_UDF_LOOKUP_DIR_ENTRY (see page 471)	Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	_UDF_LOOKUP_FILE_ENTRY (see page 471)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	_UDF_TREE_ITEM (see page 472)	Structure that represents UDF tree item
	_WAVE_FILE_HEADER (see page 474)	Structure that represents WAVE file header
	_WAVE_FORMAT_CHUNK (see page 475)	Structure that represents WAVE format chunk
	CDB_FAILURE_INFORMATION (see page 480)	Structure that represents CDB failure information
	DAO_DISC_LAYOUT (see page 480)	Structure that represents DAO disc layout
	DAO_DISC_LAYOUT_ENTRY (see page 481)	Structure that represents DAO disc layout entry (track)
	DISC_FILESYSTEM (see page 482)	Structure with disc file system flags
	DISC_LAYOUT (see page 482)	Structure that represents disc layout
	DISC_LAYOUT_ENTRY (see page 483)	Structure that represents disc layout entry (track)
	DVD_VIDEO_CONTROL_BLOCK (see page 486)	Structure that represents DVD-Video control block and pointer to DVD-Video control block
	FULL_TOC_ENTRY_RAW (see page 491)	Structure that represents full TOC entry
	ISO9660_DATE_TIME (see page 492)	Structure that represents ISO9660 date and time
	NAME_COLLISION_INFO (see page 493)	Structure that represents information about name collision
	PCDB_FAILURE_INFORMATION (see page 498)	Structure that represents CDB failure information
	PDAO_DISC_LAYOUT (see page 499)	Structure that represents DAO disc layout
	PDAO_DISC_LAYOUT_ENTRY (see page 499)	Structure that represents DAO disc layout entry (track)
	PDISC_FILESYSTEM (see page 500)	Structure with disc file system flags
	PDISC_LAYOUT (see page 500)	Structure that represents disc layout
	PDISC_LAYOUT_ENTRY (see page 501)	Structure that represents disc layout entry (track)
	PDVD_VIDEO_CONTROL_BLOCK (see page 504)	Structure that represents DVD-Video control block and pointer to DVD-Video control block
	PFULL_TOC_ENTRY_RAW (see page 509)	Structure that represents full TOC entry
	PISO9660_DATE_TIME (see page 510)	Structure that represents ISO9660 date and time
	PNAME_COLLISION_INFO (see page 511)	Structure that represents information about name collision
	PPQ_SUBCHANNEL (see page 513)	Structure that represents formatted PQ sub-channel
	PPSTARBURN_BDRE_FORMAT_PROFILE (see page 514)	Structure that represents format profile for BD-RE media
	PPSTARPORT_DEVICE_LIST (see page 517)	Structure that represents StarPort device list
	PPSTARPORT_DEVICE_LIST_ENTRY (see page 517)	Structure that represents StarPort device list entry
	PPSTARWAVE2_COMPRESSION_PROFILE (see page 519)	
	PQ_SUBCHANNEL (see page 520)	Structure that represents formatted PQ sub-channel
	PSCSI_DEVICE_ADDRESS (see page 522)	Structure that represents SCSI device address
	PSTARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS (see page 523)	Structure that represents advanced supported media formats
	PSTARBURN_BDRE_FORMAT_PROFILE (see page 525)	Structure that represents format profile for BD-RE media
	PSTARBURN_DISC_ATIP_INFORMATION (see page 527)	Structure that represents Disc ATIP information
	PSTARBURN_DISC_INFORMATION (see page 528)	Structure that represents Disc information
	PSTARBURN_STARWAVE2_INIT_PARAMS (see page 532)	Structure that represents initialization parameters for audio conversion
	PSTARBURN_TRACK_INFORMATION (see page 533)	Structure that represents Track information
	PSTARBURN_TRACK_INFORMATION_EX (see page 534)	Structure that represents Track information extended
	PSTARPORT_DEVICE_LIST (see page 536)	Structure that represents StarPort device list
	PSTARPORT_DEVICE_LIST_ENTRY (see page 536)	Structure that represents StarPort device list entry
	PSTARWAVE2_COMPRESSION_PROFILE (see page 539)	

	PTOC_ENTRY (see page 539)	Structure that represents TOC entry
	PTOC_INFORMATION (see page 541)	Structure that represents TOC information
	PUDF_CONTROL_BLOCK (see page 542)	Structure that represents UDF control block
	PUDF_FILE_EXTENT (see page 542)	Structure that describes a single file extent
	PUDF_FILE_HANDLE (see page 543)	Structure that represents UDF file handle
	PUDF_FILE_LOOKUP_ENTRY (see page 543)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function
	PUDF_LOOKUP_DIR_ENTRY (see page 544)	Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	PUDF_LOOKUP_FILE_ENTRY (see page 544)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	PUDF_TREE_ITEM (see page 545)	Structure that represents UDF tree item
	PWAVE_FILE_HEADER (see page 546)	Structure that represents WAVE file header
	PWAVE_FORMAT_CHUNK (see page 547)	Structure that represents WAVE format chunk
	SCSI_DEVICE_ADDRESS (see page 549)	Structure that represents SCSI device address
	STARBURN_ADVANCED_SUPPORTED_MEDIA_FORMATS (see page 550)	Structure that represents advanced supported media formats
	STARBURN_BDRE_FORMAT_PROFILE (see page 553)	Structure that represents format profile for BD-RE media
	STARBURN_DISC_ATIP_INFORMATION (see page 554)	Structure that represents Disc ATIP information
	STARBURN_DISC_INFORMATION (see page 555)	Structure that represents Disc information
	STARBURN_ISO9660_DIRECTORY_INFO (see page 558)	This is type STARBURN_ISO9660_DIRECTORY_INFO.
	STARBURN_ISO9660_FILE_INFO (see page 558)	ISO9660 file information
	STARBURN_STARWAVE2_INIT_PARAMS (see page 562)	Structure that represents initialization parameters for audio conversion
	STARBURN_TRACK_INFORMATION (see page 563)	Structure that represents Track information
	STARBURN_TRACK_INFORMATION_EX (see page 564)	Structure that represents Track information extended
	STARBURN_UDF2_DIRECTORY_INFO (see page 566)	UDF directory information
	STARBURN_UDF2_FILE_DATE_TIME (see page 566)	Structure that represents UDF2 file date and time
	STARBURN_UDF2_FILE_INFO (see page 567)	UDF file information
	STARPORT_DEVICE_LIST (see page 568)	Structure that represents StarPort device list
	STARPORT_DEVICE_LIST_ENTRY (see page 568)	Structure that represents StarPort device list entry
	STARWAVE2_COMPRESSION_PROFILE (see page 571)	
	TOC_ENTRY (see page 571)	Structure that represents TOC entry
	TOC_INFORMATION (see page 573)	Structure that represents TOC information
	UDF_CONTROL_BLOCK (see page 574)	Structure that represents UDF control block
	UDF_FILE_EXTENT (see page 574)	Structure that describes a single file extent
	UDF_FILE_HANDLE (see page 575)	Structure that represents UDF file handle
	UDF_FILE_LOOKUP_ENTRY (see page 575)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookup (see page 192) function
	UDF_LOOKUP_DIR_ENTRY (see page 576)	Structure with directory entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	UDF_LOOKUP_FILE_ENTRY (see page 576)	Structure with file entry in callback from StarBurn_CdvdBurnerGrabber_UDFFileSystemLookupEx (see page 193) function
	UDF_TREE_ITEM (see page 577)	Structure that represents UDF tree item
	WAVE_FILE_HEADER (see page 579)	Structure that represents WAVE file header
	WAVE_FORMAT_CHUNK (see page 580)	Structure that represents WAVE format chunk

Types

Name	Description
CFuncStarWaveConversionCallback (see page 582)	type of callback function
PCALLBACK (see page 582)	Callback type (function of this type must be passed as callback)
PSTARBURN_CLOSEHANDLE_CALLBACK (see page 584)	This is type PSTARBURN_CLOSEHANDLE_CALLBACK.
PSTARBURN_CREATEFILE_CALLBACK (see page 584)	This is type PSTARBURN_CREATEFILE_CALLBACK.
PSTARBURN_DELETEFILE_CALLBACK (see page 585)	This is type PSTARBURN_DELETEFILE_CALLBACK.
PSTARBURN_FILE_OBJECT (see page 585)	Structure that represents StarBurn file object used internally with file I/O
PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK (see page 585)	This is type PSTARBURN_FLUSH_FILE_BUFFERS_CALLBACK.
PSTARBURN_ISO2_COLLISION_CALLBACK (see page 585)	StarBurn ISO name collision handling callback
PSTARBURN_ISO2_PROGRESS_CALLBACK (see page 586)	StarBurn ISO progress callback

PSTARBURN_ISO2_UNICODE_COLLISION_CALLBACK (see page 586)	StarBurn ISO Unicode name collision handling callback
PSTARBURN_ISO2_UNICODE_PROGRESS_CALLBACK (see page 587)	StarBurn ISO UNICODE progress callback
PSTARBURN_ISO9660_READ_CALLBACK (see page 587)	StarBurn ISO2 read callback (for import)
PSTARBURN_READFILE_CALLBACK (see page 588)	This is type PSTARBURN_READFILE_CALLBACK.
PSTARBURN_SET_FILE_POINTER_CALLBACK (see page 588)	This is type PSTARBURN_SET_FILE_POINTER_CALLBACK.
PSTARBURN_STARWAVE_CALLBACK (see page 588)	StarWave callback passed to I/O functions. Return anything except zero from it to cancel I/O operation
PSTARBURN_UDF2_COLLISION_CALLBACK_EX (see page 589)	StarBurn UDF ansi name collision handling callback
PSTARBURN_UDF2_IMPORT_CALLBACK (see page 589)	StarBurn UDF2 import callback
PSTARBURN_UDF2_PROGRESS_CALLBACK (see page 590)	StarBurn UDF2 progress callback
PSTARBURN_UDF2_UNICODE_COLLISION_CALLBACK_EX (see page 590)	StarBurn UDF Unicode name collision handling callback
PSTARBURN_UDF2_UNICODE_PROGRESS_CALLBACK (see page 591)	StarBurn UDF2 UNICODE progress callback
PSTARBURN_WRITEFILE_CALLBACK (see page 591)	This is type PSTARBURN_WRITEFILE_CALLBACK.

Index

A

B

C

D

E

F

H

I

K

L

M

N

P

R

S

T

U

W