NeroCOM

1.6.0.0

The Nero Type Library

***NeroCOM* will only work with**
a fully installed *Nero* version!

Nero AG, Im Stoeckmaedle 18, 76307 Karlsbad, Germany

1. Contents

2. License Agreement

IMPORTANT: PLEASE READ THE SOFTWARE LICENSE AGREEMENT ("LICENSE") CAREFULLY BEFORE USING THE SOFTWARE.

USING THE SOFTWARE INDICATES YOUR ACKNOWLEDGMENT THAT YOU HAVE READ THE LICENSE AND AGREE TO ITS TERMS.

The license agreement is contained in a text file, “NeroSDK_License.txt”, to be found in the root folder of the installation package.

3. Introduction

3.1. *NeroCOM*

NeroCOM is a set of COM (Component Object Model) classes and interfaces that can be used in any programming language that supports COM philosophy (practically all modern languages for Win32 platforms including C/C++, Delphi, Visual Basic, Java, etc.). *NeroCOM* is language independent.

NeroCOM consists of a single DLL file that depends on no other libraries.

Although *NeroCOM* can be used in many programming language environments, it was primarily designed and tested using Visual Basic. This is because the majority of COM clients are Visual Basic programs. The choice of Visual Basic as design and testing environment should not influence the usability of *NeroCOM* in other programming environments. It is just that Visual Basic has its specifics in terms of how it uses COM technology and these had to be addressed if *NeroCOM* was to be successfully used by Visual Basic client programs.

NeroCOM is nothing without a properly installed *Nero Burning ROM* or some other application that installs a shared *NeroAPI* DLL. *NeroCOM* is just a smart COM wrapper around the *NeroAPI*.

3.2. Motivation

The *NeroCOM* interface lets you use the power of *Nero* and the *NeroAPI* with as little programmatic overhead as possible.

This document has been designed to help you rapidly get familiar with *NeroCOM*.

Unfortunately nothing is ever perfect. So the author would be grateful if you sent your suggestions or pointed out errors, both in our code and documentation.

3.3. Overview

This paper, the documentation of the *NeroCOM* interface, contains some practical guidelines on how to use and not use the component. A complete Visual Basic sample is included where a Visual Basic application that uses *NeroCOM* is created from scratch. This documentation is completed by a list of types and objects.

3.4. Requirements

This documentation assumes that *Nero6* or later is already installed on your computer. The current *NeroCOM* version is supported by *Nero6*.

3.5. Required Skills

This documentation is directed towards Software developers who have gathered some experience in Visual Basic programming, or are familiar with the usage of COM objects in any other programming language.

Although the declarations of types and objects are displayed in Visual Basic style, it should be easy enough determine their counterparts in any other programming language.

3.6. The NeroSDK Forum

We provide a forum for all users of the *NeroSDK* to get in dialog with each other at http://www.nero.com/link.php?topic_id=17. We will also monitor the messages from time to time and try to help where possible.

4. Available Files

Here is a list of files that are available for *NeroCOM.*

4.1. Documentation

Path \ File Name	Description
NeroCOM.pdf	This document.

4.2. Examples

NeroFiddlesCOM.NET is an extensive example application to demonstrate how *NeroCOM* can be used from C#.

Path \ File Name	Description
App.ico	The Icon for the application.
AssemblyInfo.cs	Assembly information.
BrowseForFolder.cs	Implements the browse for folder functionality.
BurnDiscCopy.cs	Implements the copy disc feature.
BurnDiscCopy.resx	Resource file for the disk copy feature.
BurnImage.cs	Implementation of the burn image feature.
BurnImage.resx	Resource file for the burn image feature
BurnIsoAudio.cs	Implementation file for ISO and audio burning.
BurnIsoAudio.resx	Resource file for ISO and audio burning.
BurnProgress.resx	Burn progress resource file.
ChooseBurnFlags.cs	Implementation file for burn flags setting.
ChooseBurnFlags.resx	Resource file for burn flags setting.
ChooseMedia.cs	Implementation file for media selection.
ChooseMedia.resx	Resource file for media selection.
ChooseSession.cs	Implementation file for session selection.
ChooseSession.resx	Resource file for session selection.
ChooseSpeed.cs	Implementation file for speed selection.
ChooseSpeed.resx	Resource file for speed selection.
ControlEnabler.cs	Control enabler implementation file.
DAE.cs	Digital audio extraction implementation file.
DAE.resx	Resource file for digital audio extraction.
DAEProgress.cs	Digital audio extraction progress display implementation.
DAEProgress.resx	Resource file for digital audio extraction progress.
DiscInfo.cs	Implementation file for disc information.
DiscInfo.resx	Resource file for disk information.
DriveInfo.cs	Implementation fine for drive information.
DriveInfo.resx	Resource file for drive information.
Erase.cs	CD/RW erasing feature implementation.
Erase.resx	Resource file for CD/RW erasing.

Path \ File Name	Description
EraseProgress.cs	CD/RW erasing progress display implementation.
EraseProgress.resx	CD/RW erasing progress display resource file.
EstimateTrackSize.cs	Track size estimation implementation.
EstimateTrackSize.resx	Track size estimation resource file.
NeroFiddlesCOM.NET.sln	Microsoft Visual Studio Solution File.
SpeedComboBox.cs	Speed combo box implementation.
SpeedComboBox.resx	Speed combo box resource file.
VMSInfo.cs	Virtual multi session information implementation.
VMSInfo.resx	Virtual multi session information resource file.
WaitCD.cs	Wait CD dialog implementation.
WaitCD.resx	Wait CD dialog resource file.
BurnProgress.cs	Burned progress display implementation.
Main.cs	Main dialog implementation file.
Main.resx	Main dialog resource file.
NVAPI.cs	*Nero Vision API* access dialog implementation
NVAPI.resx	*Nero Vision API* access dialog resource file.
NVAPIEstimate.cs	*Nero Vision API* estimation feature implementation.
NVAPIEstimate.resx	*Nero Vision API* estimation resource file.
NeroFiddlesCOM.NET.csproj	C# project file.
NeroFiddlesCOM.NET.csproj.user	Project configuration file.

4.3. Running the Precompiled Sample Application

After installing *Nero*, you should be able to run the sample.

In the Windows Explorer navigate to the sample directory and double click on “NeroFiddlesCOM.NET.exe”.

5. *NeroCOM* Objects

There are many *NeroCOM* objects that correspond to C/C++ data structures defined in *NeroSDK*'s NeroAPI.h. Some of these *NeroCOM* objects are **creatable** and some are not. This means that some of them can be directly created by the client while the others are created by *NeroCOM* itself and returned to the client indirectly.

Here is a list of all objects in *NeroCOM* separated in two groups in terms of whether they are created directly by the client or indirectly.

Created by the client	Created by *NeroCOM*
Nero	NeroBurnAtOnceInfo
NeroVideoItem	NeroDataInputStreams
NeroFile	NeroDrive
NeroFolder	NeroDrives
NeroISOTrack	NeroSpeeds
NeroFileSystemDescContainer	NeroCDInfo
NeroFileProducer	NeroTracks
NeroAudioTrack	NeroTrack
NeroFreestyleTrack	NeroRelativeIndexBlkPositions
NeroFreestyleTracks	NeroFolders
NeroVideoItems	NeroFiles
NeroAudioTracks	NeroCDStamp
NeroFSSecNo	NeroDirectoryContainer
	NeroDataInputStream
	NeroDirectoryEntryContainer
	NeroFileContent
	NeroImportDataTrackInfo
	NeroFSPartitionInfo
	NeroFileSystemBlock

The first object that client needs to create is **Nero**. This object has a number of events associated with it. These events correspond to *NeroAPI*'s DLG_xxx user dialog callback constants. This is why the Nero object is so important. It is responsible for handling all of the DLG_xxx callbacks that are essential for *NeroAPI* operation. The usual Visual Basic piece of code to declare a variable would be:

```
Dim WithEvents Nero as Nero
```

The **WithEvents** tells Visual Basic to use COM connection points to establish a callback mechanism to the Nero object. This way Visual Basic can handle DLG_xxx events fired from a Nero object.

An example of a Visual Basic event on a Nero object would be:

```
Private Sub Nero_OnWaitCD(WaitCD As NEROAPI_WAITCD_TYPE, WaitCDLocalizedText
As String)
End Sub
```

The usual sequence of events is to first create the Nero object:

```
Set Nero = CreateObject("Nero.Nero")
```

… and then set one or more of the object’s properties, such as **LanguageFile**, **OverburnSize**, **EnableOverburn** and **WriteBufferSize**. These properties are read/write but can be written to only before the first access to the **GetDrives** function. This function returns a drives collection object that enumerates the drives in the machine. **GetDrives** returns a **NeroDrives** object. This object is an example of a **non-creatable** object. It can only be created indirectly by *NeroCOM* itself.

NeroDrives object behaves like a “standard” collection object such that it complies with Visual Basic’s FOR EACH language statement, e.g.:

```
Dim drives as NeroDrives
Dim drive as NeroDrive

Set drives = Nero.GetDrives(MEDIA_CDROM)

For Each drive In drives
‘ some code here
Next
```

NeroDrives is a drives collection. It is used to get access to **NeroDrive** objects that correspond to the physical devices as found and provided by the *NeroAPI*. NeroDrive object is another example of a non-creatable object that is created indirectly by *NeroCOM* and returned to the client.

The NeroDrive object, just like the Nero object, has a set of associated events such as **OnProgress** (notifies the client of progress), **OnWriteDAE** (provides client with audio data in the process of digital audio extraction), **OnDoneDAE** (notifies client that DAE is finished), etc. Basically, everything revolves around a NeroDrive.

5.1. Handling Events

In COM, events are usually only used as a method of notifying the client that something happened. In *NeroCOM* we have the need to also collect some information from a client. Although not practical in terms of COM architecture, we are using events for both client notifications as well as for gathering information from a client. In addition, in COM architecture events are designed in such a fashion that the client **can** but is not obliged to handle **any** events. A Visual Basic (or any other) client program can choose to **not** handle any of the events provided by *NeroCOM*. This is simply by COM design but it is not suitable for *NeroAPI* as there are critical events that simply must be handled. To solve this problem, *NeroCOM* provides a default behavior for each of the events exposed to clients so that even if the client provides no event handlers, everything still works fine. Of course, if the client chooses not to handle the events, it cannot expect to have some specific behavior other than default. It is best for the client to handle **all** events (that return information) and at least provide some sensible default behavior. This way the client will **know** the defaults because it provided them instead of the *NeroCOM*'s defaults.

All this talk is needed in order to understand that the client may create a Nero object to start things running and then the Nero object might get out of scope of the current block and get destroyed. If any other objects (descending from Nero object) remain instantiated, *NeroAPI* will remain active but as there is no Nero object, critical events exposed by the Nero object **cannot** be handled. This kind of programming practice is something that cannot lead to a successful program as it cannot use some of the basic *NeroCOM* features.

5.2. Important Considerations

Basically, there are a few things that should be stressed out for the use of *NeroCOM*:

- The Nero object must first be created before any other, and it should be the last to get destroyed, as much depends on it.
- All events that are created by the Nero object and return a value should be handled. If any of them is not handled, it is not the end of the world but the default behavior is probably not going to be satisfactory for any serious application. A serious application should handle all notifications from the *NeroAPI* forwarded by *NeroCOM* through Nero object events.
- The GetDrives function of a Nero object is what triggers the *NeroAPI* initialization. After it has initialized the four aforementioned properties can no longer be written to. They are used only before initialization.
- Do not register more than one handler for events (i.e. do not create multiple sink objects). If your programming environment forces you to, make sure that each callback is only processed once!

6. The C# Example Application

In the following chapters you will find an introduction to the C# example application for *NeroCOM* that is distributed as part of the *NeroSDK*.

The example was created with Visual Studio C# .NET.

6.1. Nero Keeps Fiddling!

In the *NeroAPI* documentation we explained how to create a simple MFC application – *NeroFiddles*, because supposedly Nero fiddled while Rome burned.

NeroFiddlesCOM is a Visual Basic example that offers the same limited amount of functionality. It is available as part of the *NeroSDK*.

In the following, however, we will concentrate on the C# example, ***NeroFiddlesCOM.NET***. This application offers a feature demonstration of almost the entire functionality provided by *NeroCOM*:

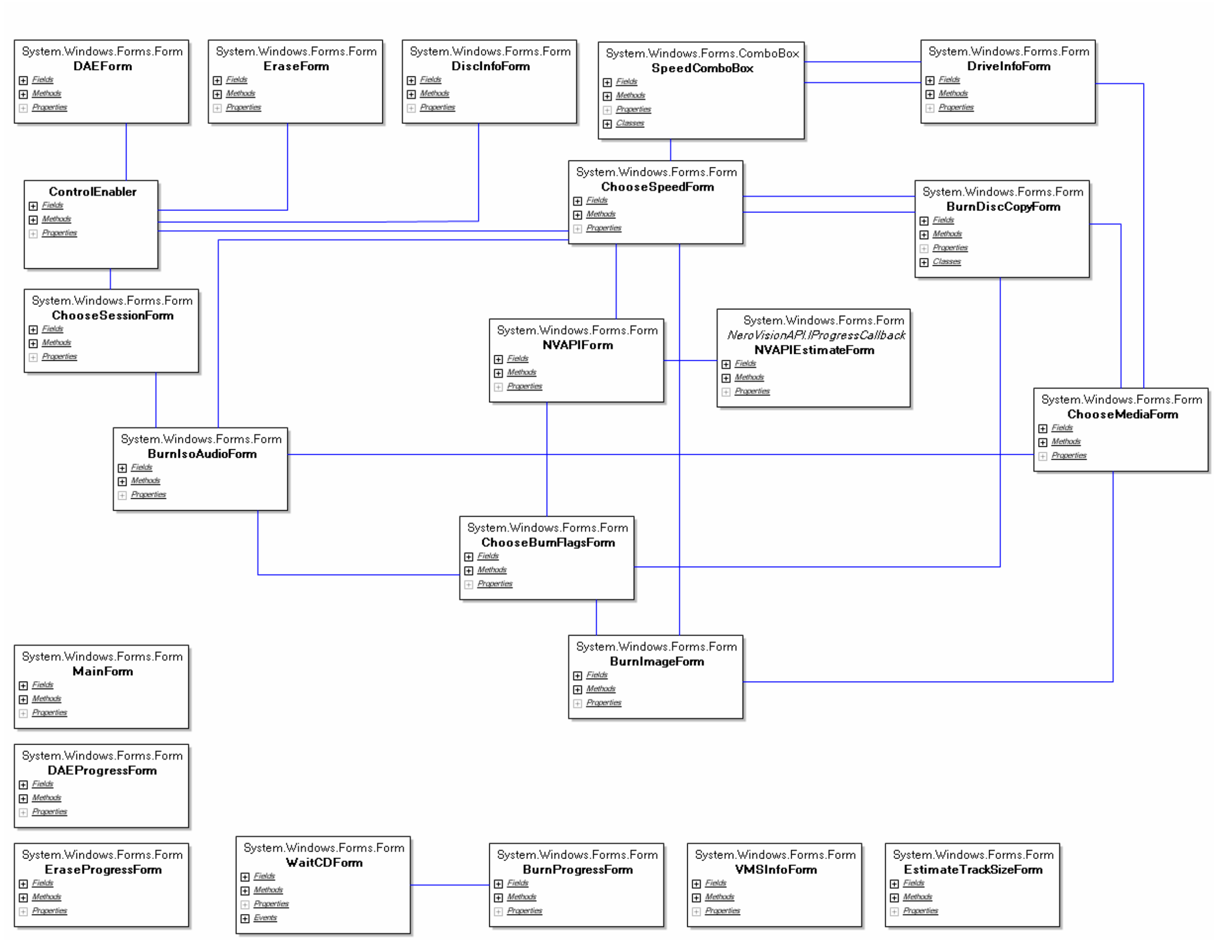
- drive information
- disc information
- image information
- burn ISO/Audio disc
- burn image
- copy disc
- burn *Nero Vision API* project
- eject disc
- load disc
- Digital Audio Extraction
- erase rewritable media

6.2. Running the Application

Choose “Debug/Start” from Visual Studio.NET after loading the project or use the Windows Explorer to navigate to the executable and open it.

6.3. Class Overview

The following diagram shows a class overview of the example application. The individual classes are explained in more detail in the following sections.

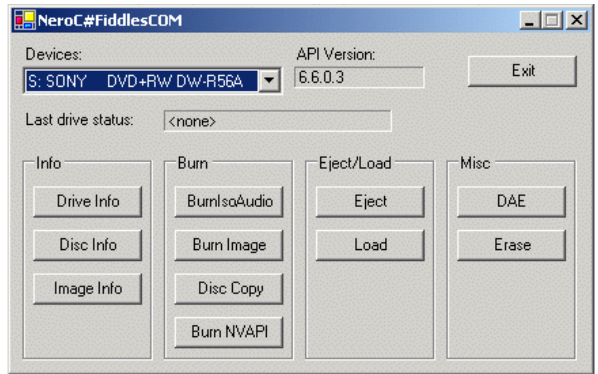


6.4. Main Dialog

This is the main dialog of the NeroFiddlesCOM.NET application.

It displays the found Nero API version and populates a list box with available devices.

The buttons will branch to different feature demonstrations of *NeroCOM* capabilities.



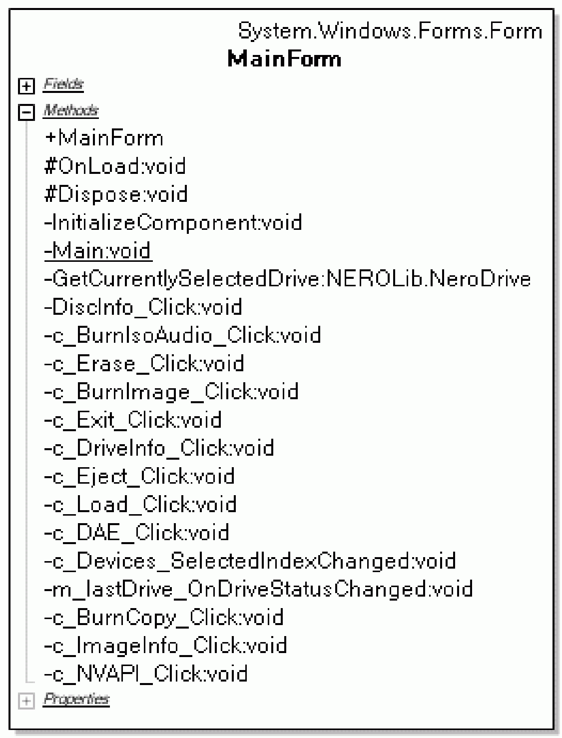
This class diagram shows the methods that are implemented in the MainForm class.

Mainly, it contains the application main entry point and a number of button event handling functions.

When the main form loads, the Nero object is created first.

Then the available devices are retrieved and fed to the combo box.

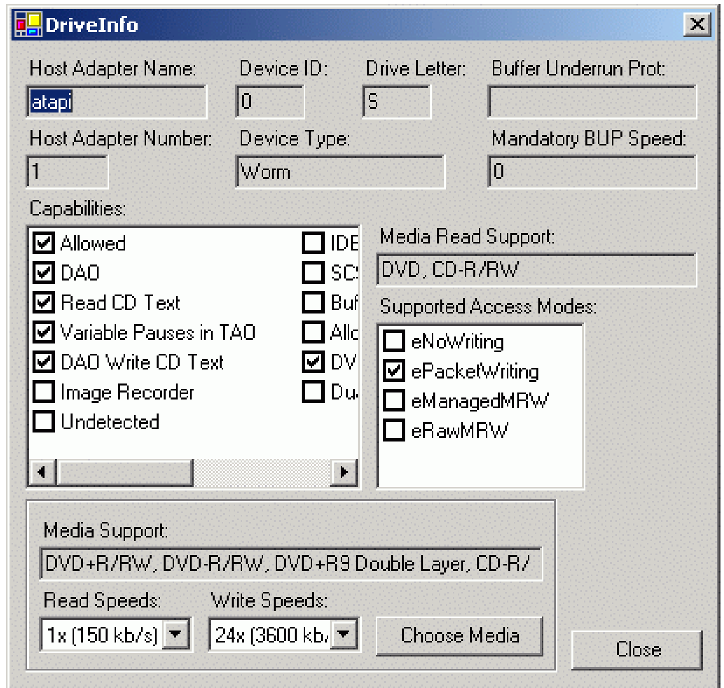
Each of these button handler methods instantiates the form class that provides the requested feature.



6.5. Drive Info

The drive information dialog displays the properties of a device that are relevant in the context of *NeroCOM*:

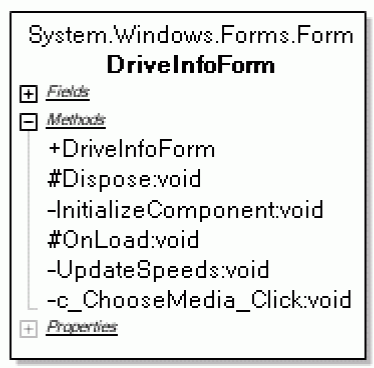
- host adapter name
- device ID
- drive letter
- capabilities
- supported media
- access modes
- supported read speeds
- supported write speeds



The "Choose Media" allows selecting one ore more media types. After the selection, the application will update the contents of the read and write speed dropdown boxes, which are dependant on the chosen media.

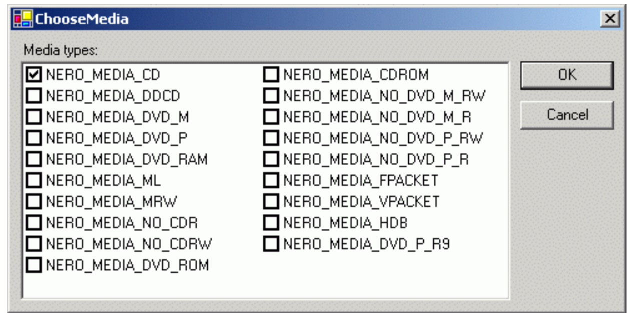
The OnLoad method uses the NeroDrive parameter that was provided by the calling class to determine the features of the drive.

The UpdateSpeeds method is used to populate the speed list boxes.



6.6. Choose Media

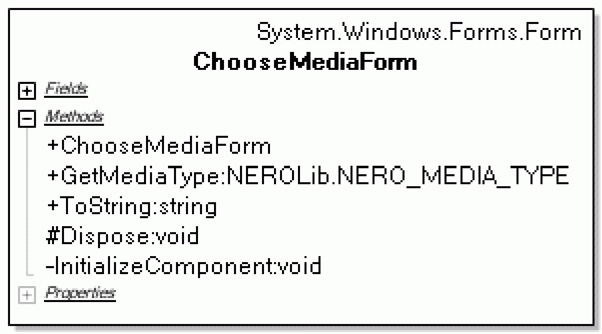
This is the media selection dialog. It allows the user to select one or more supported media types. The dialog is re-used in a number of different contexts that require media information.



The GetMediaType method evaluates the checked entries and returns a NERO_MEDIA_TYPE value.

It is referenced by these classes:

- DriveInfoForm
- BurnDiscCopyForm
- DriveInfoForm
- BurnIsoAudioForm

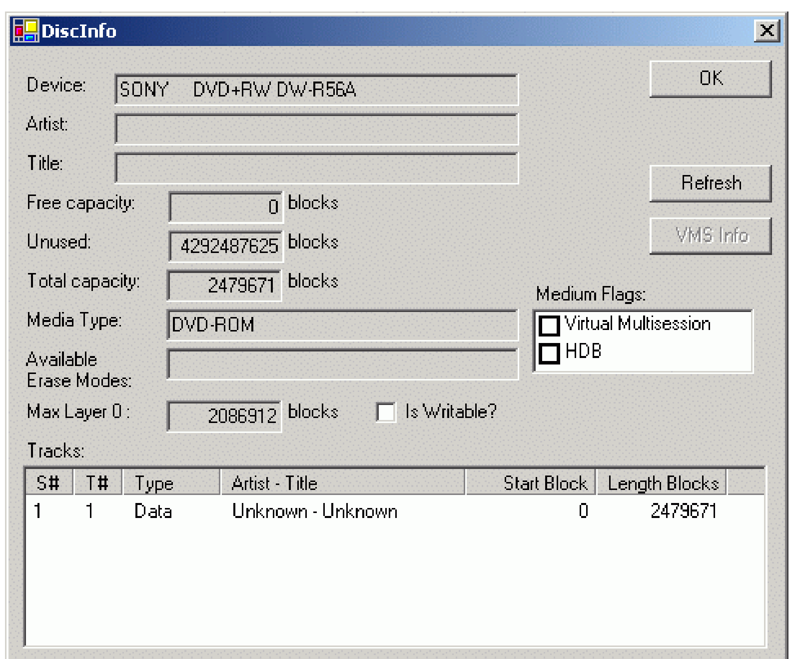


6.7. Disc Information

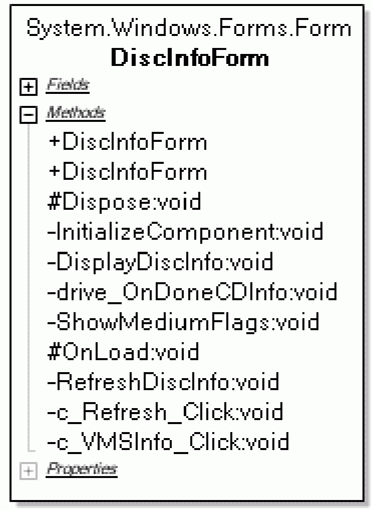
The disc info dialog provides information on

- device details
- artist
- title capacity
- media type
- media flags
- available erase modes
- tracks

When the media is of Virtual Multi Session kind, the "VMS Info" button is enabled.



The disc information dialog is used to display information about discs. Those discs may either exist in actual devices or as image files.



The DiscInfoForm constructor is polymorphic and has forms for either physical drives or image files.

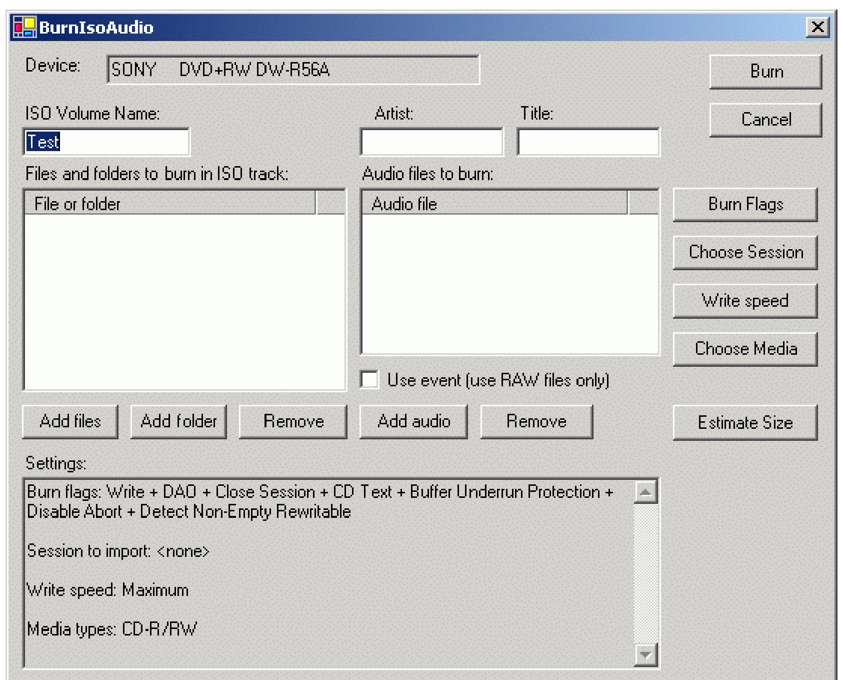
Depending on physical disc or image selection, the OnLoad method either calls GetDiscImageInfo directly or registers for the OnDoneCDInfo event.

If the media is a physical disc, the CDInfo method of NeroDrive is called to retrieve the information.

OnDoneCDInfo is fired when CDInfo is finished. It unsubscribes from the event and calls DisplayDiscInfo.

DisplayDiscInfo displays the information and enables or disables the "VMS Info" button, depending on the existence of NCDIMF_VIRTUALMULTISESSION.

6.8. Burn ISO/Audio



The dialog for ISO and Audio burning allows adding data and audio files or entire data folders to a compilation.

Through additional dialogs, burn flags can be altered, a session can be chosen for import, the write speed can be changed and the required media kind selected.

In addition, an estimate for the resulting size can be requested from *NeroCOM*.

When adding files to the compilation, multiple files can be selected.

If audio files are added, a filter is applied so that only files containing supported formats can be selected.

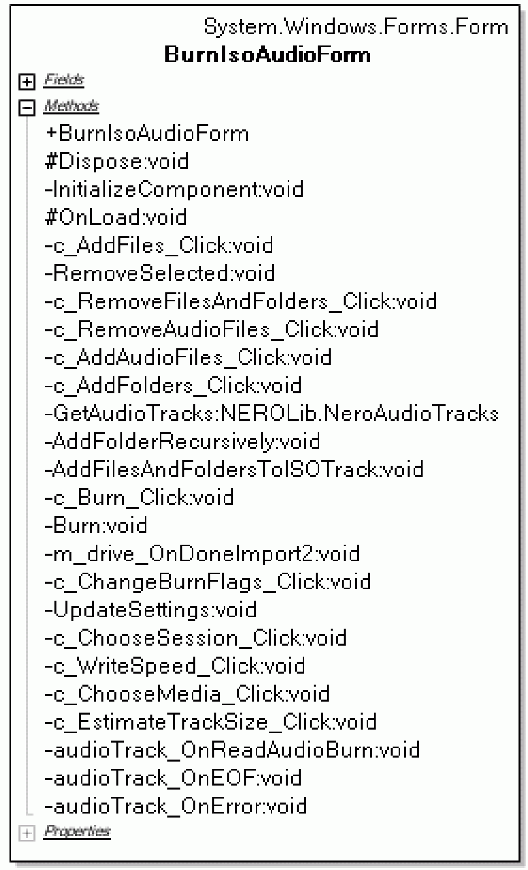
The BrowseForFolderClass is used when a folder is added.

GetAudioTracks is a helper function that assembles a NeroAudioTracks collection from the audio files in a list view.

If track size estimation is requested, the ISO track is created, all required elements are assigned and a form with further track size estimation options is displayed.

The Burn method performs the actual burning.

In addition, the event handlers OnReadAudioBurn, one of and OnError are implemented.

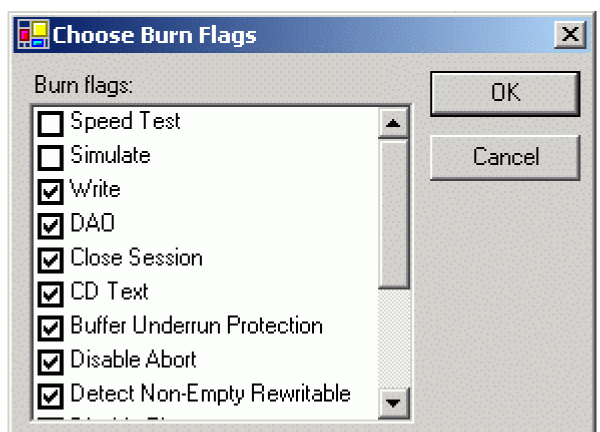


6.9. Burn Flags

The burn flags dialog allows changing the settings for disc compilations.

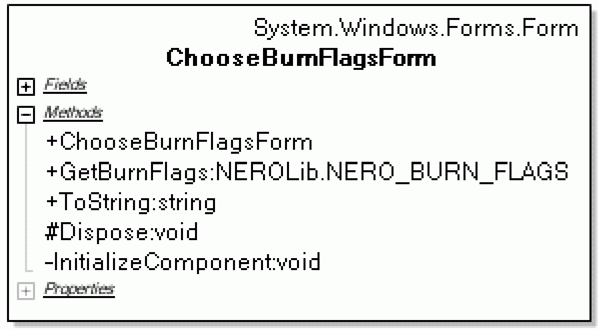
The dialog is used by

- BurnIsoAudioForm
- BurnImageForm
- NVAPIForm
- BurnDiscCopyForm



The public GetBurnFlags function provides the caller with the state of the selected burn flags at any time.

The ToString function returns a textual representation of the set of burn flags.

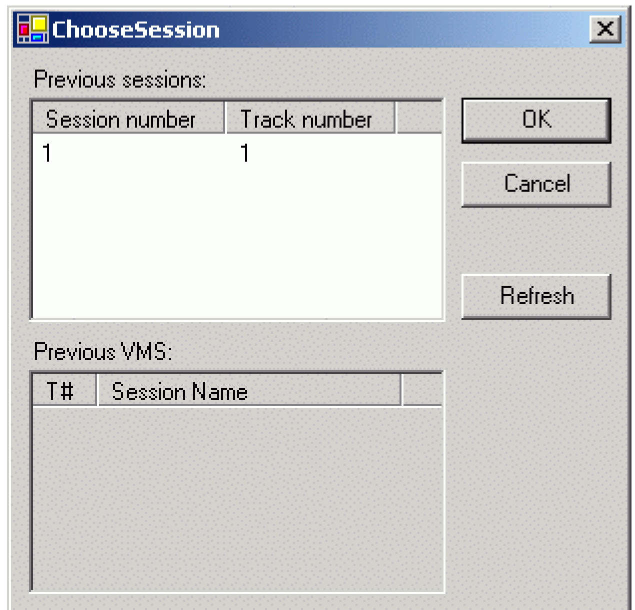


6.10. Choose Session

The choose session dialog provides an option for referencing an existing session from a new track.

Regular ISO sessions and Virtual Multi Sessions are supported.

This dialog is used by the BurnIsoAudioForm class.

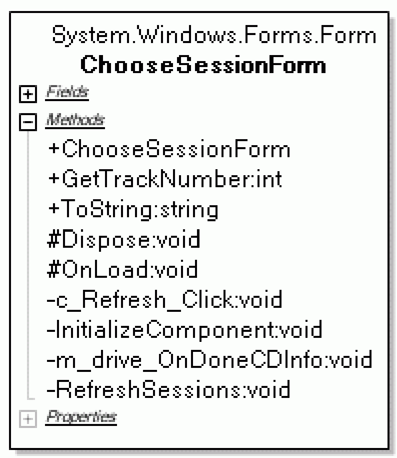


The RefreshSession method subscribes to the disc info event and calls the CDInfo method of NeroDrive.

When the disc info event fires, the dialog returns to the previous state of all controls, restores the cursor and unsubscribe from the event.

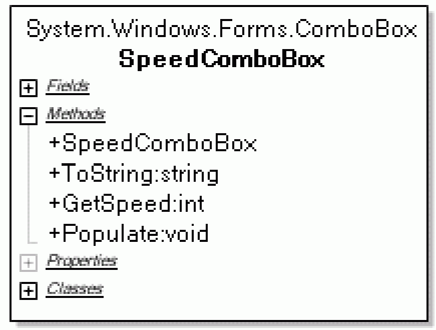
GetTrackNumber is a public function for the outside world to find out which is the selected previous session, if any. It is required that NERO_IMPORT_ISO_TRACK_FLAGS variable is passed by reference so that it is modified if needed.

ToString returns the description of the currently selected session.



6.11. Choose Speed

The choose speed dialog is a generic dialog that contains the available read and write speeds, e.g. to let the user decide to burn with less than maximum speed.



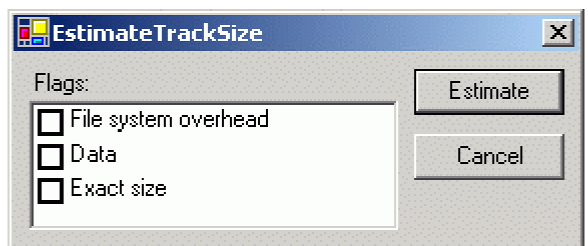
The Populate method fills the combo box with values from the NeroSpeeds parameter.

GetSpeed returns the currently selected speed.

CBItem is a private class that represents an item of the combo box. We use it to store both textual representation and the actual integer value.

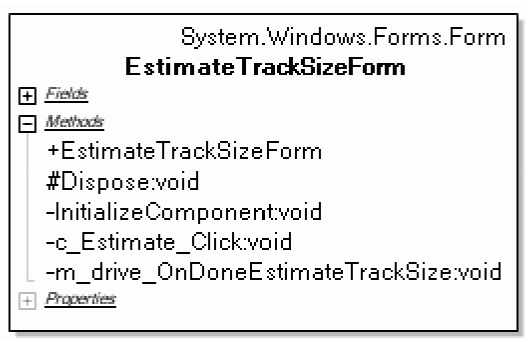
6.12. Estimate Size

This dialog allows setting the parameters for track size estimation.



NeroCOM can estimate the total size of a track including data and overhead for the file system.

Use the flags to specify what exactly has to be taken into account for the calculation.

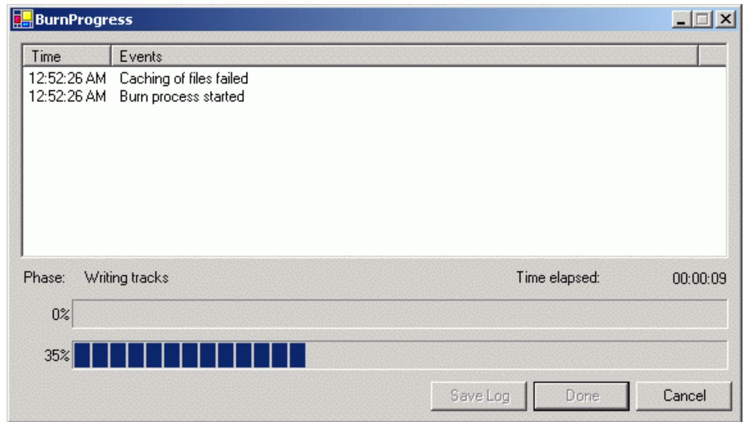


When estimate is requested, the required information is prepared and passed to the EstimateTrackSize function of NeroDrive.

When the estimation is done, the application unsubscribes from the OnDoneEstimate event and displays success or failure.

6.13. Burn Progress

This dialog will be displayed when burning is in progress. After completion, the log file can be saved.

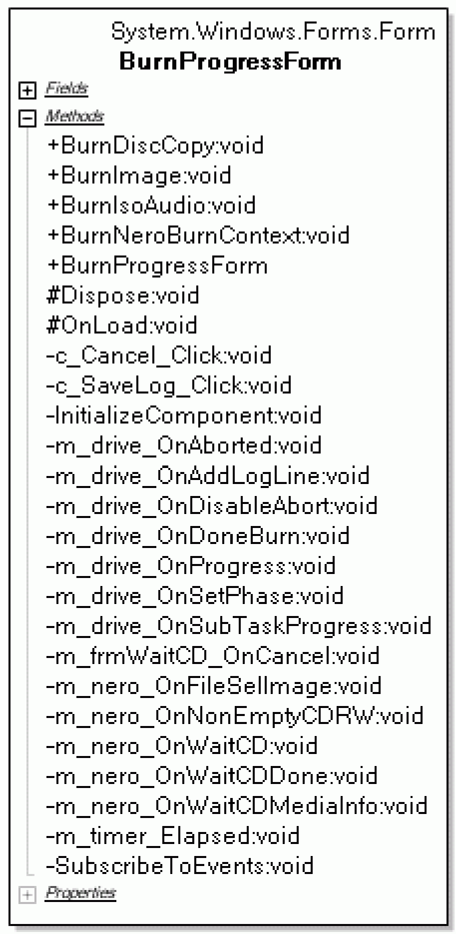


The BurnDiscCopy function burns a copy of a disc in the source drive.

BurnNeroBurnContext burns a *Nero Vision API* project.

BurnImage burns an image file and BurnIsoAudio burns an ISO/Audio disc.

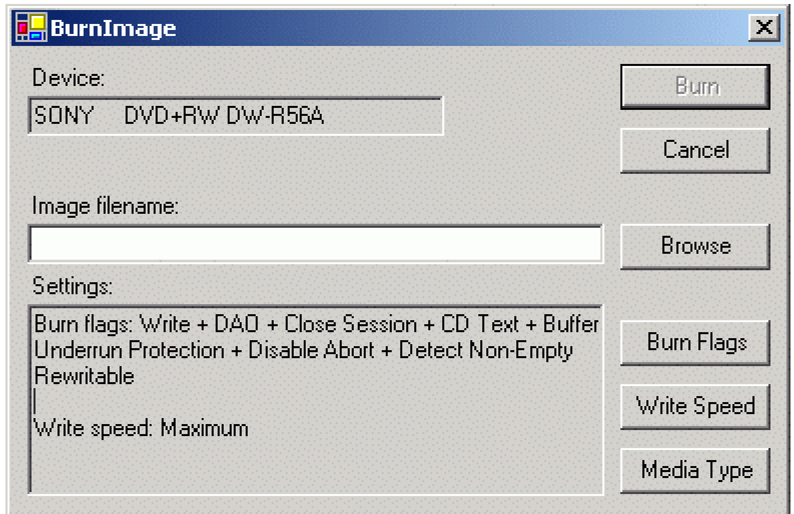
Apart from that, the burn progress form mainly reacts on events that are generated during the burn process.



6.14. Burn Image

This dialog allows the user to burn an existing image file to a writable disc.

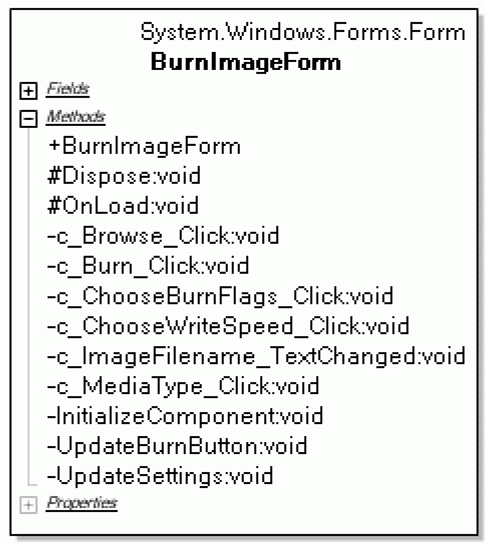
Through additional dialogs, burn flags can be altered, the write speed can be changed and the required media kind selected.



The burn image form gathers the required settings to burn a disc from an image file.

The browse button handler will filter for files with .nrg and .iso extension.

When burning is requested, all parameters are passed, a progress form is created and the burn process is started.

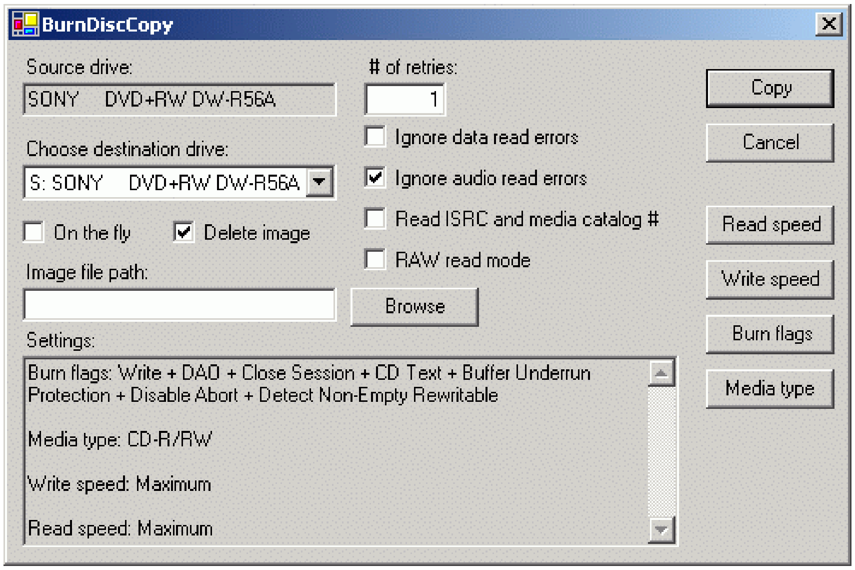


6.15. Burn Disc Copy

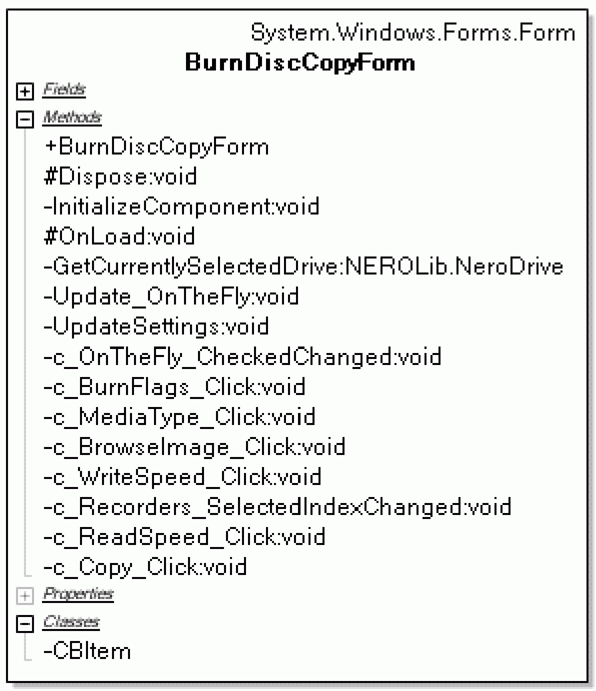
This dialog provides access to disc copy functionality.

Through additional dialogs, burn flags can be altered, the read and write speeds can be changed and the required media kind selected.

In addition, the user can decide to change certain read parameters and, for example, ignore audio read errors.



The burn disc copy form gathers the required settings to burn the copy of a disc.

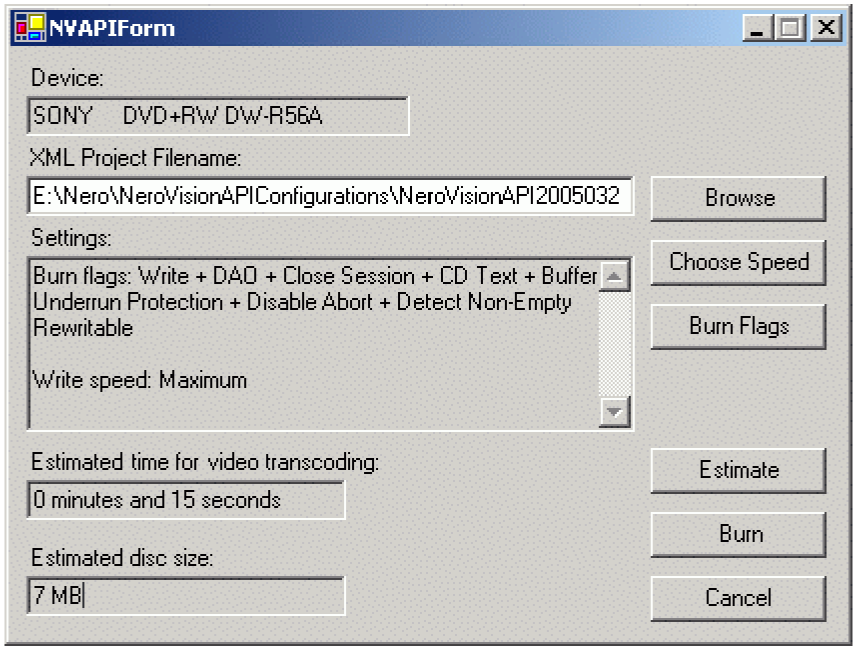


6.16. Burn NVAPI

This dialog offers access to the *Nero Vision API*.

Depending on a project file in XML format, the referenced material will be burned on a disc.

Please refer to the *Nero Vision API* documentation for details about the project description in XML format.

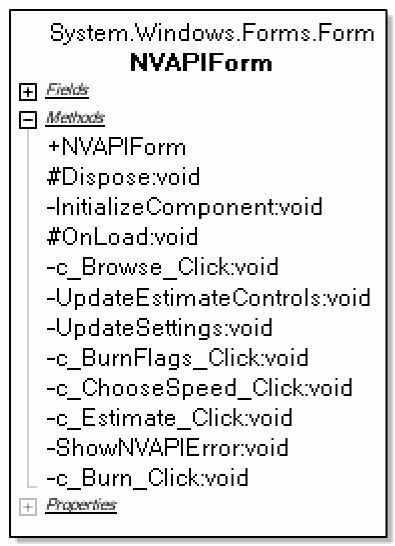


Burning is not enabled unless estimation has been performed first. After estimation, the Burn button is enabled.

When the Burn button is clicked, an estimation form is created which is used for showing transcoding progress as well.

Then the progress form is displayed. INeroBurnContext will be returned as a result.

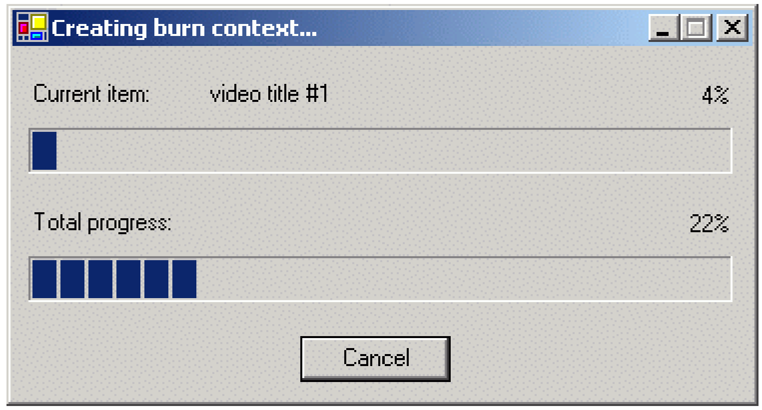
If transcoding went well, the function proceeds with burning and creates a burn progress form.



6.17. NVAPI Estimation

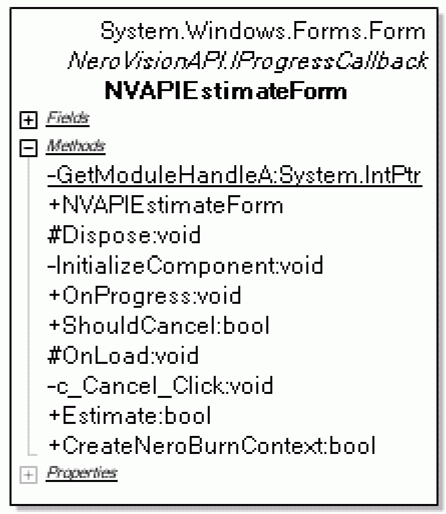
This progress dialog is used for both estimating and transcoding the *Nero Vision API* project.

The dialog contains progress bars for individual items and total progress to provide progress information in a granular form.



The OnLoad function will wait until either estimation or transcoding is completed. OnProgress is used to provide feedback during the process.

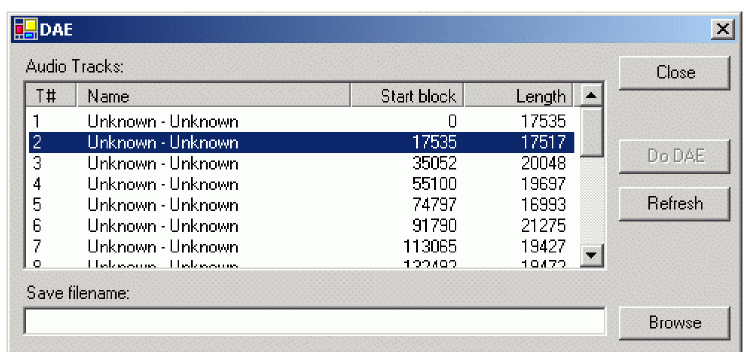
OnProgress is an IProgressCallback method implementation. It is called frequently to report progress.



6.18. DAE

This is the Digital Audio Extraction dialog.

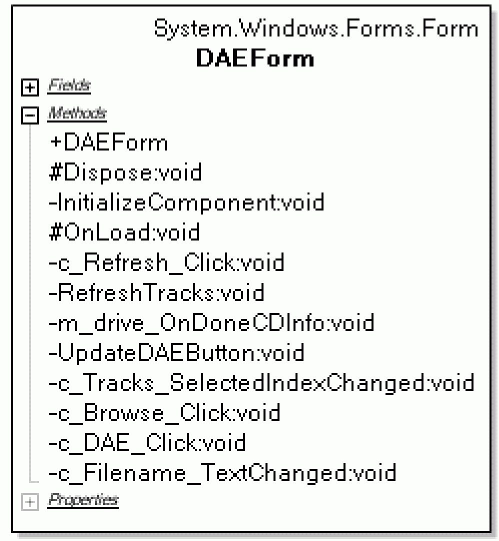
It allows the user to select an audio track to perform DAE with.



Initially, the application will subscribe to the OnDoneCDInfo event, disable controls and show the wait cursor. Then perform the disc info query.

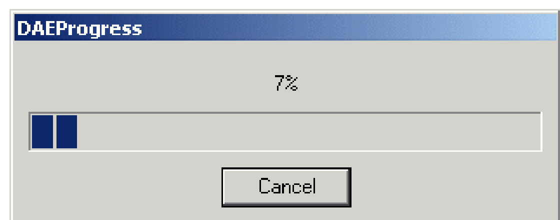
When the OnDoneCDInfo event fires, the track list is populated.

The DAE button will be enabled if an audio track is selected and a file name exists.



6.19. DAE Progress

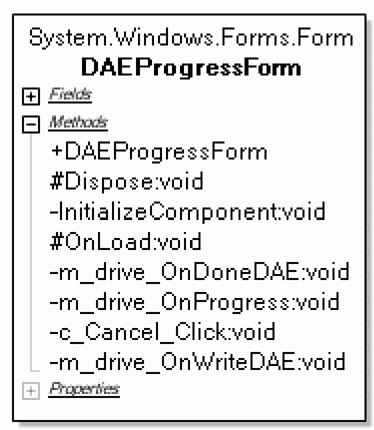
This dialog will appear when the user starts Digital Audio Extraction.



OnLoad subscribes to events that are of interest when performing Digital Audio Extraction. Then the DAE process is started. If filename is not empty, it will be used for storing the audio data in WAV format. If it is empty, the class' events will be called to allow arbitrary operation on the PCM data.

OnProgress will be called periodically to report progress and provide the user with the option to cancel the operation.

The OnDoneDAE handler will be called when the extraction has either completed or terminated with an error.

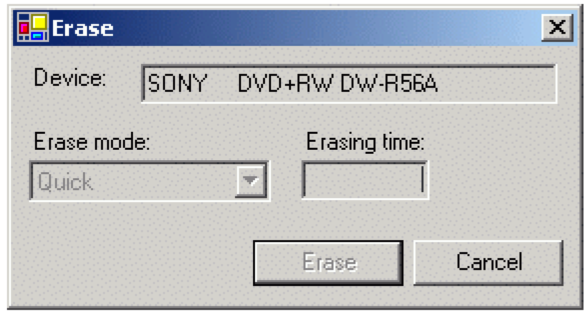


6.20. Erase

This dialog will appear when the user decides to erase a rewritable media.

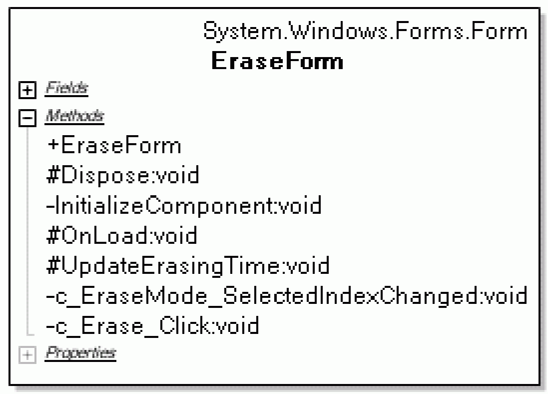
Erasing can be performed in either quick or normal mode.

An estimate for the required erasing time is displayed.



UpdateErasingTime obtains an estimate for the erasing time, depending on the chosen method.

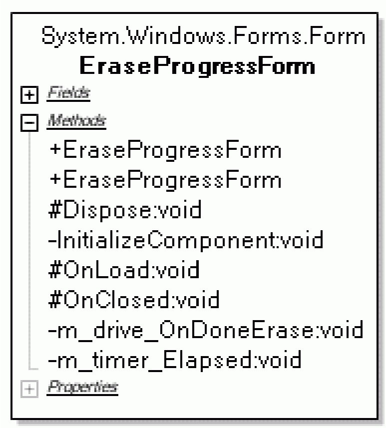
When erasing is requested, the current form is hidden and the erase progress form appears. ShowDialog will start the actual process of erasing the media.



6.21. Erase Progress

The erase progress dialog implementation is straight forward.
The OnLoad method will Subscribe to the OnDoneErase event and start the erase process. As soon as the OnDoneErase event fires, the form is closed.

A timer event is used to update the display of expired and remaining time.

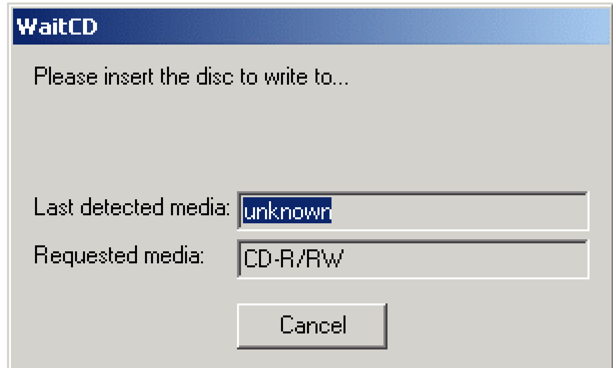


6.22. Wait For CD

This dialog is displays the requested media type when the correct media type is not in the drive.

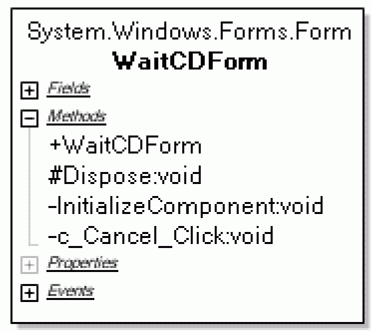
It will be displayed until the correct media is found in the drive or the user clicks cancel.

This dialog is non-modal. Therefore, the calling application has to lock itself against undesired user input.



Upon loading a burn progress dialog, a new WaitCD form is created and the owner will subscribe to its OnCancel event.

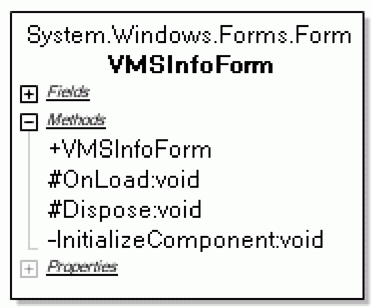
The dialog will be closed will be closed on Cancel, which also terminates the current burn operation, or when the owner is notified of insertion of the requested media.



6.23. VMS Information

The Virtual Multi Session information form is used to obtain and display VMS information.

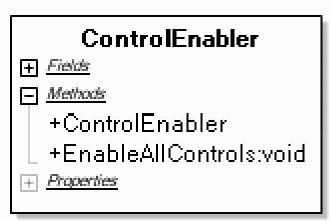
The OnLoad function will add each VMS entry to the list view.



6.24. Control Enabler

The control enabler class allocates an array big enough to store enable states for all controls on a form.

It stores the states of all controls and disables each of them accordingly.



7. *NeroCOM* Enumerations and Objects

7.1. Enumerations

There are a few enumeration types that are used to exchange data between *NeroCOM* and applications.

7.1.1. AccessMode

The enum is used when creating reader or writer interfaces from devices.

Enumerator	Description
eIllegalAccessMode	Illegal Access mode.
eManagedMRW	Use this for defective managed MRW mode for all media types.
eNoWriting	Use this to instantiate an INeroFileSystemBlockAccess object for read-only access.
ePacketWriting	Use this for DVD+RW, DVD-RW, CD-RW media in non-MRW mode.
eRawMRW	Use this for raw MRW mode (defective management turned off).

7.1.2. ImageAccessMode

Used when creating a block access interface from an image.

Enumerator	Description
eIAIllegalAccessMode	Illegal Access mode.
eIAReadOnly	Read only access.
eIAReadWrite	Read and write access.

7.1.3. NERO_ACCESS_TYPE

This enum will be used when requesting available speeds from a device.

Enumerator	Description
NERO_ACCESSTYPE_READ	Read access.
NERO_ACCESSTYPE_WRITE	Write access.

7.1.4. NERO_AUDIO_PROBLEMS

This type is used in the OnAudioProblems event of the Nero object.

Enumerator	Description
NERO_AUP_COPYPROT_SETTINGS	Problem with copyright settings.
NERO_AUP_FIRST_TR_PAUSE	First track must have 2-3 seconds pause.
NERO_AUP_INDEX_SETTINGS	Problem with audio index settings.
NERO_AUP_ISRC_SETTINGS	Problem with ISRC settings.

Enumerator	Description
NERO_AUP_NO_PROBLEM	No problems found.
NERO_AUP_PAUSE_SETTINGS	Problem with audio pause settings (tracks > 1).

7.1.5. NERO_AUDIO_TRACK_TYPE

This type is used in the TrackType member of the NeroAudioTrack object.

Enumerator	Description
NERO_AUDIO_TRACK_FILE_MP3	Track data comes from an MP3 file.
NERO_AUDIO_TRACK_FILE_WAV	Track data comes from a WAVE file.
NERO_AUDIO_TRACK_FILE_WMA	Track data comes from a WMA file.
NERO_AUDIO_TRACK_THROUGH_EVENT	Track data is provided through events.

7.1.6. NERO_BAO_FLAGS

Burn-at-once flags.

Enumerator	Description
NBAOF_DEFAULT	Default value.
NBAOF_CANCELED	User canceled operation.
NBAOF_FAILED	An operation has failed.

7.1.7. NERO_BURN_ERROR

This type is used in the OnDoneBurn event of the NeroDrive object.

Enumerator	Description
NERO_BURN_DRIVE_NOT_ALLOWED	Operation not allowed on that drive.
NERO_BURN_FAILED	Burn operation failed.
NERO_BURN_FUNCTION_NOT_ALLOWED	A particular operation is not allowed.
NERO_BURN_INVALID_DRIVE	The drive is not valid.
NERO_BURN_OK	No problems.
NERO_BURN_UNKNOWN_CD_FORMAT	The format of the media could not be determined.
NERO_BURN_USER_ABORT	User aborted burn operation.

7.1.8. NERO_BURN_FLAGS

This type is used in the burn methods of the NeroDrive object.

Enumerator	Description
NERO_BURN_FLAG_BOOKTYPE_DVDROM	If this flag is set, the booktype of a burned DVD will be set to DVDROM
NERO_BURN_FLAG_BUF_UNDERRUN_PROT	Enable safer burn mode.
NERO_BURN_FLAG_CD_TEXT	Write CD text - will be ignored if not supported by drive.
NERO_BURN_FLAG_CD_TEXT_IS_JAPANESE	If this flag is set with NERO_BURN_FLAG_CD_TEXT, then the CD Text is treated as Japanese CD Text.
NERO_BURN_FLAG_CLOSE_SESSION	Only close the session and not the whole disc.
NERO_BURN_FLAG_DAO	Write in DAO.
NERO_BURN_FLAG_DETECT_NON_EMPTY_CDRW	The OnNonEmptyCDRW event will be fired when trying to burn onto a non empty CDRW.
NERO_BURN_FLAG_DISABLE_ABORT	The OnAbort event will be fired.
NERO_BURN_FLAG_DISABLE_EJECT	CD will not be ejected at the end of the burn process.
NERO_BURN_FLAG_DVDP_BURN_30MM_AT_LEAST	DVD+R/RW high compatibility mode (at least 1GB will be written).
NERO_BURN_FLAG_SIMULATE	Simulate writing before actually writing.
NERO_BURN_FLAG_SPEED_IN_KBS	Interpret the dwSpeed as KB/s instead of multiple of 150 KB/s.
NERO_BURN_FLAG_SPEED_TEST	Test speed of source first.
NERO_BURN_FLAG_VERIFY	Verify file system after writing. Works for ISO only.
NERO_BURN_FLAG_WRITE	Really write at the end.

7.1.9. NERO_BURN_OPTIONS

This type is used in the NeroISOTrack and NeroFileSystemDescContainer objects.

Enumerator	Description
NERO_BURN_OPTION_CREATE_HFS_FS	Not yet available.
NERO_BURN_OPTION_CREATE_ISO_FS	Create an ISO File System Track.
NERO_BURN_OPTION_CREATE_UDF_FS	Create an Universal Disk Format File System Track.
NERO_BURN_OPTION_DVDVIDEO_REALLOC	Perform reallocation of files in the VIDEO_TS directory.
NERO_BURN_OPTION_RELAX_JOLIET	Relax joliet filename length limitations and allow a maximum of 109 characters per filename.

Enumerator	Description
NERO_BURN_OPTION_USE_ALLSPACE	Use all space available on the medium for the volume to be created. Supported for DVD+-RW only.
NERO_BURN_OPTION_USE_JOLIET	Create a Joliet Track.
NERO_BURN_OPTION_USE_MODE2	Create a Mode 2 Track.
NERO_BURN_OPTION_USE_ROCKRIDGE	Create a RockRidge Track.

7.1.10. NERO_CAPABILITIES

This type is used in the Capabilities property of the NeroDrive object.

Enumerator	Description
NERO_CAP_ALLOW_CHANGE_BOOKTYPE	DVD recorder can change booktype of burned medium.
NERO_CAP_ALLOWED	The drive can only be used if this flag is set.
NERO_CAP_BUF_UNDERRUN_PROT	Drive has a buffer underrun protection feature (not necessarily Burn Proof).
NERO_CAP_DAO	Can write in DAO.
NERO_CAP_DAO_WRITE_CD_TEXT	Writes CD text in DAO; never supported in TAO.
NERO_CAP_DVDPLUSVR_SUPPORTED	This recorder can write DVD+VR.
NERO_CAP_IDE_BUS	Drive is an IDE device.
NERO_CAP_IMAGE_RECORDER	Drive is the image recorder.
NERO_CAP_READ_CD_TEXT	Can read CD text.
NERO_CAP_SCSI_BUS	Drive is an SCSI device.
NERO_CAP_UNDETECTED	No driver could be assigned to this drive.
NERO_CAP_VARIABLE_PAUSES_IN_TAO	Variable pauses in TAO are allowed.

7.1.11. NERO_CDINFO_FLAGS

This type is used in the CDInfo property of the NeroDrive object.

Enumerator	Description
NERO_READ_CD_TEXT	CD info in CD Text format.
NERO_READ_ISRC	CD info in International Standard Recording Code.

7.1.12. NERO_COMMNODE_TYPE

Communication node types for the robot interface.

Enumerator	Description
NCT_PORT_SERIAL	Serial interface.
NCT_PORT_PARALLEL	Parallel port.
NCT_PORT_USB	USB interface.

7.1.13. NERO_DEVICE_OPTION

This type is used in NeroDrive's GetDeviceOption and SetDeviceOption methods.

Enumerator	Description
NERO_DEVICE_OPTION_BOOKTYPE_ DVDROM	Change the booktype of a DVD+R and DVD+RW for subsequent writes until next power cycle to DVD-ROM.

7.1.14. NERO_DRIVE_ERROR

This type is used by NeroDrive's UpdateDeviceInfo method.

Enumerator	Description
NDE_DISC_NOT_PRESENT	No media in drive, status of tray unknown.
NDE_DISC_NOT_PRESENT_TRAY_CLOSED	No media - tray closed.
NDE_DISC_NOT_PRESENT_TRAY_OPEN	No media - tray open.
NDE_DRIVE_IN_USE	Drive cannot be locked; maybe another application uses this drive at the moment.
NDE_DRIVE_NOT_READY	Drive is not ready.
NDE_GENERIC_ERROR	Error, not handled with other enums.
NDE_NO_DRIVE	The given device is not available. Probably removed by the user (USB/Firewire).
NDE_NO_ERROR	No error occurred/ drive is ready.

7.1.15. NERO_DRIVECHANGE_RESULT

This type is used by the OnDriveStatusChanged event of the Nero object.

Enumerator	Description
NDR_DRIVE_ADDED	A drive has been added.
NDR_DRIVE_REMOVED	A drive has been removed.

7.1.16. NERO_DRIVESTATUS_RESULT

This type is used by NeroDrive's OnDriveStatusChanged event.

Enumerator	Description
NDR_DISC_INSERTED	Disc has been inserted.
NDR_DISC_REMOVED	Disc has been removed.
NDR_DRIVE_IN_USE	The drive is in use.
NDR_DRIVE_NOT_IN_USE	The drive is not in use.

7.1.17. NERO_DRIVESTATUS_TYPE

This type is used in NeroDrive's EnableStatusCallback method.

Enumerator	Description
NDT_DISC_CHANGE	The disc in the drive has been changed. Warning: This change notification is based on Windows notifying about media changes. If an application has disabled this notification, the callback will not be called. If you want to be sure to recognize all media changes, you should use timer events and query the drive status.
NDT_IN_USE_CHANGE	The in-use status of the drive has been changed.

7.1.18. NERO_ENTRY_ERROR

This type is used by some of the File System Objects.

Enumerator	Description
NERO_ENTRY_ERROR_NOT_A_FILE	This entry is not a file.
NERO_ENTRY_FEATURE_NOT_AVAILABLE	This feature is not available for this file system type.
NERO_ENTRY_INTERFACE_ERROR	The overridden function has tried to get another interface for one object and has failed.
NERO_ENTRY_NO_ERROR	No error.
NERO_ENTRY_NOT_AVAILABLE	The content of this file cannot be requested at all.
NERO_ENTRY_SEQUENCING_ERROR	Indicates that the content for this file may not be requested at this moment.

7.1.19. NERO_ERASE_MODE

This type is used in NeroDrive's EraseDisc method.

Enumerator	Description
NERO_ERASE_MODE_DEFAULT	Use the default mode for the recorder.
NERO_ERASE_MODE_DISABLE_EJECT	CD will not be ejected at the end of the erasing, even if this is recommended for the selected recorder.
NERO_ERASE_MODE_EJECT_AFTER_ERASE	Eject disc after erasing, no matter if this is recommended for the recorder or not.

7.1.20. NERO_ESTIMATE_FLAGS

Use the flags to specify what exactly has to be taken into account for track size estimation.

Enumerator	Description
NETS_DATA	Calculate data size.
NETS_EXACT_SIZE	Calculate exactly. If this option is specified, file system overhead as well as file data are taken into account. The optional fields of the NERO_ESTIMATETRACKSIZE_OPTIONS structure need to be filled out.
NETS_FILESYSTEM_OVERHEAD	Calculate file system overhead.

7.1.21. NERO_FREESTYLE_SOURCE_TYPE

This type is used in the SourceType property of the NeroFreestyleTrack object.

Enumerator	Description
NERO_FREESTYLE_SOURCE_EVENT	Data will be provided by events.
NERO_FREESTYLE_SOURCE_FILE	Data will be provided as a file.

7.1.22. NERO_FREESTYLE_TRACK_TYPE

This type is used in the TrackType property of NeroFreestyleTrack.

Enumerator	Description
NERO_FREESTYLE_TRACK_AUDIO	2352 Bytes per sector, standard audio track.
NERO_FREESTYLE_TRACK_MODE1	2048 Bytes per sector data track.
NERO_FREESTYLE_TRACK_MODE2_FORM1	2048 Bytes per sector, used for multisession.

7.1.23. NERO_IMPORT_DATA_TRACK_RESULT

This type is used by the OnDoneImport2 event of the NeroDrive object.

Enumerator	Description
NIDTR_DRIVE_ERROR	Drive error.
NIDTR_GENERIC_ERROR	Undefined error.
NIDTR_INVALID_FS	Error while reading from the disc. Parts of the file system may have been imported nevertheless.
NIDTR_NO_ERROR	No error.
NIDTR_READ_ERROR	Error while reading from the disc. Parts of the file system may have been imported nevertheless.

Enumerator	Description
NIDTR_INVALID_FS	Errors in the file system on the disc.
NIDTR_INVALID_TRACKNUMBER	The given track number is invalid.

7.1.24. NERO_IMPORT_ISO_TRACK_FLAGS

This type is used as a parameter of the ImportIsoTrack method of the NeroDrive object.

Enumerator	Description
NERO_IMPORT_ISO_ONLY	Import ISO session only.
NERO_IMPORT_ROCKRIDGE	Will be ignored, RockRidge is now always imported if present.
NERO_IMPORT_UDF	Import UDF Session.
NERO_PREFER_ROCKRIDGE	Will be ignored.

7.1.25. NERO_LAST_ERRORS_FLAGS

This type is used as a parameter of the ImportIsoTrack method of the NeroDrive object.

Enumerator	Description
NERO_LAST_ERRORS_FLAG_ALL	Also include errors which do not contain a description to be shown in the GUI.
NERO_LAST_ERRORS_FLAG_REPORT	Format the errors as in the *NeroAPI* error log.

7.1.26. NERO_MAJOR_PHASE

This type is used by the OnMajorPhase event of the NeroDrive object.

Enumerator	Description
NERO_PHASE_ABORT_CACHE	Caching aborted.
NERO_PHASE_ABORT_PREPARE_CD	Preparing CD aborted.
NERO_PHASE_ABORT_SIMULATE	Simulation aborted.
NERO_PHASE_ABORT_SIMULATE_NOSPD	nospd simulation aborted.
NERO_PHASE_ABORT_TEST	Testing aborted.
NERO_PHASE_ABORT_WRITE	Writing aborted.
NERO_PHASE_ABORT_WRITE_NOSPD	nospd writing aborted.
NERO_PHASE_BUP_ACTIVATED	Buffer Underrun Protection activated.
NERO_PHASE_CONTINUE_FORMATTING	Formatting is continued.
NERO_PHASE_DONE_CACHE	Done with caching.
NERO_PHASE_DONE_PREPARE_CD	Done with preparing the media.
NERO_PHASE_DONE_SIMULATE	Done with simulation.
NERO_PHASE_DONE_SIMULATE_NOSPD	Done with simulation.
NERO_PHASE_DONE_TEST	Done testing.
NERO_PHASE_DONE_WRITE	Done writing.
NERO_PHASE_DONE_WRITE_NOSPD	Done writing nospd.
NERO_PHASE_DVDVIDEO_DETECTED	DVD Video detected.

Enumerator	Description
NERO_PHASE_DVDVIDEO_REALLOC_COMPLETED	DVD Video reallocation completed.
NERO_PHASE_DVDVIDEO_REALLOC_FAILED	DVD Video reallocation failed.
NERO_PHASE_DVDVIDEO_REALLOC_NOTNEEDED	DVD Video reallocation not required.
NERO_PHASE_DVDVIDEO_REALLOC_STARTED	DVD Video reallocation started.
NERO_PHASE_ENCODE_VIDEO	Encode video.
NERO_PHASE_FAIL_CACHE	Caching failed.
NERO_PHASE_FAIL_PREPARE_CD	Preparing media failed.
NERO_PHASE_FAIL_SIMULATE	Simulation failed.
NERO_PHASE_FAIL_SIMULATE_NOSPD	nospd simulation failed.
NERO_PHASE_FAIL_TEST	Testing failed.
NERO_PHASE_FAIL_WRITE	Writing failed.
NERO_PHASE_FAIL_WRITE_NOSPD	nospd writing failed.
NERO_PHASE_FORMATTING_FAILED	Formatting failed.
NERO_PHASE_FORMATTING_SUCCESSFUL	Formatting completed successfully.
NERO_PHASE_PREPARE_CD	Preparing media failed.
NERO_PHASE_PREPARE_ITEMS	Preparing items failed.
NERO_PHASE_SEAMLESSLINK_ACTIVATED	Seamlesslink activated.
NERO_PHASE_START_CACHE	Caching started.
NERO_PHASE_START_SIMULATE	Simulation started.
NERO_PHASE_START_SIMULATE_NOSPD	Simulation started.
NERO_PHASE_START_TEST	Testing started.
NERO_PHASE_START_WRITE	Writing started.
NERO_PHASE_START_WRITE_NOSPD	Writing started.
NERO_PHASE_UNSPECIFIED	No phase specified.
NERO_PHASE_VERIFY_ABORTED	Verification aborted.
NERO_PHASE_VERIFY_COMPILATION	Verifying a compilation.
NERO_PHASE_VERIFY_END_FAIL	Verification failed.
NERO_PHASE_VERIFY_END_OK	Verification successful.
NERO_PHASE_ENCODE_VIDEO	Encoding video.
NERO_PHASE_SEAMLESSLINK_ACTIVATED	Activated seamless link.
NERO_PHASE_BUP_ACTIVATED	Buffer underrun protection activated.
NERO_PHASE_START_FORMATTING	Formatting started.
NERO_PHASE_CONTINUE_FORMATTING	Formatting is being continued.
NERO_PHASE_FORMATTING_SUCCESSFUL	Formatting successful.
NERO_PHASE_FORMATTING_FAILED	Formatting failed.
NERO_PHASE_PREPARE_CD	Preparing disc.
NERO_PHASE_DONE_PREPARE_CD	Done preparing disc.
NERO_PHASE_FAIL_PREPARE_CD	Failed preparing disc.
NERO_PHASE_ABORT_PREPARE_CD	Aborted preparing disc.
NERO_PHASE_DVDVIDEO_DETECTED	DVD-Video detected.
NERO_PHASE_DVDVIDEO_REALLOC_STARTED	DVD-Video reallocation started.

Enumerator	Description
NERO_PHASE_DVDVIDEO_REALLOC_ COMPLETED	DVD-Video reallocation completed.
NERO_PHASE_DVDVIDEO_REALLOC_ NOTNEEDED	DVD-Video reallocation not required.
NERO_PHASE_DVDVIDEO_REALLOC_FAILED	DVD-Video reallocation failed.
NERO_PHASE_DRM_CHECK_FAILURE	Digital Rights Management check failed.
NERO_PHASE_START_WRITE_DATAFILES	Burn-at-once phase. Writing data files started.
NERO_PHASE_DONE_WRITE_DATAFILES	Burn-at-once phase. Writing data files completed.
NERO_PHASE_FAIL_WRITE_DATAFILES	Burn-at-once phase. Writing data files failed.
NERO_PHASE_WARN_WRITE_DATAFILES	Burn-at-once phase. Problem writing data files.
NERO_PHASE_START_BAO_FINALIZE	Burn-at-once phase. Finalization started.
NERO_PHASE_FAIL_BAO_FINALIZE	Burn-at-once phase. Finalization failed.
NERO_PHASE_DONE_BAO_FINALIZE	Burn-at-once phase. Finalization completed.
NERO_PHASE_FAIL_BAO_PREPARE	Burn-at-once phase. Preparing finalization.
NERO_PHASE_FAIL_BAO_WRITEFILE	Burn-at-once phase. Writing file failed.
NERO_PHASE_BURN_LAYER_1	Burning Layer 1.
NERO_PHASE_BURN_LAYER_2	Burning Layer 2.

7.1.27. NERO_MEDIA_TYPE

This type is used by various objects.

Enumerator	Description
NERO_MEDIA_CD	CD-R/RW.
NERO_MEDIA_CDR	CD-R.
NERO_MEDIA_CDROM	CD-ROM (non writable).
NERO_MEDIA_CDRW	CD-RW.
NERO_MEDIA_DDCD	DDCD-R/RW.
NERO_MEDIA_DVD_ANY	Any DVD media.
NERO_MEDIA_DVD_M	DVD-R/RW.
NERO_MEDIA_DVD_M_R	DVD-R.
NERO_MEDIA_DVD_M_RW	DVD-RW.
NERO_MEDIA_DVD_P	DVD+RW.
NERO_MEDIA_DVD_P_R	DVD+R.
NERO_MEDIA_DVD_P_RW	DVD+RW.
NERO_MEDIA_DVD_RAM	DVD-RAM.
NERO_MEDIA_DVD_ROM	DVD-ROM (non writable).
NERO_MEDIA_FPACKET	Fixed Packet writing.
NERO_MEDIA_HDB	HD-Burn.
NERO_MEDIA_ML	ML (Multi Level disc).
NERO_MEDIA_MRW	Mt. Rainier.
NERO_MEDIA_NO_CDR	Exclude CD-R.
NERO_MEDIA_NO_CDRW	Exclude CD-RW.
NERO_MEDIA_NO_DVD_M_R	Exclude DVD-R.
NERO_MEDIA_NO_DVD_M_RW	Exclude DVD-RW.
NERO_MEDIA_NO_DVD_P_R	Exclude DVD+R.
NERO_MEDIA_NO_DVD_P_RW	Exclude DVD+RW.
NERO_MEDIA_NONE	No media present.
NERO_MEDIA_PACKETW	A bit mask for packet writing.
NERO_MEDIA_VPACKET	Variable Packet writing.

7.1.28. NERO_MEDIUM_FLAGS

This type is used as part of disc information.

Enumerator	Description
NCDIMF_HDB_SUPPORTED	The media supports HDBURN.
NCDIMF_VIRTUALMULTISESSION	The media is a virtual multisession media. Note: This flag only tells you, that if multisession is written, VMS is used. But not that this medium contains multiple sessions.

7.1.29. NERO_RESPONSE

This type is used by all events of the Nero object, which require a response.

Enumerator	Description
NERO_RETURN_CONTINUE	Continue at own risk.
NERO_RETURN_EXIT	Exit application, stop current process.
NERO_RETURN_FALSE	False/no.
NERO_RETURN_INSTALL_DRIVER	Install IO driver which temporarily disables auto insert.
NERO_RETURN_OFF_RESTART	Change auto insert and restart Windows.
NERO_RETURN_ON_RESTART	Turn on disconnect and restart Windows.
NERO_RETURN_RESTART	Do not change disconnect option and restart Windows.
NERO_RETURN_TRUE	True/yes.

7.1.30. NERO_ROBO_FLAG

This enumeration is used when setting runtime options for the robot, i.e. settings that the robot will not remember beyond the current session.

.Enumerator	Description
NERO_ROBO_FLAG_CLEANUP	Set this value in order to have the robot perform cleanup functions for itself. This might include removing any remaining discs from the trays etc. Note that the resulting behavior may sometimes look like a bad control flow. For example, the robot might try to remove discs when there are none. Therefore, you should not use this option unless you are cleaning up after a hard server crash or have another good reason to do so.
NERO_ROBO_INSERTCD_RETRIES	This flag specifies how often the *NeroAPI* shall try to insert another disc from the input tray if the disc in the drive is not writable. The current default value for the number of retries is 5, a value of 0 means trying indefinitely.

7.1.31. NERO_SCSI_DEVTYPE

This is the type of NeroDrive’s DevType property.

Enumerator	Description
NERO_SCSI_DEVTYPE_CDROM	Read only. A CD-ROM drive.
NERO_SCSI_DEVTYPE_UNKNOWN	Type information not available.
NERO_SCSI_DEVTYPE_UNSUPPORTED_ WORM	Can write but is not supported by *NeroAPI*.
NERO_SCSI_DEVTYPE_WORM	Write once. A CD-burner.

7.1.32. NERO_TEXT_TYPE

This type is used by NeroDrive's OnAddLogLine event.

Enumerator	Description
NERO_TEXT_DRIVE	A message concerning a CD-ROM drive or recorder.
NERO_TEXT_EXCLAMATION	Important information.
NERO_TEXT_FILE	A message concerning a file.
NERO_TEXT_INFO	Informative text.
NERO_TEXT_QUESTION	A question which requires an answer.
NERO_TEXT_STOP	Some operation stopped prematurely.
NERO_TEXT_UNSPECIFIED	No type specified.

7.1.33. NERO_TRACK_TYPE

This type is the type of NeroTrack's TrackType property.

Enumerator	Description
NERO_TT_AUDIO	Audio track.
NERO_TT_DATA	Data track.
NERO_TT_UNKNOWN	Unknown track type.

7.1.34. NERO_VIDEO_ITEM_TYPE

Type of the ItemType property of the NeroVideoItem property.

Enumerator	Description
NERO_JPEG_ITEM	Item is of JPEG type.
NERO_MPEG_ITEM	Item is of MPEG type.
NERO_NONENCODED_VIDEO_ITEM	The source file name will be an AVI file which will be encoded into MPG by *NeroAPI*.

7.1.35. NERO_VIDEO_RESOLUTION

Resolution type for video content.

Enumerator	Description
NERO_VIDEO_RESOLUTION_NTSC	NTSC resolution.
NERO_VIDEO_RESOLUTION_PAL	PAL resolution.

7.1.36. NERO_WAIT_FOR_DISC_FLAGS

This type is used when waiting for a media.

Enumerator	Description
NERO_REQUIRE_EMPTY_DISC	Media must be empty.

7.1.37. NERO_WAITCD_TYPE

This type is used by the Nero object's OnWaitCD event.

Enumerator	Description
NERO_WAITCD_AUTOEJECTLOAD	"Please do not remove the disc! Your recorder requires this eject between simulation and burning. The disc will be reloaded automatically before continuing with burning..."
NERO_WAITCD_DISCINFOS_FAILED	"Disc analysis failed. The error log contains more information about the reason."
NERO_WAITCD_EMPTYCD	"The disc is not empty. Please insert an empty disc."
NERO_WAITCD_MAX	"unknown NERO_WAITCD_TYPE"
NERO_WAITCD_MEDIUM_UNSUPPORTED	"The recorder does not support this type of media! Please insert a correct disc to write to..."
NERO_WAITCD_MULTI_REINSERT	"Please do not remove the disc! Your recorder requires this eject between simulation and burning. Please reinsert the same Multisession disc..."
NERO_WAITCD_MULTISESSION	"Nero is checking for the disc, please wait ... To burn this multisession compilation you need the disc that contains the previous backup sessions. Please insert this disc if you haven't done it before."
NERO_WAITCD_MULTISESSION_SIM	"To simulate this multisession compilation you need the disc, which contains the previous backup sessions. Please insert this disc. (Nothing will be written on disc)."
NERO_WAITCD_NEWORIGINAL	"The disc is blank, invalid nor a multisession disc. Please insert original disc."
NERO_WAITCD_NEXTCD	"Please remove the disc and insert the next recordable disc to write to..."
NERO_WAITCD_NOTENOUGHSPACE	"There is not enough space to burn this compilation onto this disc. Please insert another disc that provides more space..."
NERO_WAITCD_ORIGINAL	"Please insert the original disc."
NERO_WAITCD_REINSERT	"Please do not remove the disc! Your recorder requires this eject between simulation and burning. Please reinsert the disc..."
NERO_WAITCD_SIMULATION	"Please insert a disc to use during simulation... (Nothing will be written on the disc.)"
NERO_WAITCD_SIMULATION_EMPTY	"Please insert an empty disc to use during simulation... (Nothing will be written on the disc)."

Enumerator	Description
NERO_WAITCD_WAITING	"--- Accessing disc, please wait ---"
NERO_WAITCD_WRITE	"Please insert the disc to write to..."
NERO_WAITCD_WRITE_EMPTY	"Please insert an empty disc to write to..."
NERO_WAITCD_WRITEPROTECTED	"This disc is not writable. Please insert a writable disc..."
NERO_WAITCD_WRITEWAVE	"The disc is blank. Please insert original disc…"

7.1.38. NEROAPI_ROBO_ERROR

Robot interface errors.

Enumerator	Description
NEROAPI_ROBO_OK	No error.
NEROAPI_ROBO_NOTFOUND	Robot not found.
NEROAPI_ROBO_NOTSUPPORTED	Robot not supported.

7.1.39. NERODLG_ICON_TYPE

Determines what kind of icon to display with a user dialog.

Enumerator	Description
NDIT_ERROR	An error icon.
NDIT_INFO	An info icon.
NDIT_QUESTION	A question icon.
NDIT_WARNING	A warning icon.

7.1.40. NERODLG_MESSAGE_TYPE

Determines the kind of dialog message to be displayed.

Enumerator	Description
NDMT_ABORTRETRYIGNORE	A dialog with a abort, a retry and a ignore button. Return DLG_RETURN_IGNORE for Ignore, DLG_RETURN_RETRY for Retry and DLG_RETURN_ABORT for Abort.
NDMT_OK	An info dialog with only an OK button. The return value is ignored.
NDMT_OKCANCEL	A dialog with a ok and a cancel button. Return DLG_RETURN_OK for OK and DLG_RETURN_CANCEL for Cancel.
NDMT_RETRYCANCEL	A dialog with a retry and a cancel button. Return DLG_RETURN_RETRY for Retry and DLG_RETURN_CANCEL for Cancel.
NDMT_YESNOCANCEL	A dialog with a yes and a no button. Return DLG_RETURN_YES for Yes and DLG_RETURN_NO for No.

7.1.41. NeroFSBlockAccessExtensionsType

Type of an extension.

Enumerator	Description
etBlankAreaControlExtension	Blank area control extension.
etDVDPRWFormatExtension	DVD-P/RW format extension.
etFragmentReservation	Fragment reservation extension.
etHDPartitionInfo	HD partition info extension.
etHDPartitionLocker	HD partition locker extension.
etHDUsedBlockAccessExtention	HD use blocked access extension.
etIllegalBlockAccessExtension	Illegal block access extension.
etMediumCDStamp	Medium CD stamp extension.
etMRWReadDefectiveManagementInfo	MRW read defective management info extension.
etSectorMappingControlExtension	Sector mapping control extension.
etSectorPatchControlExtension	Sector patch control extension.
etSessionControl	Session control extension.
etMediumInfo	Medium info extension.
etUDFImageEditExtension	UDF image edit extension.

7.1.42. NeroFSError

This type is used to report the results of packet reading/writing operations.

Enumerator	Description
errEndOfDir	Deprecated. Should never be returned, to be treated as errOK.
errEndOfFile	See the libc read command for reference.
errFileLocked	The file is locked.
errIllegalArgument	An illegal argument has been passed.
errInternalError	An internal error has occurred.
errInvalidFS	The files system is not valid.
errNoDirectory	It has been attempted to perform a directory operation on an object that is no directory.
errNoFile	It has been attempted to perform a file operation on an object that is no file.
errNotSupported	Operation not supported.
errOK	Operation successful.
errReadError	A read error has occurred.
errWriteError	A write error has occurred.

7.1.43. NeroFSTrackType

Used by the PartitionType member of the NeroFSPartitionInfo object.

Enumerator	Description
vtAudio	Audio Track.
vtData	Data Track.

7.1.44. ROBOMOVENODE

Node type for robot movement.

Enumerator	Description
RMN_INPUT	Input.
RMN_RECORDER	Recorder.
RMN_OUTPUT	Output.
RMN_PRINTER	Printer.
RMN_WASTEBIN	Waste bin.

7.1.45. ROBOUSERMESSAGETYPE

Enumeration type for robot messages to the user.

Enumerator	Description
RUMT_ERROR	Error.
RUMT_WARNING	Warning.
RUMT_QUESTION	Question.
RUMT_HINT	Hint.

7.2. General Objects

7.2.1. IInt64

Type for Int64 representation.

Properties

Property	read-only	Description
HiPart	no	High part.
Property HiPart As Long		
LoPart	no	Low part.
Property LoPart As Long		

7.2.2. Nero

The first object that a client needs to create is **Nero**. This object has a number of events associated with it. These events correspond to *NeroAPI*'s DLG_xxx user dialog callback constants. This is why the Nero object is so important. It is responsible for handling all of the DLG_xxx callbacks that are essential for *NeroAPI* operation.

Properties

Property	read-only	Description
DVDOverburnSize	no	DVD overburn size in blocks. Set 0 to disable.
Property DVDOverburnSize As Long		
EnableOverburn	read descr.	The *NeroAPI* can try to write more than the nominal capacity of a media is this is true. This property can be written before *NeroAPI* initialization, becomes read-only afterwards. Used only as *NeroAPI* initialization parameter.
Property EnableOverburn As Boolean		
ErrorLog	yes	Retrieve the entire *NeroAPI* error log.
Property ErrorLog As String		
LanguageFile	read descr.	Name of the Nero language file. This property can be written before *NeroAPI* initialization, becomes read-only afterwards. Used only as *NeroAPI* initialization parameter.
Property LanguageFile As String		
LastDriveError	yes	Retrieve the last drive error from the *NeroAPI*.
Property ErrorLog As String		

Property	read-only	Description
LastError	yes	Retrieve the last error from the *NeroAPI*.
Property LastError As String		
OverburnSize	read descr.	The overburn size in blocks. This property can be written before *NeroAPI* initialization, becomes read-only afterwards. Used only as *NeroAPI* initialization parameter.
Property OverburnSize As Long		
RuntimeLanguageFile	no	This property can be used to change the language file.
Property RuntimeLanguageFile As String		
TypeNameOfMedia	yes	The name of the current media type.
Property TypeNameOfMedia(lVal As NERO_MEDIA_TYPE) As String		
WriteBufferSize	read descr.	The size of the write buffer in Byte. This property can be written before *NeroAPI* initialization, becomes read-only afterwards. Used only as *NeroAPI* initialization parameter.
Property WriteBufferSize As Long		

Methods

Method	Description
Abort	Can be called at any time to abort the current operation. It is not harmful to call it even when *NeroAPI* has not yet been initialized but it is not very useful either.
Abort()	
APIVersion	Retrieve the *NeroAPI* version number. The version is coded like this: • Major version high part • Major version low part • Minor version high part • Minor version low part
APIVersion(pmajhi As Integer, pmajlo As Integer, pminhi As Integer, pminlo As Integer)	
ClearErrors	Clear errors and log.
Sub ClearErrors()	
CreateBlockAccessFromImage	Create a block access interface for the specified image file. Instead of an image file, you may pass a drive letter here to read from a specific device supported by the operating system.
Function CreateBlockAccessFromImage(Filename As String, ImageAccessMode As ImageAccessMode) As INeroFileSystemBlockWriter	
EnableDriveChangeCallback	Enable the drive change callback.
Sub EnableDriveChangeCallback(bEnable As Boolean)	

GetDiscImageInfo	Retrieve disc image information.
Function GetDiscImageInfo(bstrImageFilename As String) As INeroCDInfo	
GetDrives	The GetDrives function is initialized by the *NeroAPI* upon first invocation. Returns NeroDrives collection that is used to gain access to a particular NeroDrive. It can be called many times to get a different collection of drives.
Function GetDrives(lVal As NERO_MEDIA_TYPE) As NeroDrives	
GetRoboDrivers	This function will return a list of available robot drivers through the INeroRoboDrivers collection.
Function GetRoboDrivers() As INeroRoboDrivers	
LastErrors	This function returns recently logged errors. The number of returned errors is determined by iNum.
Function LastErrors(iNum As Long, Flags As NERO_LAST_ERRORS_FLAGS) As String	

Events

Event	Description
OnAudioProblems	Return NERO_RETURN_TRUE to fix the problems by adapting the track settings. Return NERO_RETURN_FALSE to stop writing.
Event OnAudioProblems(AudioProblems As NERO_AUDIO_PROBLEMS, Response As NERO_RESPONSE)	
OnAutoInsert	Change auto insert and restart Windows.
Event OnAutoInsert(Response As NERO_RESPONSE)	
OnAutoInsertRestart	This event is fired after rebooting within program failed and user has to do it manually.
OnAutoInsertRestart()	
OnCopyFullRisk	PROCEED AT YOUR OWN RISK message.
Event OnCopyFullRisk(Response As NERO_RESPONSE)	
OnCopyQualityLoss	Tell the user that there will be quality loss during the copy and ask if he wants to continue anyway.
Event OnCopyQualityLoss(Response As NERO_RESPONSE)	

Event	Description
OnDisconnect	"Disconnect is turned off in the system configuration. This may cause serious problems while burning: your CD might be damaged, or the system might hang up." Return NERO_RETURN_ON_RESTART to turn on disconnect and restart windows. Return NERO_RETURN_RESTART to not change disconnect option and restart windows. Return NERO_RETURN_CONTINUE to continue at own risk.
Event OnDisconnect(Response As NERO_RESPONSE)	
OnDisconnectRestart	Same as OnDisconnect, but restarting has been selected already and must not be canceled, so valid return codes are only NERO_RETURN_ON_RESTART and NERO_RETURN_RESTART.
Event OnDisconnectRestart(Response As NERO_RESPONSE)	
OnDriveStatusChanged	Drive status has changed.
Event OnDriveStatusChanged(hostID As Long, targetID As Long, driveStatus As NERO_DRIVECHANGE_RESULT)	
OnFileSelImage	Ask the user for the path of the file which will be generated by the Image Recorder. The image path may be 256 bytes long.
Event OnFileSelImage(Filename As String)	
OnMegaFatal	Mega fatal internal problem that cannot be fixed!
OnMegaFatal()	
OnNonEmptyCDRW	Tell the user that the CDRW is not empty. Will be called only if the NERO_BURN_FLAG_DETECT_NON_ EMPTY_CDRW flag is given to the burn function. Returning NERO_RETURN_EXIT will stop the burn process. Returning NERO_RETURN_CONTINUE will continue the burn process. Returning NERO_RETURN_RESTART will ask the user for another CD.
Event OnNonEmptyCDRW(Response As NERO_RESPONSE)	
OnNoTrackFound	Problem because we do not have any track.
OnNoTrackFound()	

Event	Description
OnOverburn	"Sorry, this compilation contains too much data to fit on the CD with respect to the normal CD capacity. Do you want to try overburn writing at your own risk (this might cause read errors at the end of the CD or might even damage your recorder)? Note: It is also possible, that SCSI/Atapi errors occur at the end of the simulation or burning. Even in this case there is a certain chance, that the CD is readable."
Event OnOverburn(Response As NERO_RESPONSE)	
OnRestart	The system is being restarted.
OnRestart()	
OnRoboMoveCD	The disc is being moved from the source node to the destination node.
Event OnRoboMoveCD(id As Long, rmnSource As ROBOMOVENODE, rmnDestination As ROBOMOVENODE)	
OnRoboMoveCDDone	The disc moving process is completed.
Event OnRoboMoveCDDone(id As Long)	
OnRoboMoveUserMessage	A message from the robot driver.
Event OnRoboMoveUserMessage(messageType As ROBOUSERMESSAGETYPE, bstrMessage As String, Response As NERO_RESPONSE)	
OnSettingsRestart	"Nero detected some modifications of your PC system configuration and needs to modify some settings. Please restart your PC to make the changes become effective." Returning NERO_RETURN_RESTART will restart the system. Returning NERO_RETURN_CONTINUE will continue the operation without restarting.
Event OnSettingsRestart(Response As NERO_RESPONSE)	
OnTempSpace	This event allows the application to set a directory that is used for storing temporary files.
Event OnTempSpace(bstrCurrentDir As String, i64FreeSpace, i64SpaceNeeded, bstrNewTempDir As String)	
OnWaitCD	Tell the user that the system is waiting for the insertion of a CD.
Event OnWaitCD(WaitCD As NERO_WAITCD_TYPE, WaitCDLocalizedText As String)	
OnWaitCDDone	Close the message box again, we are done.
OnWaitCDDone()	
OnWaitCDMediaInfo	Provide information about which media is expected and which media is currently present in the recorder.
Event OnWaitCDMediaInfo(LastDetectedMedia As NERO_MEDIA_TYPE, LastDetectedMediaName As String, RequestedMedia As NERO_MEDIA_TYPE, RequestedMediaName As String)	

Event	Description
OnWaitCDReminder	It is time to remind the user of inserting the CD: play a jingle, flash the screen, etc. Called only once after a certain amount of time of no CD being inserted.
Event OnWaitCDReminder()	

7.2.3. NeroAudioTrack

This object is creatable by the client. It is used when the client wants to burn an audio track.

When **TrackType** is set to **NERO_AUDIO_TRACK_THROUGH_EVENT**, NeroAudioTrack fires **OnReadAudioBurn** that asks for data from the client. The client should return the appropriate track data that the burn engine will write to the disc. **OnEOF** and **OnError** ask the client to provide EOF and error information to the burn engine.

All events mentioned here are handled by *NeroCOM* internally in a way so that even if the client does not handle the events, some defaults are passed to the *NeroAPI*. But, it must be stressed again that there is not much point for the client to create an audio or a freestyle track or a **NeroFileProducer** object and not handle the crucial events that provide data to the burn engine. If the client wanted an audio track, it is most sensible to provide the data that it wants written to the disc. If the data is not provided through the event, what good is an audio track?

NERO_AUDIO_TRACK_TYPE and **NERO_FREESTYLE_TRACK_TYPE** enums determine the way in which the burn engine will collect the data. These can be set to audio objects by the **TrackType** property. Events will be fired only if **NERO_AUDIO_TRACK_THROUGH_EVENT** is set for an audio track. Other enum constants are available for other data supply methods.

Properties

Property	read-only	Description
Artist	no	Name of the artist.
Property Artist As String		
Filename	no	The name of the file to write.
Property Filename As String		
Index0ContainsData	no	True, if audio data shall be written into index 0. Data for index 0 must be provided. This can be used to prevent silent pauses between tracks.
Property Index0ContainsData As Boolean		
LengthInBlocks	no	The length in blocks.
Property LengthInBlocks As Long		
PauseInBlksBeforeThisTrack	no	The pause in blocks before this track.
Property PauseInBlksBeforeThisTrack As Long		

Property	read-only	Description
RelativeIndexBlkPositions	yes	Offsets between one index position and the next one.
Property RelativeIndexBlkPositions As NeroRelativeIndexBlkPositions		
Title	no	Name of the title.
Property Title As String		
TrackType	no	MP3, Wave, WMA or data passed by events.
Property TrackType As NERO_AUDIO_TRACK_TYPE		

Events

Event	Description
one of	Set EOF to True if the end of the file has been encountered.
Event OnEOF(EOF As Boolean)	
OnError	Set Error to True if an Error has occurred.
Event OnError(Error As Boolean)	
OnReadAudioBurn	Provide Audio Data for the burn engine.
Event OnReadAudioBurn(Length As Long, Data)	

7.2.4. NeroBurnAtOnceInfo

This object contains offset information which is relevant for Burn-at-once functionality. This object is available as property of NeroDataInputStream.

Since this is the first object in this documentation that is related to Burn-at-once, we have gathered the introductory material regarding the Burn-at-once features here. Also, please be advised to study the corresponding chapter of the *NeroAPI* documentation which offers a broader coverage of the subject.

In general, the Burn-at-once interface allows burning a DVD-Video without storing intermediate files on your hard disk.

There are two approaches to using Burn-at-once. One approach requires fewer actions on the client side while the other one offers more control to the client.

Single method approach

The easier approach uses the BurnAtOnce method of the INeroDrive4 interface. All parameters are prepared beforehand and then passed to the function. Once done, it fires OnDoneBurn just as any other burning function.

All IFO and BUP files must have the exact file size set.

All other video files must have the estimated size set. The size will be updated during burning. Note, that the estimated size should be as close to the actual size as possible.

For each video title set only add the first VOB file (e.g. VTS_01_1.VOB) to the file system container and pass all the video data of this title set in the file producer of this file.
The *NeroAPI* will automatically split the file at the appropriate position (1 GB - logical block size). DVD-Video files will be sorted automatically.

It is guaranteed that the file data will be requested in the following order:

- VMGM_VOB (VIDEO_TS.VOB) [if present]
- For each video title set (VTS):
  - VTSM_VOB (e.g. VTS_01_0.VOB) [if present]
  - VTSTT_VOBS (e.g. VTS_01_1.VOB) [mandatory]
  - VTSI file (e.g. VTS_01_0.IFO) [mandatory]
  - VTSI backup (e.g. VTS_01_0.BUP) [mandatory]
- VMGMI (VIDEO_TS.IFO) [mandatory]
  Backup for VMGMI (VIDEO_TS.BUP) [mandatory]

More complex approach

If using the harder approach, the other INeroDrive4 methods should be used. There is a strict sequence of method invocation. First there should be a BAOCreateHandle call that will initiate the Burn-at-once process. This method can take longer to finish. That is why it has an accompanying OnDoneBAOCreateHandle. No actual handle is returned to the client. It is stored internally.

Then, for each file burned there should be a BAOOpenFile, BAOWriteToFile and BAOCloseFile. Only BAOWriteToFile will fire an OnDoneBAOWriteToFile event, while the open/close functions will complete their task and return without the need to fire any events.

Finally, when the burning work is done, a BAOCloseHandle should be called with one of the NERO_BAO_FLAGS values indicating the status. This method also takes longer to finish and therefore an event OnDoneBAOCloseHandle is fired when the work is done.

When using *NeroCOM,* events are fired to notify the client of task completion while the *NeroAPI* blocks until the method returns.

If using NeroFileProducer in conjunction with NeroFileSystemDescContainer and friends (producing file data on the fly instead of using existing disk files), OnProduceFile event receives an INeroDataInputStream interface pointer. The object that implements INeroDataInputStream now also implements INeroDataInputStream2 (this should be transparent for VB clients and a simple QueryInterface away for C++ clients). The INeroDataInputStream2 provides a method to obtain an INeroBurnAtOnceInfo interface pointer.

Of course, the INeroBurnAtOnceInfo interface pointer will be returned only if in the middle of burn-at-once burning process. If burning something else, the value returned will be NULL as it is senseless to get a burn-at-once info during a non-burn-at-once burning process.

Properties

Property	read-only	Description
Offset	yes	The offset to the start of the VIDEO_TS.IFO file in blocks.
Property Offset As Long		

7.2.5. NeroAudioTracks

A collection of NeroAudioTrack objects.

Properties

Property	read-only	Description
Count	yes	Number of tracks in the collection.
Property Count As Long		

Methods

Method	Description
Add	Add a track.
Sub Add(pNewVal As NeroAudioTrack)	
Item	Retrieve a track by index.
Function Item(Index As Long) As NeroAudioTrack	
Remove	Remove a track from the track list.
Sub Remove(Index As Long)	

7.2.6. NeroCDInfo

This object will be provided by the OnDoneCDInfo event of the NeroDrive object, when the user has called the CDInfo method.

Properties

Property	read-only	Description
Artist	yes	Name of the artist.
Property Artist As String		
AvailableEraseModes	yes	Available erase modes: Entire or quick.
Property AvailableEraseModes As Long		
FreeCapacityInBlocks	yes	Number of unused blocks on the media.
Property FreeCapacityInBlocks As Long		
IsWriteable	yes	The media might be non-writeable.
Property IsWriteable As Boolean		
MediaType	yes	The type of the media.
Property MediaType As NERO_MEDIA_TYPE		
Title	yes	The title of the media.
Property Title As String		
Tracks	yes	This property gives access to the NeroTracks collection.
Property Tracks As NeroTracks		
UnusedBlocks	yes	Difference between Lead-Out position and last possible Lead-Out position.
Property UnusedBlocks As Long		

7.2.7. NeroCDStamp

This object is used when a particular CD is requested for burning.

7.2.8. NeroDataInputStream

This object is created in conjunction with the Burn-at-once process.

Properties

Property	read-only	Description
BurnAtOnceInfo	yes	This property provides access to the offset from the start of VIDEO_TS.IFO.
Property BurnAtOnceInfo As NeroBurnAtOnceInfo		

Methods

Method	Description
Write	Write buffer of size lSize.
Sub Write(varBuffer, lSize As Long)	

7.2.9. NeroDrive

The methods **EjectCD**, **LoadCD**, **EraseCDRW**, **DAE**, **CDInfo**, **ImportISOTrack**, **WaitForMedia** and all **BurnXXX** methods have their corresponding events. These **should** be handled but world will not come to an end if they are not. But it does not make much sense to not handle them anyway. For instance, **OnWriteDAE** must be handled or DAE will fail because this event is a provider of audio data and if nobody wants it, there is not much point in continuing the operation, is there? All those "long" methods return instantly and start the actual work in a separate thread. That is why the client program can continue uninterrupted. To signal the end of the operation all "long" methods have their corresponding **OnDoneXXX** events, such as **OnDoneBurn** or **OnDoneErase**.

In "long" burning operations, such as **BurnIsoAudioCD**, **NeroISOTrack** and **NeroAudioTracks** objects are **LOCKED**! Until the "long" operation finishes, any access to these objects (and the objects contained) will result in an error. This is because it would be harmful to allow the objects to be accessed, especially modified while the "long" operation uses the data contained in those objects. Basically, you should **refrain** from using the objects that are passed as parameters to "long" methods as they will be locked until the end of operation. Any access will result in a run-time error. Once **OnDoneXXX** event is fired, objects can be considered unlocked and can be used as normal. The locking feature is present primarily for the sake of data protection.

Do **not** use any objects until the appropriate event is fired!

Properties

Property	read-only	Description
AvailableSpeeds	yes	Retrieve the available speeds for a certain access type and media type (as transfer rates).
Property AvailableSpeeds(accessType As NERO_ACCESSTYPE, nmt As NERO_MEDIA_TYPE) As NeroSpeeds		
BufUnderrunProtName	yes	Retrieve the name of the buffer underrun protection technology.
Property BufUnderrunProtName As String		
Capabilities	yes	Query the capabilities of a drive.
Property Capabilities As NERO_CAPABILITIES		
CDRWErasingTime	yes	Get an estimate about the time it requires to erase a CDRW.
Property CDRWErasingTime(Quick As Boolean) As Long		
DeviceID	yes	The drive's device ID.
Property DeviceID As Long		
DeviceName	yes	The drive's device name.
Property DeviceName As String		

Property	read-only	Description
DeviceReady	yes	Indicates whether the drive is ready or not.
Property DeviceReady As Boolean		
DeviceReady2	yes	Indicates whether the drive is ready or not.
Property DeviceReady2 As NERO_DRIVE_ERROR		
DevType	yes	Information about the characteristics of the drive.
Property DevType As NERO_SCSI_DEVTYPE		
DriveBufferSize	yes	Drive buffer size (internal) in KB.
Property DriveBufferSize As Long		
DriveLetter	yes	Windows drive letter.
Property DriveLetter As String		
HostAdapterName	yes	The name of the host adapter the drive is attached to.
Property HostAdapterName As String		
HostAdapterNo	yes	The number of the host adapter.
Property HostAdapterNo As Long		
MandatoryBUPSpeed	yes	It is highly recommended to enable buffer underrun protection when burning at this speed or faster. Contains 0 if there is no recommendation.
Property MandatoryBUPSpeed As Long		
MediaReadSupport	yes	Bit field of supported media for reading.
Property MediaReadSupport As NERO_MEDIA_TYPE		
MediaSupport	yes	Bit field of supported media (constructed with the NERO_MEDIA_TYPE enum).
Property MediaSupport As Long		
ReadSpeeds	yes	Available speeds for read access. Unlike the NeroSpeeds of the AvailableSpeeds property, the write speeds are not based on transfer rates but speed ratings. For CDs, a speed rating of 1x stands for 150 KB/sec. For DVDs, a rating of 1x translates to 1.385 MB/sec.
Property ReadSpeeds As NeroSpeeds		
SupportedAccessModes	yes	Supported access modes.
Property SupportedAccessModes As Long		
WriteSpeeds	yes	Available speeds for write access. Unlike the NeroSpeeds of the AvailableSpeeds property, the write speeds are not based on transfer rates but speed ratings. For CDs, a speed rating of 1x stands for 150 KB/sec. For DVDs, a rating of 1x translates to 1.385 MB/sec.
Property WriteSpeeds As NeroSpeeds		

Methods

Method	Description
AssociateRobo	Associate a robot of the specified type connected to a given port with a port number.
Function AssociateRobo(bstrRoboIdentifier As String, port As NERO_COMMNODE_TYPE, lPortNum As Long) As NEROAPI_ROBO_ERROR	
BAOCloseFile	This is a Burn-at-once method that needs to be called after all data for a file has been written.
Function BAOCloseFile() As NERO_BURN_ERROR	
BAOCloseHandle	This Burn-at-once method is called after all video data has been written. Now additional files that are present in the file system container will be written and then the disc will be finalized. The flags can be used to terminate the Burn-at-once process, either because the user canceled or a problem occurred. The burn process will then be stopped immediately.
Sub BAOCloseHandle(Flags As NERO_BAO_FLAGS)	
BAOCreateHandle	This method is used to obtain a Burn-at-once handle.
Sub BAOCreateHandle(pContent As NeroFileSystemDescContainer, bstrTempDirectory As String, Flags As NERO_BURN_FLAGS, Speed As Long)	
BAOOpenFile	When performing Burn-at-once processing, for each video file in the VIDEO_TS folder, this function needs to be called. The plOffset parameter contains the offset of the file from the beginning of VIDEO_TS.IFO in blocks.
Function BAOOpenFile(bstrName As String, plOffset As Long) As NERO_BURN_ERROR	
BAOWriteToFile	This function is used to write all the data during Burn-at-once processing. The number of bytes to write must be a multiple of 2048!
Sub BAOWriteToFile(varBuffer, lNumberOfBytesToWrite As Long)	
BurnAtOnce	This function will start the Burn-at-once process. Please refer to the NeroBurnAtOnceInfo chapter for more details on the mechanism.
Sub BurnAtOnce(pContent As NeroFileSystemDescContainer, bstrTempDirectory As String, Flags As NERO_BURN_FLAGS, Speed As Long)	
BurnDiscCopy	Burn the copy of a disc.
Sub BurnDiscCopy(pDriveSource As INeroDrive, bOnTheFly As Boolean, bstrImageFilePath As String, bDeleteImage As Boolean, lReadSpeed As Long, lNumberOfRetries As Long, bIgnoreDataReadErrors As Boolean, bIgnoreAudioReadErrors As Boolean, bReadIsrcAndMediaCatalogNo As Boolean, bRawReadMode As Boolean, Flags As NERO_BURN_FLAGS, Speed As Long, Type As NERO_MEDIA_TYPE)	

Method	Description
BurnFileSystemContent	Burn from a NeroFileSystemDescContainer object.
Sub BurnFileSystemContent(pContent As NeroFileSystemDescContainer, Flags As NERO_BURN_FLAGS, Speed As Long, type As NERO_MEDIA_TYPE)	
BurnFreestyleCD	Burn a Freestyle CD.
Sub BurnFreestyleCD(pContent As NeroFileSystemDescContainer, Artist As String, Title As String, CDExtra As Boolean, ISOTrack As NeroISOTrack, FreestyleTracks As NeroFreestyleTracks, CDStamp As NeroCDStamp, Flags As NERO_BURN_FLAGS, Speed As Long, type As NERO_MEDIA_TYPE)	
BurnImage	Burn from an image file.
Sub BurnImage(ImageFilename As String, Flags As NERO_BURN_FLAGS, Speed As Long)	
BurnImage2	Burn from an image file, also providing a media type
Sub BurnImage2(ImageFilename As String, Flags As NERO_BURN_FLAGS, Speed As Long, MediaType As NERO_MEDIA_TYPE)	
BurnIsoAudioCD	Burn an ISO/Audio CD.
Sub BurnIsoAudioCD(Artist As String, Title As String, CDExtra As Boolean, ISOTrack As NeroISOTrack, AudioTracks As NeroAudioTracks, CDStamp As NeroCDStamp, Flags As NERO_BURN_FLAGS, Speed As Long, type As NERO_MEDIA_TYPE)	
BurnNVAPI	Burn a *Nero Vision API* project.
Sub BurnNVAPI(pBurnContext As INeroBurnContext, Flags As NERO_BURN_FLAGS, Speed As Long)	
BurnVideoCD	Burn a VCD or SVCD.
Sub BurnVideoCD(VideoItems As NeroVideoItems, SuperVideoCD As Boolean, TempPath As String, ISOTrack As NeroISOTrack, Flags As NERO_BURN_FLAGS, Speed As Long)	
BurnVideoCD2	Burn a VCD or SVCD, providing the desired resolution.
Sub BurnVideoCD2(VideoItems As NeroVideoItems, SuperVideoCD As Boolean, TempPath As String, ISOTrack As INeroISOTrack, Flags As NERO_BURN_FLAGS, Speed As Long, resolution As NERO_VIDEO_RESOLUTION)	
CDInfo	Gather information about a media.
Sub CDInfo(Flags As NERO_CDINFO_FLAGS)	
CreateBlockReader	Use this function to obtain a block reader.
Function CreateBlockReader(AccessMode As AccessMode) As INeroFileSystemBlockReader	
CreateBlockWriter	Use this function to obtain a block writer.
Function CreateBlockWriter(AccessMode As AccessMode) As INeroFileSystemBlockWriter	
DAE	Start Digital Audio Extraction.
Sub DAE(TrackStartBlock As Long, TrackLengthInBlocks As Long, Filename As String, Speed As Long)	
EjectCD	Eject a CD.
Sub EjectCD()	
EnableStatusCallback	Enable the status callback.
Sub EnableStatusCallback(driveStatus As NERO_DRIVESTATUS_TYPE, bEnable As Boolean)	
EraseCDRW	Erase a CDRW.
Function EraseCDRW(Quick As Boolean) As Long	
EraseDisc	Erase a media.
Function EraseDisc(Quick As Boolean, eraseMode As NERO_ERASE_MODE) As Long	

Method	Description
EstimateTrackSize	Estimate the resulting track size of an ISO track.
Sub EstimateTrackSize(ISOTrack As INeroISOTrack, Flags As NERO_ESTIMATE_FLAGS, pOptions As NeroFileSystemTrackOptions)	
GetDeviceOption	Get information about a special low level option from a device, e.g. if a device is capable of changing the booktype of a DVD.
Function GetDeviceOption(deviceOption As NERO_DEVICE_OPTION)	
GetVMSInfo	Retrieve Virtual Multi Session information.
Function GetVMSInfo() As NeroVMSInfo	
ImportIsoTrack	Import an ISO track from a previous session.
Sub ImportIsoTrack(TrackNumber As Long, dwFlags As NERO_IMPORT_ISO_TRACK_FLAGS)	
InitImageRecorder	Set the image file for the image recorder. This can be used to initialize the image recorder for packet writing.
Sub InitImageRecorder(imageFilePath As String, MediaType As NERO_MEDIA_TYPE)	
LoadCD	Load a CD.
Sub LoadCD()	
SetDeviceOption	Set a special option for a device. For example, the booktype can be changed to DVDROM - if the device allows it
Sub SetDeviceOption(deviceOption As NERO_DEVICE_OPTION, varOption)	
SetRoboFlag	This function sets runtime options for the robot. Those will not be remembered beyond the current session.
Function SetRoboFlag(roboFlag As NERO_ROBO_FLAG, lRoboValue As Long) As NEROAPI_ROBO_ERROR	
UpdateDeviceInfo	Force an update of the device information.
Function UpdateDeviceInfo(MediaType As NERO_MEDIA_TYPE) As NERO_DRIVE_ERROR	
WaitForDisc	Request a certain disc, providing additional wait flags.
Sub WaitForDisc(MediaType As NERO_MEDIA_TYPE, BurnFlags As NERO_BURN_FLAGS, pCDStamp As NeroCDStamp, Flags As NERO_WAIT_FOR_DISC_FLAGS)	
WaitForMedia	Request a certain media.
Sub WaitForMedia(MediaType As NERO_MEDIA_TYPE, BurnFlags As NERO_BURN_FLAGS, pCDStamp As NeroCDStamp)	

Events

Event	Description
OnAborted	Checks whether the operation has been aborted.
Event OnAborted(Abort As Boolean)	
OnAddLogLine	Provides information that might be relevant to the user.
Event OnAddLogLine(TextType As NERO_TEXT_TYPE, Text As String)	
OnDisableAbort	This event will only be fired if NERO_BURN_FLAG_DISABLE_ABORT has been provided as burn option.
Event OnDisableAbort(EnableAbort As Boolean)	
OnDoneBAOCloseHandle	Burn-at-once handle closed.
Event OnDoneBAOCloseHandle(StatusCode As NERO_BURN_ERROR)	
OnDoneBAOCreateHandle	Burn-at-once handle created.
Event OnDoneBAOCreateHandle(StatusCode As NERO_BURN_ERROR)	
OnDoneBAOWriteToFile	Data was written.
Event OnDoneBAOWriteToFile(StatusCode As NERO_BURN_ERROR, lNumberOfBytesWritten As Long)	
OnDoneBurn	Burning has been completed – whether it was successful can be determined by value of StatusCode.
Event OnDoneBurn(StatusCode As NERO_BURN_ERROR)	
OnDoneCDInfo	CD info gathering completed.
Event OnDoneCDInfo(pCDInfo As NeroCDInfo)	
OnDoneDAE	Digital Audio Extraction completed.
Event OnDoneDAE(Ok As Boolean)	
OnDoneErase	Erasing CDRW completed.
Event OnDoneErase(Ok As Boolean)	
OnDoneEstimateTrackSize	Track size estimation completed.
Event OnDoneErase(Ok As Boolean)	
OnDoneImport	ISO track import completed.
Event OnDoneImport(Ok As Boolean, Folder As NeroFolder, CDStamp As NeroCDStamp)	
OnDoneImport2	
Event OnDoneImport2(bOk As Boolean, pFolder As NeroFolder, pCDStamp As NeroCDStamp, pImportInfo As NeroImportDataTrackInfo, importResult As NERO_IMPORT_DATA_TRACK_RESULT)	
OnDoneWaitForMedia	The media that the N*eroAPI* had waited for has been inserted.
Event OnDoneWaitForMedia(Success As Boolean)	
OnDriveStatusChanged	´The drive status has changed.
Event OnDriveStatusChanged(driveStatus As NERO_DRIVESTATUS_RESULT)	
OnMajorPhase	A major phase change has occurred.
Event OnMajorPhase(phase As NERO_MAJOR_PHASE)	
OnProgress	The progress of an operation has changed.
Event OnProgress(ProgressInPercent As Long, Abort As Boolean)	

Event	Description
OnRoboPrintLabel	This event is used for label printing. It will be called by *NeroCOM* when it is time to print the label during robot operation. It must return TRUE if the label was printed successfully, FALSE otherwise.
Event OnRoboPrintLabel(pbSuccess As Boolean)	
OnSetPhase	The current process has entered a new phase.
Event OnSetPhase(Text As String)	
OnSubTaskProgress	Provides the write buffer fill level.
Event OnSubTaskProgress(ProgressInPercent As Long, Abort As Boolean)	
OnWriteDAE	
Event OnWriteDAE(TrackNumber As Long, Data)	

7.2.10. NeroDrives

A read-only collection of NeroDrive objects.

Properties

Property	read-only	Description
Count	yes	Number of drives in the collection.
Property Count As Long		

Methods

Method	Description
Item	Access a drive by its index number.
Function Item(Index As Long) As NeroDrive - **default**	

7.2.11. NeroFreestyleTrack

This object is creatable by the client when the client wants to burn a freestyle track. Similarly to the **NeroFileProducer**'s **OnProduceFile** event, this object fires an **OnReadFreestyleBurn** event that ask for data from the client. The client should return the appropriate track data that the burn engine will write to the disc. **OnEOF** and **OnError** ask the client to provide EOF and error information to the burn engine.

Properties

Property	read-only	Description
Artist	no	The name of the artist.
Property Artist As String		
Filename	no	The file name.
Property Filename As String		

Property	read-only	Description
LengthInBlocks	no	The length of the track in blocks.
Property LengthInBlocks As Long		
PauseInBlksBeforeThisTrack	no	Pause to insert before the current track in blocks.
Property PauseInBlksBeforeThisTrack As Long		
RelativeIndexBlkPositions	yes	Offsets between one index position and the next one.
Property RelativeIndexBlkPositions As NeroRelativeIndexBlkPositions		
SourceType	no	Data provided by a file or through events.
Property SourceType As NERO_FREESTYLE_SOURCE_TYPE		
Title	no	The title of the track.
Property Title As String		
TrackType	no	Audio Mode 1 Mode 2 form 1
Property TrackType As NERO_FREESTYLE_TRACK_TYPE		

Events

Event	Description
OnEOF	Fired to check whether the end of the file has been read.
Event OnEOF(EOF As Boolean)	
OnError	Fired to check whether an error has occurred.
Event OnError(Error As Boolean)	
OnReadFreestyleBurn	Fired when *NeroCOM* requires data for burning.
Event OnReadFreestyleBurn(Length As Long, Data)	

7.2.12. NeroFreestyleTracks

A collection of NeroFreestyleTrack objects.

Properties

Property	read-only	Description
Count	yes	A collection of NeroFreestyleTrack items.
Property Count As Long		

Methods

Method	Description
Add	Add a new track.
Sub Add(pNewVal As NeroFreestyleTrack)	

Method	Description
Item	Retrieve a track by its index number.
Function Item(Index As Long) As NeroFreestyleTrack – **default**	
Remove	Remove a track.
Sub Remove(Index As Long)	

7.2.13. NeroImportDataTrackInfo

This object is used by the OnDoneImport2 event of the NeroDrive object.

Properties

Property	read-only	Description
VolumeName	yes	Volume name.
Property VolumeName As String		

7.2.14. NeroISOTrack

This object is passed to the BurnIsoAudioTrack and BurnFreestyleTrack methods of the NeroDrive object.

Properties

Property	read-only	Description
Abstract	no	Abstract file.
Property Abstract As String		
Application	no	The application, which created this track.
Property Application As String		
Bibliographic	no	Bibliographic file.
Property Bibliographic As String		
BurnOptions	no	Combination of burn flags to use Joliet, Mode 2 etc.
Property BurnOptions As NERO_BURN_OPTIONS		
Copyright	no	Copyright file.
Property Copyright As String		
DataPreparer	no	The preparer of this track.
Property DataPreparer As String		
Name	no	The name of the track.
Property Name As String		
Publisher	no	The publisher of this track.
Property Publisher As String		
RootFolder	no	The root folder as NeroFolder object.
Property RootFolder As NeroFolder		
RootFolderWrapper	no	
Property RootFolderWrapper As NeroFolder		
SystemIdentifier	no	System identifier.
Property SystemIdentifier As String		

Property	read-only	Description
VolumeSet	no	This name is used when multiple media are part of one logical unit.
Property VolumeSet As String		

7.2.15. NeroRelativeIndexBlkPositions

Offsets between one index position and the next one, used by various track objects.

Properties

Property	read-only	Description
Count	yes	Number of position entries in the collection.
Property Count As Long		

Methods

Method	Description
Add	Add a position.
Sub Add(NewVal As Long)	
Item	Retrieve a position.
Function Item(Index As Long) As Long – **default**	
Remove	Remove a position
Sub Remove(Index As Long)	

7.2.16. NeroSpeeds

This object is used in the various speed properties of the NeroDrive object.

Properties

Property	read-only	Description
BaseSpeedKBs	yes	Speed corresponding to 1X for the selected media in KB/s.
Property BaseSpeedKBs As Long		
Count	yes	Number of supported speeds.
Property Count As Long		

Methods

Method	Description
Item	Retrieve a NeroSpeeds entry by its index number.
Function Item(Index As Long) As Long	

7.2.17. NeroTrack

An object to describe a single track.

Properties

Property	read-only	Description
Artist	yes	The artist's name for Audio Tracks.
Property Artist As String		
BlockSize	yes	Block size.
Property BlockSize As Long		
ISRC	yes	International Standard Recording Code.
Property ISRC As String		
SessionNumber	yes	Number of the recorded session.
Property SessionNumber As Long		
Title	yes	Title of the track for Audio Tracks.
Property Title As String		
TrackLengthInBlks	yes	The length of the track in blocks.
Property TrackLengthInBlks As Long		
TrackNumber	yes	The number of the track.
Property TrackNumber As Long		
TrackStartBlk	yes	Start Block of Track.
Property TrackStartBlk As Long		
TrackType	yes	Can be MP3, Wave, Windows Media or track data provided through events.
Property TrackType As NERO_TRACK_TYPE		

7.2.18. NeroTracks

A collection of track.

Properties

Property	read-only	Description
Count	yes	The number of tracks in the collection.
Property Count As Long		

Methods

Method	Description
Item	Retrieve a track by its index number.
Function Item(Index As Long) As NeroTrack - **default**	

7.2.19. NeroVideoItem

A single video item.

Properties

Property	read-only	Description
ItemType		JPEG, MPEG or non-encoded video data.
Property ItemType As NERO_VIDEO_ITEM_TYPE		
PauseAfterItem		Length of the pauses after this item.
Property PauseAfterItem As Long		
SourceFileName		MPG or JPG or AVI file name.
Property SourceFileName As String		

7.2.20. NeroVideoItems

A collection of NeroVideoItem objects.

Properties

Property	read-only	Description
Count	yes	Collection of NeroVideoItem objects.
Property Count As Long		

Methods

Method	Description
Add	Add an item.
Sub Add(pNewVal As NeroVideoItem)	
Item	Retrieve and item by its index number.
Function Item(Index As Long) As NeroVideoItem – **default**	
Remove	Remove an item.
Sub Remove(Index As Long)	

7.2.21. NeroVMSInfo

Virtual Multi Session information.

Properties

Property	read-only	Description
Count	yes	Number of VMS sessions.
Property Count As Long		
NextWritableAddress	yes	
Property NextWritableAddress As Long		

Methods

Method	Description
Item	Retrieves a single session by index number.
Function Item(Index As Long) As NeroVMSSession	

7.2.22. NeroVMSSession

A single VMS session.

Properties

Property	read-only	Description
CreationTime	yes	
Property CreationTime As Date		
NextWritableAddress	yes	
Property NextWritableAddress As Long		
SessionName	yes	
Property SessionName As String		

7.3. Packet Writing Objects

7.3.1. INeroFileSystemBlockReader

This is an interface for reading from block devices. It will provide necessary data about the underlying medium as well as cache data if necessary.

Properties

Property	read-only	Description
PartitionCount	yes	Number of partitions.
Property PartitionCount(piCount As Long)		
PartitionInfo	yes	Partition information.
Property PartitionInfo(iIndex As Long) As NeroFSPartitionInfo		
PartitionInfoForSector	yes	Returns the partition info for the partition a given sector resides in.
Property PartitionInfoForSector(pSecNo As NeroFSSecNo) As NeroFSPartitionInfo		

Methods

Method	Description
ReadSectorsBuffered	The buffered reading method will use a cache to optimize file system access. It should be used when reading directory structures. This method returns error codes as described in NeroFSError. Your read requests may not cross partition boundaries!
Function ReadSectorsBuffered(pvarBuffer, pStartSector As NeroFSSecNo, pNoSectors As NeroFSSecNo, ppNoSectorsRead As NeroFSSecNo) As NeroFSError	
ReadSectorsUnBuffered	The unbuffered reading method should be used when reading file contents. This method returns error codes as described in NeroFSError. Your read requests may not cross partition boundaries!
Function ReadSectorsUnBuffered(pvarBuffer, pStartSector As NeroFSSecNo, pNoSectors As NeroFSSecNo, pNoSectorsRead As NeroFSSecNo) As NeroFSError	

7.3.2. INeroFileSystemBlockWriter

As is the case with the reader interface, the writer interface also provides two methods for sector access. While WriteSectorsUnBuffered will merely ensure the consistency of the read cache (write thru), WriteSectorsBuffered will not write anything to the block device immediately but will cache a certain amount of sectors before doing so.

The latter increases performance considerably but is prone to data loss in an unstable environment.

Please note that regardless of which method you use, you **must** call FlushSectorCache if you want all your data to be at their final physical location. The reason is that even when writing in UnBuffered mode, the driver may decide to not write away your data immediately. This depends on the underlying writing scheme (e.g. packet writing will always try to collect a certain amount of sectors).

Properties

Property	read-only	Description
PartitionCount	yes	Number of partitions.
Property PartitionCount(piCount As Long)		
PartitionInfo	yes	Partition information.
Property PartitionInfo(iIndex As Long) As NeroFSPartitionInfo		
PartitionInfoForSector	yes	Returns the partition info for the partition a given sector resides in.
Property PartitionInfoForSector(pSecNo As NeroFSSecNo) As NeroFSPartitionInfo		

Methods

Method	Description
FlushSectorCache	Force the flushing of the sector cache. FlushSectorCache will be performed implicitly upon deleting the block writer object.
Sub FlushSectorCache()	
ReadSectorsBuffered	The buffered reading method will use a cache to optimize file system access. It should be used when reading directory structures. This method returns error codes as described in NeroFSError. Your read requests may not cross partition boundaries!
Function ReadSectorsBuffered(pvarBuffer, pStartSector As NeroFSSecNo, pNoSectors As NeroFSSecNo, ppNoSectorsRead As NeroFSSecNo) As NeroFSError	

Method	Description
ReadSectorsUnBuffered	The unbuffered reading method should be used when reading file contents. This method returns error codes as described in NeroFSError. Your read requests may not cross partition boundaries!
Function ReadSectorsUnBuffered(pvarBuffer, pStartSector As NeroFSSecNo, pNoSectors As NeroFSSecNo, pNoSectorsRead As NeroFSSecNo) As NeroFSError	
WriteSectorsBuffered	Method for buffered writing.
Function WriteSectorsBuffered(pvarBuffer, pStartSector As NeroFSSecNo, pNoSectors As NeroFSSecNo, ppNoSectorsWritten As NeroFSSecNo) As NeroFSError	
WriteSectorsUnBuffered	Method for unbuffered writing.
Function WriteSectorsUnBuffered(pvarBuffer, pStartSector As NeroFSSecNo, pNoSectors As NeroFSSecNo, ppNoSectorsWritten As NeroFSSecNo) As NeroFSError	

7.3.3. NeroFSPartitionInfo

This object is used by the INeroFileSystemBlockReader and INeroFileSystemBlockWriter interfaces.

Properties

Property	read-only	Description
PartitionNumber	yes	The current partition number.
Property PartitionNumber(piNumber As Long)		
PartitionSize	yes	The number of sectors this Partition contains.
Property PartitionSize As NeroFSSecNo		
PartitionStart	yes	The start sector for this Partition.
Property PartitionStart As NeroFSSecNo		
PartitionType	yes	The type of Partition.
Property PartitionType As NeroFSTrackType		
SectorSize	yes	Sector size for this Partition.
Property SectorSize As Long		

7.3.4. NeroFSSecNo

Sector number type.

Properties

Property	read-only	Description
HiPart	no	High part.
Property HiPart As Long		
LoPart	no	Low Part.
Property LoPart As Long		

7.4. File System Objects

7.4.1. NeroDataInputStream

Used by the file producer object.

Methods

Method	Description
Write	This method allows the file producer to return the data.
Sub Write(varBuffer, lSize As Long)	

7.4.2. NeroDirectoryContainer

This object represents the content of a directory.

Properties

Property	read-only	Description
Count	yes	The number of entries in the directory.
Property Count As Long		

Methods

Method	Description
AddDirectory	This function will return a directory object when provided a name and a priority. Priority specifies the position in the directory. Priority specifies some user-defined ordinal defining the order in which the files are being written to the disc physically (like .ifo comes before .vob). Priorities are valid across directories. The file entry order in a directory is defined by the directoryPriority parameter which is the primary sort criterion when arranging the files in a directory. (Note that this is only true for file systems that do not require files to be sorted in the directory, e.g. UDF). If any of the priority indicators is -1, the producer does not care about the priority and the *NeroAPI* will put the file where it thinks it fits.
Function AddDirectory(bstrName As String, lDirectoryPriority As Long) As NeroDirectoryContainer	

Method	Description
AddFile	Adds a file to the directory.
Function AddFile(bstrName As String, pProducer As NeroFileProducer, iHiWordLength As Long, iLoWordLength As Long, iPriority As Long, iDirectoryPriority As Long) As NeroDirectoryEntryContainer	
AddFile2	Adds a file to the directory.
Function AddFile2(bstrName As String, bstrSourcePath As String, iPriority As Long, iDirectoryPriority As Long) As NeroDirectoryEntryContainer	
Entry	Returns a NeroDirectoryEntryContainer for a given name.
Function Entry(bstrName As String) As NeroDirectoryEntryContainer	
Entry2	Returns a NeroDirectoryEntryContainer by a given index number
Function Entry2(iIndex As Long) As NeroDirectoryEntryContainer	
Item	Returns a directory entry by index number.
Function Item(Index As Long) As INeroDirectoryEntry - **default**	
RemoveEntry	Removes an entry.
Function RemoveEntry(bstrName As String) As Boolean	
SubDirectory	Retrieves a NeroDirectoryContainer for a subdirectory.
Function SubDirectory(bstrName As String) As NeroDirectoryContainer	

7.4.3. NeroDirectoryEntryContainer

This object provides a means of describing a file or directory.

Properties

Property	read-only	Description
Content	yes	Status of the NeroFileContainer.
Property Content(pContent As NeroFileContent) As NERO_ENTRY_ERROR		
DataStartSec	no	DataStartSec represents the sector number that will be saved into the directory structure.
Property DataStartSec As Long		
DirOrder	no	This property can be used to readjust the directory priority. Priority numbers will be used in upward order: the file with smaller values first.
Property DirOrder As Long		
FileNumber	no	This number will be written to the Mode 2 sub header.
Property FileNumber As Long		
FilePriority	yes	The file priority.
Property FilePriority As Long		
FixedDataStartSec	no	SetFixedDataStartSec represents the physical position of the file in the file system.
Property FixedDataStartSec As Long		

Property	read-only	Description
Flags	no	MODE2_FORM2, FIXED_INSIDE_VOLUME_SPACE, FIXED_OUTSIDE_VOLUME_SPACE, NO_OWN_CONTENT
Property Flags(bEnable As Boolean) As Long		
id	yes	The object's ID.
Property id As Long		
Name	yes	The file or directory name.
Property Name As String		
Priority	no	Defines the sequence of files on the media.
Property Priority As Long		
Size	no	Using this property, the file size can be changed after having added the entry to the directory.
Property Size(lHiWordSize As Long) As Long		
SourceFilePath	yes	The source file path. Empty if the file is generated.
Property SourceFilePath As String		
SubDirectory	yes	Will return NULL if the entry is not a directory.
Property SubDirectory As INeroDirectory		

7.4.4. NeroFile

An object to describe a single file.

Properties

Property	read-only	Description
EntryTime	no	Date and time for an entry.
Property EntryTime As Date		
Name	no	The name of an entry.
Property Name As String		
SourceFilePath	no	The path to the source file for the entry.
Property SourceFilePath As String		

7.4.5. NeroFileContent

Used by the Content property of the NeroDirectoryEntryContainer.

Properties

Property	read-only	Description
EndOfFile	yes	Indicates whether the end of a file has been reached.
Property EndOfFile As Boolean		
Error	yes	Indicates whether an error has occurred.
Property Error As Boolean		

Methods

Method	Description
Read	Read lLenght bytes into a buffer.
Function Read(pVar, lLength As Long) As Long	

7.4.6. NeroFileProducer

This object is specific in the sense that it can fire one event: **OnProduceFile**. The object itself is creatable by the client and is used as a way for *NeroCOM* to obtain file data while burning is underway. The client should create an instance of the object and pass it to *NeroCOM* where appropriate. *NeroCOM* will fire this object's **OnProduceFile** event when it needs file data for burning process.

Properties

Property	read-only	Description
id	yes	The NeroFileProducer object's ID.
Property id As Long		

Events

Event	Description
OnProduceFile	*NeroCOM* needs data for burning.
Event OnProduceFile(id As Long, pStream As NeroDataInputStream, pError As NERO_ENTRY_ERROR)	

7.4.7. NeroFiles

A collection of NeroFile objects.

Properties

Property	read-only	Description
Count	yes	Number of files in the collection.
Property Count As Long		

Events

Event	Description
Add	Add a file.
Sub Add(pNewVal As NeroFile)	
Item	Retrieve a file by its index number.
Function Item(Index As Long) As NeroFile	
Remove	Remove a file from the collection.
Sub Remove(Index As Long)	

7.4.8. NeroFileSystemDescContainer

This object represents the content of a file system.

Properties

Property	read-only	Description
BurnOptions	no	A combination of NERO_BURN_OPTIONS.
Property BurnOptions As NERO_BURN_OPTIONS		
Name	yes	The name for read access.
Property Name As String		
Name2	no	The name for write access.
Property Name2 As String		
Root	yes	The root of the file system as INeroDirectory.
Property Root As INeroDirectory		
RootDirectoryContainer	yes	The root of the file system as NeroDirectoryContainer.
Property RootDirectoryContainer As NeroDirectoryContainer		

7.4.9. NeroFolder

Used by various objects that accept or return folders.

Properties

Property	read-only	Description
Files	yes	Files in the folder.
Property Files As NeroFiles		
Folders	yes	A collection of subfolders.
Property Folders As NeroFolders		
Name	no	Name of the folder
Property Name As String		

7.4.10. NeroFolders

A collection of NeroFolder items.

Properties

Property	read-only	Description
Count	yes	A collection of NeroFolder items.
Property Count As Long		

Methods

Method	Description
Add	Add a NeroFolder.
Sub Add(pNewVal As NeroFolder)	
Item	Retrieve a NeroFolder by its index number.
Function Item(Index As Long) As NeroFolder – **default**	
Remove	Remove a NeroFolder.
Sub Remove(Index As Long)	

8. FAQ

This is a collection of frequently asked questions from the *NeroSDK* forum and *Nero AG* customer support.

If you cannot find the answer to your question here, please also refer to the FAQ section of the *NeroAPI* documentation. It covers a number of fundamental aspects and might prove useful with respect to your particular issue.

Where can I obtain *NeroCOM*?

NeroCOM is part of *Nero* 6. When you install *Nero* 6, *NeroCOM* becomes available automatically.

What is *NeroCOM*?

NeroCOM is a set of COM classes and interfaces that can be used in any programming language that supports COM philosophy (practically all modern languages for Win32 platforms including C/C++, Delphi, Visual Basic, Java, etc.).

Using *NeroCOM* you have access to almost all the functionality *NeroAPI* offers.

Can *NeroCOM* be used with .NET?

Yes.

I get an error "The type of the video item is not supported and you use NERO_NONENCODED_VIDEO_ITEM as NERO_VIDEO_ITEM_TYPE" when initiating the NeroVideoItem class. What can I do?

This is a known *NeroCOM* bug. It has been fixed in version 1.1.0.1 of *NeroCOM*.

I get an "Could not load referenced components" error. What can I do?

This usually refers to

- COMDLG32.OCX(1.2.0)
- comctl32.ocx(1.3.0)
- comct232.ocx(2.0.0)

This is a bug in Visual Studio .NET. Please follow the instructions given in Microsoft's knowledgebase article 318597:

http://support.microsoft.com/default.aspx?kbid=318597&product=vbNET.

Basically, the article tells you to install Visual Basic 6 or patch the Windows registry.

The OnDoneCDInfo event never fires after calling CDInfo. What's wrong?

This is a known problem and has been fixed in *NeroCOM* 1.1.0.1.

Can I use *NeroCOM* with *Nero* prior version 6.0?

No, that it not possible.

What does the "Class does not support Automation or does not support expected interface" error mean when adding a freestyle track?

This is a known problem which has been fixed with *NeroCOM* version 1.3.0.0.

Please upgrade your version of *Nero Burning ROM* to version 6.3.1.15 or later.

How do I burn more then one file (but not an entire directory)?

You have to get an instance of NeroFile for each file you want to burn, and add them, sequentially to a NeroFolder.

How do I use ImportIsoTrack?

You should call ImportIsoTrack on a drive of interest with a track number and flags. When the operation finishes OnDoneImport and OnDoneImport2 will be fired. You can handle any of them and use the information provided as you find fit. The most common task is to import a previous track, add more files to it and then burn it back.

How do I write an ISO image to to a CD?

It is as easy as using BurnImage of NeroDrive. Pass it a file path, flags and speed.

Can an Audio CD have a volume label?

No, an Audio CD cannot have a volume label. You can specify Artist and Title instead.

For a data CD use the name property of NeroISOTrack.

I cannot see Artist and Title of my Audio CD in the Windows Explorer. What is wrong?

The Windows Explorer does not display this information.

Try and see if *Nero Burning ROM's* Disc Info displays Artist/Title CD text information. If it does, then CD text is definitely there.

How do I pop up a messagebox after erasing a media?

You should handle the OnDoneErase event and display your message box there. The event belongs to NeroDrive type, not to the Nero object.

In addition, you must declare the variable with WithEvents.

Is it possible to copy a complete folder with all files and subfolders to a new folder on a disc?

Yes it is possible but you have to recursively build the ISO tree yourself.

My OnDriveStatusChange event handler is never called. What could be the reason?

If OnDriveStatusChange is the only event you do not receive, you should enable this event first by calling EnableStatusCallback(NDT_DISC_CHANGE, true).

Events that can be enabled with this method are initially disabled by design.

Can anyone give me some guidelines on how to burn all the files in the selected folder to the CD?

NeroFolder(s) and NeroFile(s) are the objects of interest here. You have to create a NeroFile for each file and NeroFolder for each folder you want to burn. You must also create a proper tree hierarchy just the way you want it on your destination disc.

Burning with *NeroCOM* is slow. How can I speed it up?

You can change the speed in the BurnIsoAudioCD call (Speed As Integer).

Passing zero should select the default burning speed and any other value should burn at that speed (if supported).

If you have been using a trial version of *Nero Burning ROM* past the evaluation period, only the slowest speed is available for burning!

How can I use “CD-Info” to determine whether a disc is empty or not?

You can use the FreeCapacityInBlocks property on the NeroCDInfo object to get the amount of free space. In a similar fashion, you can use the Tracks property to see if there are any tracks. If there no tracks, the disc is empty.

Why do I keep having problems when using the Image Recorder?

You probably forgot to handle the OnFileSelImage event. It must be handled in order to let *NeroCOM* know the name of the file to use for output image.

9. Bibliography

9.1. COM Books

If you decide to really dive into COM, this is the book:

Don Box: Essential COM