Magic CD/DVD Burner (C/C++ Library) v1.0.0

Magic CD/DVD Burner is developed to fulfill the requirement of all kinds of Data CD/DVD burning needs.

Magic CD/DVD Burner includes all the features required by the developers in such software plus more, offering the reliability, stability and ease of use in extremely affordable price.

Highlights:

- Add Full Featured Data CD/DVD recording functionality to your programs.
- Built-in ASPI Layer for Windows NT/2000/XP/2003.
- True Unicode support.
- Supports CD-R/CD-RW/DVD-R/DVD-RW/DVD-RAM/DVD+RW/DVD+R.
- On the Fly recording; No need to build ISO file first. No disk space is required for temporary files.
- Additional features for Building and Burning ISO images.
- Create ISO compatible (100% DOS/Windows) CD/DVD.
- Supports Joliet (Long File Name) File System.
- Create Multi-Session CD.
- Import any of the available previous sessions.
- Burn-Proof/Just Link support.
- Bootable CD/DVD support.
- High Performance File/Disc Caching.
- Highly customizable File/Directory layout.
- All Windows Versions (except Windows 3.1x) are supported.
- Tested with SCSI, IDE, EIDE, USB and Firewire CD/DVD Writers.
- Supports high write speeds.
- Supports use of Rewriteable CDs/DVDs.
- Can be used to make backup software.
- Easy to use.
- Free Life time updates.
- Source Code (in C/C++) is also available.

Registration

As a Registered User, you will have the following benefits:

- Free Life Time Updates, Upgrades and Bug Fixes.
- Availability of Latest CD/DVD Burners, being added daily.
- Free life time e-mail and online support.
- The royalty-free right to use code in your end-user applications.
- Eligible for special discounts on volume and other Binary Magic products.
- High priority on suggestions to be made in the software.

SINGLE USER SOFTWARE LICENSE AND LIMITED WARRANTY

As a proprietary product of Binary Magic, Magic CD/DVD Burner is protected by copyright laws. At all times Binary Magic retains full title to the software. Subject to your acceptance of and accordance with the terms and conditions stated in this agreement, you shall be granted a single-user software license. Any previous agreement with Binary Magic is superseded by this agreement.

THIS SOFTWARE LICENSE GIVES YOU THE RIGHT TO:

1. Install and use the Software for the sole purposes of designing, developing, testing, and deploying application programs which you create. You may install a copy of the Software on a computer and freely move the Software from one computer to another, provided that you are the only individual using the Software. If you are an entity, you must designate one individual within your organization ("Named User") to have the right to use the Software.
2. Write and compile your own application programs using the Binary Magic software contained in this package. All copies of the software you so write and distribute must include a Binary Magic copyright notice, or a valid copyright notice of your own.
3. Make one copy of the Software for backup or archival purposes or copy the Software to a single permanent storage medium provided you keep the original solely for backup or archival purposes.
4. Distribute runtime library for the sole purpose of executing application programs created with any Programming Language. Magic CD/DVD Burner runtime library available for distribution are listed in the ReadMe.doc file that is installed into the main Magic CD/DVD Burner directory.

ENGAGING IN ANY OF THE ACTIVITIES LISTED BELOW WILL TERMINATE THE SOFTWARE LICENSE. IN ADDITION TO SOFTWARE LICENSE TERMINATION, BINARY MAGIC MAY PURSUE CRIMINAL, CIVIL, OR ANY OTHER AVAILABLE REMEDIES.

1. Distribution of any files contained in this software package, other than the runtime packages explicitly listed above, including but not limited to source code files, and design-time packages.
2. Modification, de-compilation, disassembly, reverse engineering or translation of the Software.
3. Removal of proprietary notices, labels or marks from the Software or Software Documentation.
4. Inclusion of the software in a development environment.
5. Creation of an application that does not differ materially from the Software.
6. Creation of an application (whether it will be freeware, shareware or a commercial product) which competes directly or indirectly with our software Magic CD/DVD Burner.
7. Distribution of an application program created using the Software to another developer. A developer is defined as any person who is executing an application program created using the Software, on a computer which contains an installation of any programming language. In order to execute such an application, the developer must own a license to the Software, and must have installed the Software on the computer.
8. You may not use Magic CD/DVD Burner to create components or controls to be used by other developers without written approval from Binary Magic.

AGREEMENT PERTAINING TO THE RELEASE OF SOURCE CODE by Binary Magic to Recipient:

USE OF SOURCE CODE

Recipient will not utilize the source for the creation of software (whether it is freeware, shareware or a commercial product) which competes directly or indirectly with Magic CD/DVD Burner. In addition, Recipient will not disclose the source itself, nor the implementations discovered therein, to any party involved in the creation of software which competes directly or indirectly with Magic CD/DVD Burner.

DISTRIBUTION OF SOURCE CODE

Recipient will not distribute the source. Specifically this includes all the source files which Binary Magic has provided.

CHANGES TO SOURCE CODE

Binary Magic reserves the right to change any part of the source in future versions of the product. These changes may include the removal of classes, properties and methods or the creation of new classes, properties and methods.

TECHNICAL SUPPORT FOR SOURCE CODE

Binary Magic will not provide support for changes recipient makes to the source. Recipient assumes full responsibility for supporting any code or application which results from such modification. Recipient will not hold Binary Magic liable, directly or indirectly, for any changes made to the source, including changes which Recipient has made based on advice or suggestions provided by Binary Magic.

SOURCE IS PROVIDED AS IS

Binary Magic makes no warranties, express or implied, with respect to the source and hereby expressly disclaims any and all implied warranties of merchantability and fitness for a particular purpose. In no event shall Binary Magic be liable for any direct, indirect, special, or consequential damages in connection with or arising out of the performance or use of any portion of the source.

LIMITED WARRANTY/LIMITATION OF REMEMDIES:

Except as specifically stated in this agreement, the software and software documentation is provided and licensed "as is" without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

In no event will Binary Magic be liable to you for any damages, including without limitation lost profits or revenues, loss of data, business interruption loss, recovery or substitution costs, or claims by third parties, or other indirect incidental or consequential damages, arising out of the use or inability to use the software, even if Binary Magic has been advised of the possibility of such damages. In no case shall Binary Magic' liability exceed the amount of the license fee paid by you for the software.

Installation

Automatic installer/Uninstaller is included.

CMCDBurner Functions

Module
MCDBCls

class
CMCDBurner

Errors

Error Number	Description (Message Text)
0	NO ERROR
1	DATA NOT PREPARED FOR BURNING/WRITING
2	CAN'T OPEN FILE FOR READ
3	CAN'T CREATE FILE
4	ABORTED BY USER
512	NO SEEK COMPLETE
1024	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
1025	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
1026	LOGICAL UNIT NOT READY, INITIALIZING CMD. REQUIRED
1027	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
1028	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
1031	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS
1032	LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS
1033	LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS
1280	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
1536	NO REFERENCE POSITION FOUND
1792	MULTIPLE PERIPHERAL DEVICES SELECTED
2048	LOGICAL UNIT COMMUNICATION FAILURE
2049	LOGICAL UNIT COMMUNICATION TIME-OUT
2050	LOGICAL UNIT COMMUNICATION PARITY ERROR
2051	LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)
2052	UNREACHABLE COPY TARGET
2304	TRACK FOLLOWING ERROR
2305	TRACKING SERVO FAILURE
2306	FOCUS SERVO FAILURE
2307	SPINDLE SERVO FAILURE
2308	HEAD SELECT FAULT
2560	ERROR LOG OVERFLOW
2816	WARNING
2817	WARNING - SPECIFIED TEMPERATURE EXCEEDED
2818	WARNING - ENCLOSURE DEGRADED
3072	WRITE ERROR
3079	WRITE ERROR - RECOVERY NEEDED
3080	WRITE ERROR - RECOVERY FAILED
3081	WRITE ERROR - LOSS OF STREAMING
3082	WRITE ERROR - PADDING BLOCKS ADDED
3328	ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR
3329	THIRD PARTY DEVICE FAILURE
3330	COPY TARGET DEVICE NOT REACHABLE
3331	INCORRECT COPY TARGET DEVICE TYPE
3332	COPY TARGET DEVICE DATA UNDERRUN
3333	COPY TARGET DEVICE DATA OVERRUN
4352	UNRECOVERED READ ERROR
4353	READ RETRIES EXHAUSTED
4354	ERROR TOO LONG TO CORRECT
4357	L-EC UNCORRECTABLE ERROR
4358	CIRC UNRECOVERED ERROR
4365	DE-COMPRESSION CRC ERROR
4366	CANNOT DECOMPRESS USING DECLARED ALGORITHM

4367	ERROR READING UPC/EAN NUMBER
4368	ERROR READING ISRC NUMBER
4369	READ ERROR - LOSS OF STREAMING
5120	RECORDED ENTITY NOT FOUND
5121	RECORD NOT FOUND
5376	RANDOM POSITIONING ERROR
5377	MECHANICAL POSITIONING ERROR
5378	POSITIONING ERROR DETECTED BY READ OF MEDIUM
5888	RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
5889	RECOVERED DATA WITH RETRIES
5890	RECOVERED DATA WITH POSITIVE HEAD OFFSET
5891	RECOVERED DATA WITH NEGATIVE HEAD OFFSET
5892	RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
5893	RECOVERED DATA USING PREVIOUS SECTOR ID
5895	RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
5896	RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
5897	RECOVERED DATA WITHOUT ECC - DATA REWRITTEN
6144	RECOVERED DATA WITH ERROR CORRECTION APPLIED
6145	RECOVERED DATA WITH ERROR CORR. & RETRIES APPLIED
6146	RECOVERED DATA - DATA AUTO-REALLOCATED
6147	RECOVERED DATA WITH CIRC
6148	RECOVERED DATA WITH L-EC
6149	RECOVERED DATA - RECOMMEND REASSIGNMENT
6150	RECOVERED DATA - RECOMMEND REWRITE
6152	RECOVERED DATA WITH LINKING
6656	PARAMETER LIST LENGTH ERROR
6912	SYNCHRONOUS DATA TRANSFER ERROR
7424	MISCOMPARE DURING VERIFY OPERATION
8192	INVALID COMMAND OPERATION CODE
8448	LOGICAL BLOCK ADDRESS OUT OF RANGE
8449	INVALID ELEMENT ADDRESS
8450	INVALID ADDRESS FOR WRITE
9216	INVALID FIELD IN CDB
9217	CDB DECRYPTION ERROR
9472	LOGICAL UNIT NOT SUPPORTED
9728	INVALID FIELD IN PARAMETER LIST
9729	PARAMETER NOT SUPPORTED
9730	PARAMETER VALUE INVALID
9731	THRESHOLD PARAMETERS NOT SUPPORTED
9732	INVALID RELEASE OF PERSISTENT RESERVATION
9733	DATA DECRYPTION ERROR
9734	TOO MANY TARGET DESCRIPTORS
9735	UNSUPPORTED TARGET DESCRIPTOR TYPE CODE
9736	TOO MANY SEGMENT DESCRIPTORS
9737	UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE
9738	UNEXPECTED INEXACT SEGMENT
9739	INLINE DATA LENGTH EXCEEDED
9740	INVALID OPERATION FOR COPY SOURCE OR DESTINATION
9741	COPY SEGMENT GRANULARITY VIOLATION
9984	WRITE PROTECTED
9985	HARDWARE WRITE PROTECTED

9986	LOGICAL UNIT SOFTWARE WRITE PROTECTED
9987	ASSOCIATED WRITE PROTECT
9988	PERSISTENT WRITE PROTECT
9989	PERMANENT WRITE PROTECT
9990	CONDITIONAL WRITE PROTECT
10240	NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED
10241	IMPORT OR EXPORT ELEMENT ACCESSED
10496	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED
10497	POWER ON OCCURRED
10498	SCSI BUS RESET OCCURRED
10499	BUS DEVICE RESET FUNCTION OCCURRED
10500	DEVICE INTERNAL RESET
10501	TRANSCEIVER MODE CHANGED TO SINGLE-ENDED
10502	TRANSCEIVER MODE CHANGED TO LVD
10752	PARAMETERS CHANGED
10753	MODE PARAMETERS CHANGED
10754	LOG PARAMETERS CHANGED
10755	RESERVATIONS PREEMPTED
10756	RESERVATIONS RELEASED
10757	REGISTRATIONS PREEMPTED
11008	COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT
11264	COMMAND SEQUENCE ERROR
11267	CURRENT PROGRAM AREA IS NOT EMPTY
11268	CURRENT PROGRAM AREA IS EMPTY
11270	PERSISTENT PREVENT CONFLICT
11776	INSUFFICIENT TIME FOR OPERATION
12032	COMMANDS CLEARED BY ANOTHER INITIATOR
12288	INCOMPATIBLE MEDIUM INSTALLED
12289	CANNOT READ MEDIUM - UNKNOWN FORMAT
12290	CANNOT READ MEDIUM - INCOMPATIBLE FORMAT
12291	CLEANING CARTRIDGE INSTALLED
12292	CANNOT WRITE MEDIUM - UNKNOWN FORMAT
12293	CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT
12294	CANNOT FORMAT MEDIUM - INCOMPATIBLE MEDIUM
12295	CLEANING FAILURE
12296	CANNOT WRITE - APPLICATION CODE MISMATCH
12297	CURRENT SESSION NOT FIXATED FOR APPEND
12304	MEDIUM NOT FORMATTED
12544	MEDIUM FORMAT CORRUPTED
12545	FORMAT COMMAND FAILED
12546	ZONED FORMATTING FAILED DUE TO SPARE LINKING
13312	ENCLOSURE FAILURE
13568	ENCLOSURE SERVICES FAILURE
13569	UNSUPPORTED ENCLOSURE FUNCTION
13570	ENCLOSURE SERVICES UNAVAILABLE
13571	ENCLOSURE SERVICES TRANSFER FAILURE
13572	ENCLOSURE SERVICES TRANSFER REFUSED
14080	ROUNDED PARAMETER
14592	SAVING PARAMETERS NOT SUPPORTED
14848	MEDIUM NOT PRESENT
14849	MEDIUM NOT PRESENT - TRAY CLOSED
14850	MEDIUM NOT PRESENT - TRAY OPEN
14851	MEDIUM NOT PRESENT - LOADABLE
14852	MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY

ACCESSIBLE	
15117	MEDIUM DESTINATION ELEMENT FULL
15118	MEDIUM SOURCE ELEMENT EMPTY
15119	END OF MEDIUM REACHED
15121	MEDIUM MAGAZINE NOT ACCESSIBLE
15122	MEDIUM MAGAZINE REMOVED
15123	MEDIUM MAGAZINE INSERTED
15124	MEDIUM MAGAZINE LOCKED
15125	MEDIUM MAGAZINE UNLOCKED
15126	MECHANICAL POSITIONING OR CHANGER ERROR
15616	INVALID BITS IN IDENTIFY MESSAGE
15872	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET
15873	LOGICAL UNIT FAILURE
15874	TIMEOUT ON LOGICAL UNIT
15875	LOGICAL UNIT FAILED SELF-TEST
15876	LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG
16128	TARGET OPERATING CONDITIONS HAVE CHANGED
16129	MICROCODE HAS BEEN CHANGED
16130	CHANGED OPERATING DEFINITION
16131	INQUIRY DATA HAS CHANGED
16132	COMPONENT DEVICE ATTACHED
16133	DEVICE IDENTIFIER CHANGED
16134	REDUNDANCY GROUP CREATED OR MODIFIED
16135	REDUNDANCY GROUP DELETED
16136	SPARE CREATED OR MODIFIED
16137	SPARE DELETED
16138	VOLUME SET CREATED OR MODIFIED
16139	VOLUME SET DELETED
16140	VOLUME SET DEASSIGNED
16141	VOLUME SET REASSIGNED
16142	REPORTED LUNS DATA HAS CHANGED
16143	ECHO BUFFER OVERWRITTEN
16144	MEDIUM LOADABLE
16145	MEDIUM AUXILIARY MEMORY ACCESSIBLE
17152	MESSAGE ERROR
17408	INTERNAL TARGET FAILURE
17664	SELECT OR RESELECT FAILURE
17920	UNSUCCESSFUL SOFT RESET
18176	SCSI PARITY ERROR
18177	DATA PHASE CRC ERROR DETECTED
18178	SCSI PARITY ERROR DETECTED DURING ST DATA PHASE
18179	INFORMATION UNIT CRC ERROR DETECTED
18180	ASYNCHRONOUS INFORMATION PROTECTION ERROR
DETECTED	
18432	INITIATOR DETECTED ERROR MESSAGE RECEIVED
18688	INVALID MESSAGE ERROR
18944	COMMAND PHASE ERROR
19200	DATA PHASE ERROR
19456	LOGICAL UNIT FAILED SELF-CONFIGURATION
19968	OVERLAPPED COMMANDS ATTEMPTED
20736	ERASE FAILURE
20737	ERASE FAILURE - INCOMPLETE ERASE OPERATION
DETECTED	
21248	MEDIA LOAD OR EJECT FAILED
21250	MEDIUM REMOVAL PREVENTED
21762	INSUFFICIENT RESERVATION RESOURCES

21763	INSUFFICIENT RESOURCES
21764	INSUFFICIENT REGISTRATION RESOURCES
22272	UNABLE TO RECOVER TABLE-OF-CONTENTS
23040	OPERATOR REQUEST OR STATE CHANGE INPUT
23041	OPERATOR MEDIUM REMOVAL REQUEST
23042	OPERATOR SELECTED WRITE PROTECT
23043	OPERATOR SELECTED WRITE PERMIT
23296	LOG EXCEPTION
23297	THRESHOLD CONDITION MET
23298	LOG COUNTER AT MAXIMUM
23299	LOG LIST CODES EXHAUSTED
23808	FAILURE PREDICTION THRESHOLD EXCEEDED
23809	MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED
23810	LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED
23811	SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED
24063	FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)
24064	LOW POWER CONDITION ON
24065	IDLE CONDITION ACTIVATED BY TIMER
24066	STANDBY CONDITION ACTIVATED BY TIMER
24067	IDLE CONDITION ACTIVATED BY COMMAND
24068	STANDBY CONDITION ACTIVATED BY COMMAND
25344	END OF USER AREA ENCOUNTERED ON THIS TRACK
25345	PACKET DOES NOT FIT IN AVAILABLE SPACE
25600	ILLEGAL MODE FOR THIS TRACK
25601	INVALID PACKET SIZE
25856	VOLTAGE FAULT
28416	COPY PROTECTION KEY EXCHANGE FAILURE - AUTHENTICATION FAILURE
28417	COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT PRESENT
28418	COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT ESTABLISHED
28419	READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION
28420	MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION
28421	DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR
29184	SESSION FIXATION ERROR
29185	SESSION FIXATION ERROR WRITING LEAD-IN
29186	SESSION FIXATION ERROR WRITING LEAD-OUT
29187	SESSION FIXATION ERROR - INCOMPLETE TRACK IN SESSION
29188	EMPTY OR PARTIALLY WRITTEN RESERVED TRACK
29189	NO MORE TRACK RESERVATIONS ALLOWED
29440	CD CONTROL ERROR
29441	POWER CALIBRATION AREA ALMOST FULL
29442	POWER CALIBRATION AREA IS FULL
29443	POWER CALIBRATION AREA ERROR
29444	PROGRAM MEMORY AREA UPDATE FAILURE
29445	PROGRAM MEMORY AREA IS FULL
29446	RMA/PMA IS ALMOST FULL

CMCDBurner Functions

CD/DVD Burning

AbortBurn
BuildISOImage
Burn
BurnISOImage
GetBlocksWritten
GetBurnProof
GetCachePosition
GetCacheSize
GetImageSize
GetPerformOPC
Prepare
SetBurnProof
SetCacheSize
SetPerformOPC

Device Related

DeviceCan
DeviceIsBurner
EjectMedium
GetComponentState
GetDeviceBufferPosition
GetDeviceBufferSize
GetDeviceCount
GetDeviceMaxReadSpeed
GetDeviceMaxWriteSpeed
GetDeviceName
GetReadSpeed
GetTestWrite
GetWriteSpeed
LoadMedium
LockMedium
SelectDevice
SetReadSpeed
SetTestWrite
SetWriteSpeed
TestUnitReady
UnlockMedium

Disc Layout Manipulation

ClearAll
SetReplaceFile
CreateDir*
ImportSession*
InsertDir*
InsertFile*
InsertFileWithName*
SetVolumeID*
GetVolumeID*

* Unicode version of these functions are also available

Media Related

DiscIs
Erasable
EraseDisc
GetFinalizeDisc
GetFreeBlocksOnDisc
GetMaxWriteSpeed
GetTotalBlocksOnDisc
SessionsOnDisc
SetFinalizeDisc
Writable

Miscellaneous

GetMCDBVersion
GetErrorNumber

AbortBurn

CMCDBurner **See Also: Burn, BurnISOImage**
Cancels Burn operation.

C Syntax:
```
void AbortBurn(FILES *cdfiles)
```

C++ Syntax:
```
void AbortBurn(void)
```

This *method* is used to abort the on-going write operation. It instantly stops writing data to the disc and leaves the disc as-is. This method should be used with care, as the disc will be left in an **open** state.

Note: This method will **not** abort an EraseDisc operation.

BuildISOImage

CMCDBurner **See Also: Prepare, BurnISOImage**
Builds ISO image file.

C Syntax:
```
BOOL BuildISOImage(FILES *cdfiles, char *FileName, void (__stdcall
*__writedone)(int, void *), void *arg)
```

C++ Syntax:
```
BOOL BuildISOImage(char *ISOFileName, void (__stdcall *__writedone)(BOOL,
void *), void *)
```

This *method* is used to build ISO image file from the data, added to layout. *ISOFileName* is the name of ISO image file, to be built.

writedone method is triggered after the write operation is completed. It is invoked irrespective of status of the write operation, therefore, it is also invoked even if the write operation is failed.

Return Value: Return value is *TRUE* if ISO image file is built successfully otherwise a *FALSE* value is return.

Burn

CMCDBurner **See Also: InsertDir, InsertFile, Prepare, BurnISOImage, GetWriteSpeed, SetWriteSpeed**

Starts the process of burning files/directories on media.

C Syntax:

```
BOOL Burn(FILES *cdfiles, void (__stdcall *__writedone)(int, void * ), void
*arg)
```

C++ Syntax:

```
BOOL Burn(void (__stdcall *__writedone)(BOOL, void *), void *)
```

This *method* begins the disc writing operation. Files/Directories, to be burned on a media, should be added to disc layout, prior to calling this *method*.

writedone method is triggered after the write operation is completed. It is invoked irrespective of status of the write operation, therefore, it is also invoked even if the write operation is failed.

Return Value: Return value is *FALSE* if an error occurs before writing begins. The return value is *TRUE* if the writing begins successfully. This operation is asynchronous i.e. this method will return a *TRUE* or a *FALSE* immediately, as soon as the writing process is initiated.

BurnISOImage

CMCDBurner **See Also: InsertDir, InsertFile, Prepare, Burn, GetWriteSpeed, SetWriteSpeed**

Starts the process of burning ISO image on media.

C Syntax:

```
BOOL BurnISOImage(FILES *cdfiles, char *FileName, void (__stdcall
*__writedone)(int, void *), void *arg)
```

C++ Syntax:

```
BOOL BurnISOImage(char *FileName, void (__stdcall *__writedone)(BOOL, void *
), void *)
```

This *method* begins the disc writing operation. *FileName* is the name of ISO image file, to be burned. ISO image file, to be burned on a media, should be built, prior to calling this method.

writedone method is triggered after the write operation is completed. It is invoked irrespective of status of the write operation, therefore, it is also invoked even if the write operation is failed.

Return Value: Return value is *FALSE* if an error occurs before writing begins. The return value is *TRUE* if the writing begins successfully. This operation is asynchronous i.e. this method will return a *TRUE* or a *FALSE* immediately, as soon as the writing process is initiated.

ClearAll

CMCDBurner **See Also: EraseDisc**
Clear all files and directories from disc layout.

C Syntax:
```
void ClearAll(FILES *cdfiles)
```

C++ Syntax:
```
void ClearAll(void)
```

This *method* clears all the added files and directories from the disc layout.

CreateDir

CMCDBurner **See Also: InsertDir, InsertFile**
Creates a directory in disc layout.

C Syntax:
```
DirEntry *CreateDir(FILES *cdfiles, char *dest_path, char *dir_name)
```

C++ Syntax:
MultiByte Character System version:
```
DirEntry *CreateDir(char *dest_path, char *dir_name)
```
Unicode version:
```
DirEntry *CreateDirW(wchar_t *dest_path, wchar_t *dir_name)
```

This *method* is used to create a new directory, in the disc layout, at the specified path. *dest_path* specifies the path where the directory will be created and *dir_name* is the name of directory, to be created.

Return Value: If the directory is created successfully then a pointer to the directory structure (i.e. *DirEntry*) is returned otherwise a *NULL* is returned.

DeviceCan

CMCDBurner **See Also: DeviceIsBurner, SelectDevice**
Validates the selected device as capable of performing an operation.

C Syntax:
```
BOOL DeviceCan(SCSIDEVICE *device, BYTE cId)
```

C++ Syntax:
```
BOOL DeviceCan(BYTE cId)
```

This *method* is used to validate that the currently selected device is capable of performaing an operation, specified by *cId*. The possible values of *cId* are as follows:

```
1    DC_READ_CDR
2    DC_READ_CDRW
3    DC_READ_DVDRAM
4          DC_READ_DVDR
5    DC_READ_DVDRW
6    DC_READ_DVDPLUSR
7    DC_READ_DVDPLUSRW

11         DC_WRITE_CDR
12         DC_WRITE_CDRW
13         DC_WRITE_DVDRAM
14         DC_WRITE_DVDR
15         DC_WRITE_DVDRW
16         DC_WRITE_DVDPLUSR
17         DC_WRITE_DVDPLUSRW
```

Return Value: Return value is *TRUE* if the device is capable of performing the specified operation otherwise a *FALSE* value is return.

DeviceIsBurner

CMCDBurner **See Also: SelectDevice, DeviceCan**
Validates the selected device as burner.

C Syntax:
```
BOOL DeviceIsBurner(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL DeviceIsBurner(void)
```

This *method* is used to validate that the selected device is capable of burning.

Return Value: Return value is *TRUE* if the selected device is capable of burning otherwise a *FALSE* value is return.

DiscIs

CMCDBurner
Returns the type of media inserted in the device.

C Syntax:
```
BYTE DiscIs(SCSIDEVICE *device)
```

C++ Syntax:
```
BYTE DiscIs(void)
```

This *method* is used to get the type of media, inserted in the currently selected device.

Return Value: Return value can be matched against one of the following values to determine the exact disc type:

0	UNKNOWN
1	CD-ROM
2	CD-R
3	CD-RW
4	DDCD-R
5	DDCD-RW
6	DVD-RAM
7	DVD-R
8	DVD-RW
9	DVD+R
10	DVD+RW

EjectMedium

CMCDBurner **See Also: LoadMedium, LockMedium, UnlockMedium, TestUnitReady**

Ejects media from the currently selected device.

C Syntax:

```
BOOL EjectMedium(SCSIDEVICE *device, char Immediate)
```

C++ Syntax:

```
BOOL EjectMedium(void)
```

This *method* is used to invoke an Eject operation of the currently selected device.

Return Value: Return value is *TRUE* if the currently selected device is ejected successfully otherwise a *FALSE* value is return.

Erasable

CMCDBurner **See Also: EraseDisc**
Validates the media as Erasable/Rewriteable.

C Syntax:
```
BOOL Erasable(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL Erasable(void)
```

This *method* detects that the media inserted in the selected device is Rewriteable/Erasable (CD-RW, DVD-RAM, DVD-RW or DVD+RW).

Return Value: Return value is *TRUE* if the inserted media is erasable otherwise a *FALSE* value is return.

EraseDisc

CMCDBurner **See Also: Erasable**
Starts the process of erasing media.

C Syntax:
```
BOOL EraseDisc(SCSIDEVICE *device, BOOL Quick, void ( __stdcall *__erasedone
)( int, void * ), void *ptr)
```

C++ Syntax:
```
BOOL EraseDisc(BOOL Quick, void ( __stdcall *__start )( int, void *), void
*arg)
```

This *method* starts erasing the media, inserted in the selected device. This *method* is used only for Rewriteable media (CD-RW, DVD-RAM, DVD-RW or DVD+RW). There are two choices available to this method; Complete and Quick. Pass *Quick* as *TRUE* to perform a quick erase operation or pass it as *FALSE* to perform a complete erase operation on media.

In Complete mode, the device is initialized fully (as was the case in the old floppy disks) and takes a long time.
In Quick mode, the device is erased quickly i.e., data is still on the media, just the File and Directory descriptor information is deleted.

start method is triggered after the erase operation is completed. It is invoked irrespective of status of the erase operation, therefore, it is also invoked even if the erase operation is failed.

Note: Almost all media available currently is completely erased once, therefore Quick method is enough to erase any data on the media.

Return Value: Return value is *TRUE* if the inserted media is erased successfully otherwise a *FALSE* value is return.

GetBlocksWritten

CMCDBurner **See Also: Burn, BurnISOImage**
Returns count of blocks actually written to media.

C Syntax:
```
DWORD GetBlocksWritten(FILES *cdfiles)
```

C++ Syntax:
```
DWORD GetBlocksWritten(void)
```

This *method* is used to get the number of blocks actually written to media, with a burn operation.

Return Value: Return value is the count of blocks, written to media.

GetBurnProof

CMCDBurner **See Also: SetBurnProof, Burn, BurnISOImage**
Returns the status of BURN-Proof flag.

C Syntax:
```
BOOL GetBurnProof(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL GetBurnProof(void)
```

This *method* is used to get the status of BURN-Proof (Buffer Under RuN error Proof) flag which helps in protection from BURN error.

Return Value: Return value is the current status of BURN-Proof flag.

GetCachePosition

CMCDBurner **See Also: GetCacheSize, SetCacheSize, GetDeviceBufferPosition**

Returns the software's internal cache buffer status.

C Syntax:

```
DWORD GetCachePosition(FILES *cdfiles)
```

C++ Syntax:

```
DWORD GetCachePosition(void)
```

This *method* is used to get the size of data (in bytes), contained in software's internal cache buffer, during a burn operation.

Return Value: Return value is the size of data present in software's internal cache buffer.

GetCacheSize

CMCDBurner **See Also: SetCacheSize, GetDeviceBufferSize, GetCachePosition**

Returns the size of software cache buffer.

C Syntax:

```
DWORD GetCacheSize(FILES *cdfiles)
```

C++ Syntax:

```
DWORD GetCacheSize(void)
```

This *method* is used to get the size of internal software cache buffer. The data, to be burned, is kept in the system RAM, before it is burned to the disc.

Return Value: Return value is the size of internal software cache buffer.

GetComponentState

CMCDBurner **See Also: Writable**
Returns the current state of library object.

C Syntax:
```
WORD GetComponentState(SCSIDEVICE *device)
```

C++ Syntax:
```
WORD GetComponentState(void)
```

This *method* is used to get the current status information of library object.

Return Value: Return value is the current status code of library object. The returned value can be matched against one of the following values to determine the exact status code:

```
0    CS_IDLE
1    CS_WRITING
2    CS_ERASING
3    CS_BACKGROUNDFORMAT
4    CS_CLOSINGTRACK
5    CS_PREPARING
6    CS_ABORTING
7          CS_IMPORTINGSESSION
9    CS_BURNERROR_ABORTING
```

GetDeviceBufferPosition

CMCDBurner **See Also: GetDeviceBufferSize, GetCachePosition**
Returns the device's internal cache buffer status.

C Syntax:
```
DWORD GetDeviceBufferPosition(SCSIDEVICE *device)
```

C++ Syntax:
```
DWORD GetDeviceBufferPosition(void)
```

This *method* is used to get the size of data (in bytes), contained in device's internal cache buffer, during a burn operation.

Return Value: Return value is the size of data, contained in internal cache buffer of currently selected device.

GetDeviceBufferSize

CMCDBurner **See Also: GetCacheSize, SetCacheSize, GetDeviceBufferPosition**
Returns the size of device cache buffer.

C Syntax:
```
DWORD GetDeviceBufferSize(SCSIDEVICE *device)
```

C++ Syntax:
```
DWORD GetDeviceBufferSize(void)
```

This *method* is used to get the size (in bytes) of internal cache buffer of currently selected device.

Return Value: Return value is the size of device cache buffer.

GetDeviceCount

CMCDBurner **See Also: GetDeviceName, DeviceIsBurner**
Returns count of devices (burners), attached to the system.

C Syntax:
```
BYTE GetDeviceCount(SCSIDEVICE *device)
```

C++ Syntax:
```
BYTE GetDeviceCount(void)
```

This *method* is used to get the total number of all available burning devices currently attached to the computer.

Return Value: Return value is the count of devices, attached to the system.

GetDeviceMaxReadSpeed

CMCDBurner **See Also: GetReadSpeed, SetReadSpeed**
Returns the maximum read speed of selected device.

C Syntax:
```
BYTE GetDeviceMaxReadSpeed(SCSIDEVICE *device)
```

C++ Syntax:
```
BYTE GetDeviceMaxReadSpeed(void)
```

This *method* is used to get the maximum physical read speed of the currently selected device.

Return Value: Return value is the maximum physical read speed of the currently selected device.

GetDeviceMaxWriteSpeed

CMCDBurner **See Also: GetWriteSpeed, SetWriteSpeed, GetMaxWriteSpeed**
Returns the maximum write speed of selected device.

C Syntax:
```
BYTE GetDeviceMaxWriteSpeed(SCSIDEVICE *device)
```

C++ Syntax:
```
BYTE GetDeviceMaxWriteSpeed(void)
```

This *method* is used to get the maximum physical write speed of the currently selected device.

Return Value: Return value is the maximum physical write speed of the currently selected device.

GetDeviceName

CMCDBurner **See Also: GetDeviceCount**
Returns the name of selected device.

C Syntax:
```
char *GetDeviceName(SCSIDEVICE *device, BYTE Id)
```

C++ Syntax:
```
char *GetDeviceName(BYTE Id)
```

This *method* returns the manufacturer's details of the device, which is identified by *Id*, including the manufacturer name, model number and firmware revision.

Return Value: Return value is the name of currently selected device.

GetFinalizeDisc

CMCDBurner **See Also: SetFinalizeDisc, GetFreeBlocksOnDisc, Burn, BurnISOImage, EraseDisc**

Returns the status of Finalize/Close Disc flag.

C Syntax:

```
BOOL GetFinalizeDisc(SCSIDEVICE *device)
```

C++ Syntax:

```
BOOL GetFinalizeDisc(void)
```

This *method* is used to determine the status of Finalize/Close Disc flag, set for media in currently selected device.

Return Value: Return value is the status of Finalize/Close Disc flag.

GetFreeBlocksOnDisc

CMCDBurner **See Also: SessionsOnDisc, GetTotalBlocksOnDisc, EraseDisc, GetFinalizeDisc, SetFinalizeDisc**
Returns the count of free blocks on media.

C Syntax:
```
DWORD GetFreeBlocksOnDisc(SCSIDEVICE *device)
```

C++ Syntax:
```
DWORD GetFreeBlocksOnDisc(void)
```

This *method* is used to get the total number of free blocks on the media inserted in the currently selected device.

Return Value: Return value is the count of free blocks on media. Return value is 0, if the disc is finalized.

Note: Block size is 2048 bytes.

GetImageSize

CMCDBurner **See Also: Prepare, GetFreeBlocksOnDisc**
Returns the total CD image size in blocks.

C Syntax:
```
DWORD GetImageSize(FILES *cdfiles)
```

C++ Syntax:
```
DWORD GetImageSize(void)
```

This *method* is used to get the actual image size, in terms of blocks (2048 bytes in each block). The image size is the size of data that will be written to the media or the size of the .ISO image file, if it is saved to the hard drive. The image size is only available after Prepare method has been invoked.

Return Value: Return value is the actual image size of data, added to disc layout so far.

GetMaxWriteSpeed

CMCDBurner **See Also: GetWriteSpeed, SetWriteSpeed, GetDeviceMaxWriteSpeed**

Returns the maximum write speed of inserted media.

C Syntax:

```
BYTE GetMaxWriteSpeed(SCSIDEVICE *device)
```

C++ Syntax:

```
BYTE GetMaxWriteSpeed(void)
```

This *method* is used to determine the maximum write speed of the media, inserted in the currently selected device. This speed is dependent on write speed of the device, in which the media is inserted.

Return Value: Return value is the maximum write speed of the media, inserted in the currently selected device.

GetMCDBVersion

CMCDBurner
Returns the version number of Magic CD Burner C/C++ library.

C++ Syntax:
```
char *GetMCDBVersion(void)
```

This *method* is used to get the version number of Magic CD Burner C/C++ library, being used.

Return Value: Return value is the version number of Magic CD Burner C/C++ library.

GetErrorNumber

CMCDBurner
Returns the last error number.

C++ Syntax:
```
DWORD GetErrorNumber(void)
```

This *method* is used to get the last error number of the physical device. The returned value is an *DWORD*.
It can be matched against one of the following values to determine the error:

```
    0 = "NO ERROR"
    1 = "DATA NOT PREPARED FOR BURNING/WRITING"
    2 = "CAN'T OPEN FILE FOR READ"
    3 = "CAN'T CREATE FILE"
    4 = "ABORTED BY USER"
    5 = "TOO MANY FILES"

  512 = "NO SEEK COMPLETE"
 1024 = "LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE"
 1025 = "LOGICAL UNIT IS IN PROCESS OF BECOMING READY"
 1026 = "LOGICAL UNIT NOT READY, INITIALIZING CMD. REQUIRED"
 1027 = "LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED"
 1028 = "LOGICAL UNIT NOT READY, FORMAT IN PROGRESS"
 1031 = "LOGICAL UNIT NOT READY, OPERATION IN PROGRESS"
 1032 = "LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS"
 1033 = "LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS"
 1280 = "LOGICAL UNIT DOES NOT RESPOND TO SELECTION"
 1536 = "NO REFERENCE POSITION FOUND"
 1792 = "MULTIPLE PERIPHERAL DEVICES SELECTED"
 2048 = "LOGICAL UNIT COMMUNICATION FAILURE"
 2049 = "LOGICAL UNIT COMMUNICATION TIME-OUT"
 2050 = "LOGICAL UNIT COMMUNICATION PARITY ERROR"
 2051 = "LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)"
 2052 = "UNREACHABLE COPY TARGET"
 2304 = "TRACK FOLLOWING ERROR"
 2305 = "TRACKING SERVO FAILURE"
 2306 = "FOCUS SERVO FAILURE"
 2307 = "SPINDLE SERVO FAILURE"
 2308 = "HEAD SELECT FAULT"
 2560 = "ERROR LOG OVERFLOW"
 2816 = "WARNING"
 2817 = "WARNING - SPECIFIED TEMPERATURE EXCEEDED"
 2818 = "WARNING - ENCLOSURE DEGRADED"
 3072 = "WRITE ERROR"
 3079 = "WRITE ERROR - RECOVERY NEEDED"
 3080 = "WRITE ERROR - RECOVERY FAILED"
 3081 = "WRITE ERROR - LOSS OF STREAMING"
 3082 = "WRITE ERROR - PADDING BLOCKS ADDED"
 3328 = "ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR"
 3329 = "THIRD PARTY DEVICE FAILURE"
 3330 = "COPY TARGET DEVICE NOT REACHABLE"
 3331 = "INCORRECT COPY TARGET DEVICE TYPE"
 3332 = "COPY TARGET DEVICE DATA UNDERRUN"
 3333 = "COPY TARGET DEVICE DATA OVERRUN"
```

```
4352 = "UNRECOVERED READ ERROR"
4353 = "READ RETRIES EXHAUSTED"
4354 = "ERROR TOO LONG TO CORRECT"
4357 = "L-EC UNCORRECTABLE ERROR"
4358 = "CIRC UNRECOVERED ERROR"
4365 = "DE-COMPRESSION CRC ERROR"
4366 = "CANNOT DECOMPRESS USING DECLARED ALGORITHM"
4367 = "ERROR READING UPC/EAN NUMBER"
4368 = "ERROR READING ISRC NUMBER"
4369 = "READ ERROR - LOSS OF STREAMING"
5120 = "RECORDED ENTITY NOT FOUND"
5121 = "RECORD NOT FOUND"
5376 = "RANDOM POSITIONING ERROR"
5377 = "MECHANICAL POSITIONING ERROR"
5378 = "POSITIONING ERROR DETECTED BY READ OF MEDIUM"
5888 = "RECOVERED DATA WITH NO ERROR CORRECTION APPLIED"
5889 = "RECOVERED DATA WITH RETRIES"
5890 = "RECOVERED DATA WITH POSITIVE HEAD OFFSET"
5891 = "RECOVERED DATA WITH NEGATIVE HEAD OFFSET"
5892 = "RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED"
5893 = "RECOVERED DATA USING PREVIOUS SECTOR ID"
5895 = "RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT"
5896 = "RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE"
5897 = "RECOVERED DATA WITHOUT ECC - DATA REWRITTEN"
6144 = "RECOVERED DATA WITH ERROR CORRECTION APPLIED"
6145 = "RECOVERED DATA WITH ERROR CORR. & RETRIES APPLIED"
6146 = "RECOVERED DATA - DATA AUTO-REALLOCATED"
6147 = "RECOVERED DATA WITH CIRC"
6148 = "RECOVERED DATA WITH L-EC"
6149 = "RECOVERED DATA - RECOMMEND REASSIGNMENT"
6150 = "RECOVERED DATA - RECOMMEND REWRITE"
6152 = "RECOVERED DATA WITH LINKING"
6656 = "PARAMETER LIST LENGTH ERROR"
6912 = "SYNCHRONOUS DATA TRANSFER ERROR"
7424 = "MISCOMPARE DURING VERIFY OPERATION"
8192 = "INVALID COMMAND OPERATION CODE"
8448 = "LOGICAL BLOCK ADDRESS OUT OF RANGE"
8449 = "INVALID ELEMENT ADDRESS"
8450 = "INVALID ADDRESS FOR WRITE"
9216 = "INVALID FIELD IN CDB"
9217 = "CDB DECRYPTION ERROR"
9472 = "LOGICAL UNIT NOT SUPPORTED"
9728 = "INVALID FIELD IN PARAMETER LIST"
9729 = "PARAMETER NOT SUPPORTED"
9730 = "PARAMETER VALUE INVALID"
9731 = "THRESHOLD PARAMETERS NOT SUPPORTED"
9732 = "INVALID RELEASE OF PERSISTENT RESERVATION"
9733 = "DATA DECRYPTION ERROR"
9734 = "TOO MANY TARGET DESCRIPTORS"
9735 = "UNSUPPORTED TARGET DESCRIPTOR TYPE CODE"
9736 = "TOO MANY SEGMENT DESCRIPTORS"
9737 = "UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE"
9738 = "UNEXPECTED INEXACT SEGMENT"
9739 = "INLINE DATA LENGTH EXCEEDED"
9740 = "INVALID OPERATION FOR COPY SOURCE OR DESTINATION"
9741 = "COPY SEGMENT GRANULARITY VIOLATION"
9984 = "WRITE PROTECTED"
```

```
9985 = "HARDWARE WRITE PROTECTED"
9986 = "LOGICAL UNIT SOFTWARE WRITE PROTECTED"
9987 = "ASSOCIATED WRITE PROTECT"
9988 = "PERSISTENT WRITE PROTECT"
9989 = "PERMANENT WRITE PROTECT"
9990 = "CONDITIONAL WRITE PROTECT"
10240 = "NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED"
10241 = "IMPORT OR EXPORT ELEMENT ACCESSED"
10496 = "POWER ON, RESET, OR BUS DEVICE RESET OCCURRED"
10497 = "POWER ON OCCURRED"
10498 = "SCSI BUS RESET OCCURRED"
10499 = "BUS DEVICE RESET FUNCTION OCCURRED"
10500 = "DEVICE INTERNAL RESET"
10501 = "TRANSCEIVER MODE CHANGED TO SINGLE-ENDED"
10502 = "TRANSCEIVER MODE CHANGED TO LVD"
10752 = "PARAMETERS CHANGED"
10753 = "MODE PARAMETERS CHANGED"
10754 = "LOG PARAMETERS CHANGED"
10755 = "RESERVATIONS PREEMPTED"
10756 = "RESERVATIONS RELEASED"
10757 = "REGISTRATIONS PREEMPTED"
11008 = "COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT"
11264 = "COMMAND SEQUENCE ERROR"
11267 = "CURRENT PROGRAM AREA IS NOT EMPTY"
11268 = "CURRENT PROGRAM AREA IS EMPTY"
11270 = "PERSISTENT PREVENT CONFLICT"
11776 = "INSUFFICIENT TIME FOR OPERATION"
12032 = "COMMANDS CLEARED BY ANOTHER INITIATOR"
12288 = "INCOMPATIBLE MEDIUM INSTALLED"
12289 = "CANNOT READ MEDIUM - UNKNOWN FORMAT"
12290 = "CANNOT READ MEDIUM - INCOMPATIBLE FORMAT"
12291 = "CLEANING CARTRIDGE INSTALLED"
12292 = "CANNOT WRITE MEDIUM - UNKNOWN FORMAT"
12293 = "CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT"
12294 = "CANNOT FORMAT MEDIUM - INCOMPATIBLE MEDIUM"
12295 = "CLEANING FAILURE"
12296 = "CANNOT WRITE - APPLICATION CODE MISMATCH"
12297 = "CURRENT SESSION NOT FIXATED FOR APPEND"
12304 = "MEDIUM NOT FORMATTED"
12544 = "MEDIUM FORMAT CORRUPTED"
12545 = "FORMAT COMMAND FAILED"
12546 = "ZONED FORMATTING FAILED DUE TO SPARE LINKING"
13312 = "ENCLOSURE FAILURE"
13568 = "ENCLOSURE SERVICES FAILURE"
13569 = "UNSUPPORTED ENCLOSURE FUNCTION"
13570 = "ENCLOSURE SERVICES UNAVAILABLE"
13571 = "ENCLOSURE SERVICES TRANSFER FAILURE"
13572 = "ENCLOSURE SERVICES TRANSFER REFUSED"
14080 = "ROUNDED PARAMETER"
14592 = "SAVING PARAMETERS NOT SUPPORTED"
14848 = "MEDIUM NOT PRESENT"
14849 = "MEDIUM NOT PRESENT - TRAY CLOSED"
14850 = "MEDIUM NOT PRESENT - TRAY OPEN"
14851 = "MEDIUM NOT PRESENT - LOADABLE"
14852 = "MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY ACCESSIBLE"
15117 = "MEDIUM DESTINATION ELEMENT FULL"
15118 = "MEDIUM SOURCE ELEMENT EMPTY"
```

```
15119 = "END OF MEDIUM REACHED"
15121 = "MEDIUM MAGAZINE NOT ACCESSIBLE"
15122 = "MEDIUM MAGAZINE REMOVED"
15123 = "MEDIUM MAGAZINE INSERTED"
15124 = "MEDIUM MAGAZINE LOCKED"
15125 = "MEDIUM MAGAZINE UNLOCKED"
15126 = "MECHANICAL POSITIONING OR CHANGER ERROR"
15616 = "INVALID BITS IN IDENTIFY MESSAGE"
15872 = "LOGICAL UNIT HAS NOT SELF-CONFIGURED YET"
15873 = "LOGICAL UNIT FAILURE"
15874 = "TIMEOUT ON LOGICAL UNIT"
15875 = "LOGICAL UNIT FAILED SELF-TEST"
15876 = "LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG"
16128 = "TARGET OPERATING CONDITIONS HAVE CHANGED"
16129 = "MICROCODE HAS BEEN CHANGED"
16130 = "CHANGED OPERATING DEFINITION"
16131 = "INQUIRY DATA HAS CHANGED"
16132 = "COMPONENT DEVICE ATTACHED"
16133 = "DEVICE IDENTIFIER CHANGED"
16134 = "REDUNDANCY GROUP CREATED OR MODIFIED"
16135 = "REDUNDANCY GROUP DELETED"
16136 = "SPARE CREATED OR MODIFIED"
16137 = "SPARE DELETED"
16138 = "VOLUME SET CREATED OR MODIFIED"
16139 = "VOLUME SET DELETED"
16140 = "VOLUME SET DEASSIGNED"
16141 = "VOLUME SET REASSIGNED"
16142 = "REPORTED LUNS DATA HAS CHANGED"
16143 = "ECHO BUFFER OVERWRITTEN"
16144 = "MEDIUM LOADABLE"
16145 = "MEDIUM AUXILIARY MEMORY ACCESSIBLE"
17152 = "MESSAGE ERROR"
17408 = "INTERNAL TARGET FAILURE"
17664 = "SELECT OR RESELECT FAILURE"
17920 = "UNSUCCESSFUL SOFT RESET"
18176 = "SCSI PARITY ERROR"
18177 = "DATA PHASE CRC ERROR DETECTED"
18178 = "SCSI PARITY ERROR DETECTED DURING ST DATA PHASE"
18179 = "INFORMATION UNIT CRC ERROR DETECTED"
18180 = "ASYNCHRONOUS INFORMATION PROTECTION ERROR DETECTED"
18432 = "INITIATOR DETECTED ERROR MESSAGE RECEIVED"
18688 = "INVALID MESSAGE ERROR"
18944 = "COMMAND PHASE ERROR"
19200 = "DATA PHASE ERROR"
19456 = "LOGICAL UNIT FAILED SELF-CONFIGURATION"
19968 = "OVERLAPPED COMMANDS ATTEMPTED"
20736 = "ERASE FAILURE"
20737 = "ERASE FAILURE - INCOMPLETE ERASE OPERATION DETECTED"
21248 = "MEDIA LOAD OR EJECT FAILED"
21250 = "MEDIUM REMOVAL PREVENTED"
21762 = "INSUFFICIENT RESERVATION RESOURCES"
21763 = "INSUFFICIENT RESOURCES"
21764 = "INSUFFICIENT REGISTRATION RESOURCES"
22272 = "UNABLE TO RECOVER TABLE-OF-CONTENTS"
23040 = "OPERATOR REQUEST OR STATE CHANGE INPUT"
23041 = "OPERATOR MEDIUM REMOVAL REQUEST"
23042 = "OPERATOR SELECTED WRITE PROTECT"
```

```
23043 = "OPERATOR SELECTED WRITE PERMIT"
23296 = "LOG EXCEPTION"
23297 = "THRESHOLD CONDITION MET"
23298 = "LOG COUNTER AT MAXIMUM"
23299 = "LOG LIST CODES EXHAUSTED"
23808 = "FAILURE PREDICTION THRESHOLD EXCEEDED"
23809 = "MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED"
23810 = "LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED"
23811 = "SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED"
24063 = "FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)"
24064 = "LOW POWER CONDITION ON"
24065 = "IDLE CONDITION ACTIVATED BY TIMER"
24066 = "STANDBY CONDITION ACTIVATED BY TIMER"
24067 = "IDLE CONDITION ACTIVATED BY COMMAND"
24068 = "STANDBY CONDITION ACTIVATED BY COMMAND"
25344 = "END OF USER AREA ENCOUNTERED ON THIS TRACK"
25345 = "PACKET DOES NOT FIT IN AVAILABLE SPACE"
25600 = "ILLEGAL MODE FOR THIS TRACK"
25601 = "INVALID PACKET SIZE"
25856 = "VOLTAGE FAULT"
28416 = "COPY PROTECTION KEY EXCHANGE FAILURE - AUTHENTICATION FAILURE"
28417 = "COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT PRESENT"
28418 = "COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT ESTABLISHED"
28419 = "READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION"
28420 = "MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION"
28421 = "DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR"
29184 = "SESSION FIXATION ERROR"
29185 = "SESSION FIXATION ERROR WRITING LEAD-IN"
29186 = "SESSION FIXATION ERROR WRITING LEAD-OUT"
29187 = "SESSION FIXATION ERROR - INCOMPLETE TRACK IN SESSION"
29188 = "EMPTY OR PARTIALLY WRITTEN RESERVED TRACK"
29189 = "NO MORE TRACK RESERVATIONS ALLOWED"
29440 = "CD CONTROL ERROR"
29441 = "POWER CALIBRATION AREA ALMOST FULL"
29442 = "POWER CALIBRATION AREA IS FULL"
29443 = "POWER CALIBRATION AREA ERROR"
29444 = "PROGRAM MEMORY AREA UPDATE FAILURE"
29445 = "PROGRAM MEMORY AREA IS FULL"
29446 = "RMA/PMA IS ALMOST FULL"
```

GetPerformOPC

CMCDBurner **See Also: SetPerformOPC, Burn, BurnISOImage**
Returns the status of flag for performing OPC.

C Syntax:
```
BOOL GetPerformOPC(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL GetPerformOPC(void)
```

This *method* is used to determine the status of flag for performing OPC, before burning.

Return Value: Return value is the current status of flag for performing OPC.

GetReadSpeed

CMCDBurner **See Also: SetReadSpeed, GetDeviceMaxReadSpeed**
Returns the current read speed of device.

C Syntax:
```
BYTE GetReadSpeed(SCSIDEVICE *device)
```

C++ Syntax:
```
BYTE GetReadSpeed(void)
```

This *method* is used to get the current read speed of the currently selected device, for reading operations.

Return Value: Return value is the current read speed of the device and it is returned as X factor (i.e. 1 for 1X, 12 for 12X).

GetTestWrite

CMCDBurner **See Also: SetTestWrite, TestUnitReady, Burn, BurnISOImage**
Returns the status of test write mode.

C Syntax:
```
BOOL GetTestWrite(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL GetTestWrite(void)
```

This *method* is used to get the status of test writing mode flag, for the currently selected device. The test write mode simulates the writing activity on the device, except that the laser is never turned on and the disc remains unwritten.

Return Value: Return value is the current status of test write mode.

GetTotalBlocksOnDisc

CMCDBurner **See Also: GetFreeBlocksOnDisc, SessionsOnDisc**
Returns the count of total blocks on media.

C Syntax:
```
DWORD GetTotalBlocksOnDisc(SCSIDEVICE *device)
```

C++ Syntax:
```
DWORD GetTotalBlocksOnDisc(void)
```

This *method* is used to get the total number of blocks on the media inserted in the currently selected device.

Return Value: Return value is the count of total blocks on media.

Note: Block size is 2048 bytes.

GetVolumeID

CMCDBurner **See Also: SetVolumeID**
Returns the volume descriptor information of media.

C Syntax:
```
char *GetVolumeID(FILES *cdfiles)
```

C++ Syntax:
```
char *GetVolumeID(void)
```

This *method* is used to get the volume label information that you set with the **SetVolumeID**.

Note: This method will **NOT** return the Volume lable of disc currently inserted in the drive. use the Windows API instead.

GetWriteSpeed

CMCDBurner **See Also: SetWriteSpeed, GetMaxWriteSpeed, GetDeviceMaxWriteSpeed**
Returns the current write speed of device.

C Syntax:
```
BYTE GetWriteSpeed(SCSIDEVICE *device)
```

C++ Syntax:
```
BYTE GetWriteSpeed(void)
```

This *method* is used to get the current write speed of the currently selected device, for writing operations.

Return Value: Return value is the current write speed of the device and it is returned as X factor (i.e. 1 for 1X, 12 for 12X).

ImportSession

CMCDBurner
Imports a previously created session.

C Syntax:
```
int ImportSession(FILES *cdfiles, BYTE SessionNo, char *dest_dir)
Unicode version:
int ImportSessionW(FILES *cdfiles, BYTE SessionNo, wchar_t *dest_dir)
```

C++ Syntax:
```
DWORD ImportSession(BYTE SessionNo, char *dest_dir)
Unicode version:
DWORD ImportSessionW(BYTE SessionNo, wchar_t *dest_dir)
```

This *method* is used to import a previously created session, represented by *SessionNo*. To import previous sessions, set *SessionNo* as 0. *dest_dir* specifies the path on media where the session[s] will be imported.

Return Value: Return value is non-zero if session is imported successfully otherwise a 0 value is return.

InsertDir

CMCDBurner **See Also: CreateDir, InsertFile, SetReplaceFileStatus**
Adds a directory, to be burned on media.

C Syntax:

MultiByte Character System version:

```
  DirEntry *InsertDir(FILES *cdfiles, char *dest_path, char *file_path, char
*file_specs, BOOL Recursive, BOOL SavePath)
```

Unicode version:

```
  DirEntry *InsertDirW(FILES *cdfiles, wchar_t *dest_path, wchar_t
*file_path, wchar_t *file_specs, BOOL Recursive, BOOL SavePath)
```

C++ Syntax:

MultiByte Character System version:

```
  DirEntry *InsertDir(char *dest_path, char *file_path, char *file_specs,
BOOL Recursive, BOOL SavePath)
```

Unicode version:

```
  DirEntry *InsertDirW(wchar_t *dest_path, wchar_t *file_path, wchar_t
*file_specs, BOOL Recursive, BOOL SavePath)
```

This *method* is used to add directories, to be burned on a CD/DVD. At least one file/directory is required before calling Burn method. *dest_path* is the path on media where the directory is inserted while *file_path* is the path of actual directory, to be inserted. *file_specs* describes the file mask. *Recursive* enforces insertion of all underlying contents of source directory. *SavePath* specifies that the actual path of source directory is used when inserting the directory at destination location.

Return Value: If the directory is inserted successfully then a pointer to the directory structure (i.e. *DirEntry*) is returned otherwise a *NULL* is returned.

InsertFileWithName

CMCDBurner **See Also: InsertFile, InsertDir, CreateDir, SetReplaceFile**
Adds a file to CD Layout with different name than on hard disk.

C Syntax:

MultiByte Character System version:

```
FileEntry *InsertFileWithName(FILES *cdfiles, char *DestPath, char
*FileName, char *LongNameOnDisc, char *ShortNameOnDisc);
```

Unicode version:

```
FileEntry *InsertFileWithNameW(FILES *cdfiles, wchar_t *DestPath, wchar_t
*FileName, wchar_t *LongNameOnDisc, wchar_t *ShortNameOnDisc);
```

C++ Syntax:

MultiByte Character System version:

```
FileEntry *InsertFileWithName(char *DestPath, char *FileName, char
*LongNameOnDisc, char *ShortNameOnDisc)
```

Unicode version:

```
FileEntry *InsertFileWithNameW(wchar_t *DestPath, wchar_t *FileName,
wchar_t *LongNameOnDisc, wchar_t *ShortNameOnDisc)
```

This *function* is used to add files to be burned on a CD/DVD. The same rules of **InsertFile** apply to this function, except you can change the name on CD/DVD with changing the name on hard disk.

InsertFile

CMCDBurner **See Also: InsetFileWithName, InsertDir, CreateDir, SetReplaceFile**
Adds a file, to be burned on media.

C Syntax:

```
MultiByte Character System version:
  FileEntry *InsertFile(FILES *cdfiles, char *dest_path, char *file_name)

Unicode version:
  FileEntry *InsertFileW(FILES *cdfiles, wchar_t *dest_path, wchar_t
*file_name)
```

C++ Syntax:

```
MultiByte Character System version:
  FileEntry *InsertFile(char *dest_path, char *file_name)

Unicode version:
  FileEntry *InsertFileW(wchar_t *dest_path, wchar_t *file_name)
```

This *method* is used to add files, to be burned on a CD/DVD. At least one file/directory is required before calling Burn method. *dest_path* is the path on media where the file is inserted while *file_name* is the path of actual file, to be inserted.

Return Value: If the file is inserted successfully then a pointer to the file structure (i.e. *FileEntry*) is returned otherwise a *NULL* is returned.

LoadMedium

CMCDBurner **See Also: EjectMedium, LockMedium, UnlockMedium, TestUnitReady**

Loads media from the currently selected device.

C Syntax:

```
BOOL LoadMedium(SCSIDEVICE *device, char Immediate)
```

C++ Syntax:

```
BOOL LoadMedium(void)
```

This *method* is used to invoke a Load operation of the currently selected device.

Return Value: Return value is *TRUE* if the media is loaded successfully otherwise a *FALSE* value is return.

LockMedium

CMCDBurner **See Also: EjectMedium, LoadMedium, UnlockMedium, TestUnitReady**

Prevents media removal from the currently selected device.

C Syntax:

```
BOOL LockMedium(SCSIDEVICE *device)
```

C++ Syntax:

```
BOOL LockMedium(void)
```

This *method* is used to Lock (no eject allowed) the currently selected device. This is a precautionary measure and will always be used so that an accidental eject does not spoil the current media.

Return Value: Return value is *TRUE* if the device is locked successfully otherwise a *FALSE* value is return.

Prepare

CMCDBurner **See Also: Burn, BurnISOImage**
Makes necessary arrangements before a burn operation is started.

C Syntax:
```
DWORD Prepare(FILES *cdfiles)
```

C++ Syntax:
```
DWORD Prepare(void)
```

This *method* should be invoked just before starting the burn operation. It sets up track information, to be burned on a CD.

Return Value: Return value is non-zero if operation is successful otherwise a 0 value is return.

SelectDevice

CMCDBurner **See Also: GetDeviceName, GetDeviceCount**
Selects a specified device.

C Syntax:
```
BOOL SelectDevice(SCSIDEVICE *device, BYTE Id)
```

C++ Syntax:
```
BOOL SelectDevice(BYTE Id)
```

This *method* is used to select a device, specified by *Id*, for burning purposes.

Return Value: Return value is *TRUE* if the device is selected successfully otherwise a *FALSE* value is return.

SessionsOnDisc

CMCDBurner **See Also: GetFreeBlocksOnDisc, GetTotalBlocksOnDisc**
Returns the count of available sessions on media.

C Syntax:
```
BYTE SessionsOnDisc(SCSIDEVICE *device)
```

C++ Syntax:
```
BYTE SessionsOnDisc(void)
```

This *method* is used to get the total number of available sessions on the media, inserted in the currently selected device.

Return Value: Return value is the count of available sessions on the media, inserted in the currently selected device.

SetBurnProof

CMCDBurner **See Also: GetBurnProof, Burn, BurnISOImage**
Sets the status of BURN-Proof flag.

C Syntax:
```
void SetBurnProof(SCSIDEVICE *device, BOOL BurnProof)
```

C++ Syntax:
```
void SetBurnProof(BOOL BurnProof)
```

This *method* is used to set the status of BURN-Proof (Buffer Under RuN error Proof) flag which helps in protection from BURN error. Set *BurnProof* as *TRUE* to enable the status of BURN-Proof flag or *FALSE* to disable it.

SetCacheSize

CMCDBurner **See Also: GetCacheSize, GetDeviceBufferSize, GetCachePosition**
Sets the size of software cache buffer.

C Syntax:
```
void SetCacheSize(FILES *cdfiles, int CacheSize)
```

C++ Syntax:
```
void SetCacheSize(DWORD CacheSize)
```

This *method* is used to set the size of internal software cache buffer. *CacheSize* should be more than 2 MB.

This *method* is used to give a basic level of **Buffer Under RuN Error Protection** using software buffers. The data, to be burnt, is kept in the system RAM, before it is burnt to the disc.

SetFinalizeDisc

CMCDBurner **See Also: GetFinalizeDisc, GetFreeBlocksOnDisc, Burn, BurnISOImage, EraseDisc**

Sets the status of Finalize/Close Disc flag.

C Syntax:

```
void SetFinalizeDisc(SCSIDEVICE *device, BOOL FinalizeDisc)
```

C++ Syntax:

```
void SetFinalizeDisc(BOOL FinalizeDisc)
```

This *method* should be used with caution. It is used to finalize/close the media in the device. If *FinalizeDisc* is set to *TRUE* on a media, no more data can be written to the media, even if there is free space available on it.

SetPerformOPC

CMCDBurner **See Also: GetPerformOPC, Burn, BurnISOImage**
Sets the status of flag for performing OPC.

C Syntax:
```
void SetPerformOPC(SCSIDEVICE *device, BOOL PerformOPC)
```

C++ Syntax:
```
void SetPerformOPC(BOOL PerformOPC)
```

This *method* is used to set the status of flag for performing OPC, before burning. Set *PerformOPC* as *TRUE* to enable the status of flag for performing OPC or *FALSE* to disable it.

SetReadSpeed

CMCDBurner **See Also: GetReadSpeed, GetDeviceMaxReadSpeed**
Sets the current read speed of device.

C Syntax:
```
BOOL SetCDSpeed(SCSIDEVICE *device, WORD ReadSpeed, WORD WriteSpeed)
```

C++ Syntax:
```
void SetReadSpeed(BYTE ReadSpeed)
```

This *method* is used to set the current read speed of the currently selected device, for reading operations. *ReadSpeed* must be set by X factor (i.e 1 for 1X, 12 for 12X). To select the maximum read speed of the device, set *ReadSpeed* as 0.

SetReplaceFile

CMCDBurner **See Also: InsertDir, InsertFile**
Sets the status of flag, used for replacing existing files/directories.

C Syntax:
```
void SetReplaceFile(SCSIDEVICE *device, BOOL ReplaceFile)
```

C++ Syntax:
```
void SetReplaceFile(BOOL ReplaceFile)
```

This *method* is used to set the status of flag used for replacing existing files on the media which have the same name and path information as that of files/directories, to be burnt.

SetTestWrite

CMCDBurner **See Also: GetTestWrite, TestUnitReady, Burn, BurnISOImage**
Sets the status of test write mode.

C Syntax:
```
void SetTestWrite(SCSIDEVICE *device, BOOL TestWrite)
```

C++ Syntax:
```
void SetTestWrite(BOOL TestWrite)
```

This *method* is used to set the status of test writing mode flag, for the currently selected device. The test write mode simulates the writing activity on the drive, except that the laser is never turned on and the disc remains unwritten. Set *TestWrite* as *TRUE* to enable test write mode or *FALSE* to disable it.

SetVolumeID

CMCDBurner **See Also: GetVolumeID**
Sets the volume descriptor information of media.

C Syntax:
```
void SetVolumeID(FILES *cdfiles, char *VolumeID)
Unicode version:
void SetVolumeID(FILES *cdfiles, wchar_t *VolumeID)
```

C++ Syntax:
```
void SetVolumeID(char *VolumeID)
Unicode version:
void SetVolumeIDW(wchar_t *VolumeID)
```

This *method* is used to set the volume descriptor information, i.e. *VolumeID*, of inserted media in currently selected device.

SetWriteSpeed

CMCDBurner **See Also: GetWriteSpeed, GetMaxWriteSpeed, GetDeviceMaxWriteSpeed**
Sets the current write speed of device.

C Syntax:
```
void SetWriteSpeed(SCSIDEVICE *device, BYTE WriteSpeed)
```

C++ Syntax:
```
void SetWriteSpeed(BYTE WriteSpeed)
```

This *method* is used to set the current write speed of the currently selected device, for writing operations. *WriteSpeed* must be set by X factor (i.e 1 for 1X, 12 for 12X). To automatically select the maximum write speed, set *WriteSpeed* as 0.

TestUnitReady

CMCDBurner **See Also: GetTestWrite, SetTestWrite**
Tests whether device is ready and media is inserted.

C Syntax:
```
BOOL TestUnitReady(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL TestUnitReady(void)
```

This *method* is used to test the status of the physical device. It checks to see if the device is ready and CD/DVD is inserted.

Return Value: Return value is *TRUE* if the selected device is ready and media is loaded in it otherwise a *FALSE* value is return.

UnlockMedium

CMCDBurner **See Also: EjectMedium, LoadMedium, LockMedium, TestUnitReady**
Allows media removal from the currently selected drive.

C Syntax:
```
BOOL UnlockMedium(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL UnlockMedium(void)
```

This *method* is used to unlock (eject allowed) the selected device.

Return Value: Return value is *TRUE* if the device is unlocked successfully otherwise a *FALSE* value is return.

Writable

CMCDBurner **See Also: GetComponentState**
Validates the status of media as burnable.

C Syntax:
```
BOOL Writable(SCSIDEVICE *device)
```

C++ Syntax:
```
BOOL Writable(void)
```

This *method* is used to find out whether the media, inserted in the selected device, can be burned.

Return Value: Return value is *TRUE* if the inserted media can be burned otherwise a *FALSE* value is return.

What is BURN-Proof?

BURN-Proof is a proprietary technology, which helps protect from Buffer Under Run Error.

Where does the BURN-Proof name come from?

BURN-Proof stands for Buffer Under RuN error Proof.

The word 'BURN-Proof' is a Trademark of SANYO Electric Co., Ltd.

Performing OPC is a special technique used in newer CD-Recorders for monitoring and maintaining the quality of the disc writing and ensuring the accuracy of all the mark and lands lengths across the disc. The term Performing OPC actually describes a general process which is also known by several trade names including "Dynamic Power Control (DPC)" and "Direct Read During Write (DRDW)". There may be differences in execution which gives some of these implementations competitive advantages over others.