NeroCOM

v1.1

The Nero Type Library

NeroCOM will only work with
a fully installed Nero version!

Ahead Software 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 *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
Doc\NeroCOM.pdf	This document.

4.2. Examples

NeroFiddlesCOM is a simple example application to demonstrate how *NeroCOM* can be used from VisualBasic.

Path \ File Name	Description
NeroFiddlesCOM\NeroFiddlesCOMForm.frx	The one and only form's binary data file .
NeroFiddlesCOM\NeroFiddlesCOMForm.frm	The one and only form.
NeroFiddlesCOM\NeroFiddlesCOM.vbw	Visual Basic Workspace for the project.
NeroFiddlesCOM\NeroFiddlesCOM.vbp	Visual Basic Project file.
NeroFiddlesCOM\NeroFiddlesCOM.exe	Executable Example.

5. 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.EXE".

6. NeroCOM Ojects

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	NeroDrive
NeroVideoItem	NeroDrives
NeroFile	NeroSpeeds
NeroFolder	NeroCDInfo
NeroISOTrack	NeroTracks
NeroFileSystemDescContainer	NeroTrack
NeroFileProducer	NeroRelativeIndexBlkPositions
NeroAudioTrack	NeroFolders
NeroFreestyleTrack	NeroFiles
NeroFreestyleTracks	NeroCDStamp
NeroVideoItems	NeroDirectoryContainer
NeroAudioTracks	NeroDataInputStream
NeroFSSecNo	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 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.

6.1. 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 notification 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.

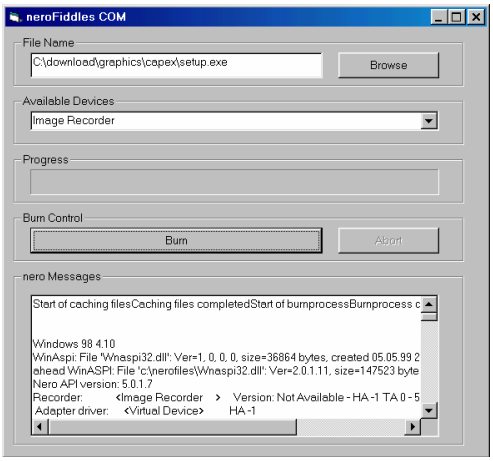
6.2. Important Considerations

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

- Nero object must first be created before any other and it should be the last to get destroyed as much depends on it.
- All Nero object's events that 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 *NeroAPI* forwarded by *NeroCOM* through Nero object's 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 writen 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!

7. Creating A Simple Visual Basic Application

7.1. Nero Keeps Fiddling!



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

The main difference now is the use of the COM Interface. The features of the application will remain the same, so we will name it *NeroFiddlesCOM*.

In software development the most important rule is the rule of consistency, and we will stick to that.

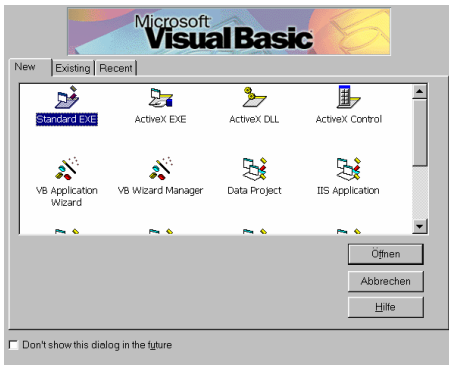
For those who do not know *NeroFiddles* - this simple application lists the available devices that can burn CDs. It lets the user choose a file and burns an ISO-CD, which contains this single file. Of course it's possible to use all the powerful features of *Nero* in your applications but a tutorial needs to be simple.

7.2. Creating The Framework

You need Microsoft Visual Basic for this tutorial! Also Nero and NeroCOM have to be installed on your system to run this sample!

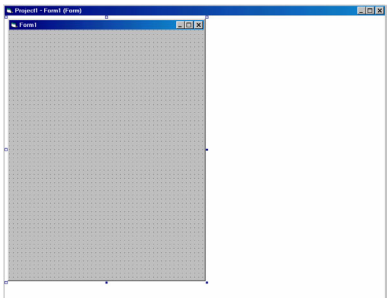
Open Microsoft Visual Basic. When the dialog displayed on the right pops up choose "Standard.EXE".

If the dialog does not display choose File/New from the menu.



7.3. Add CommandButtons and TextBoxes

Resize the resulting Form. We need some space.

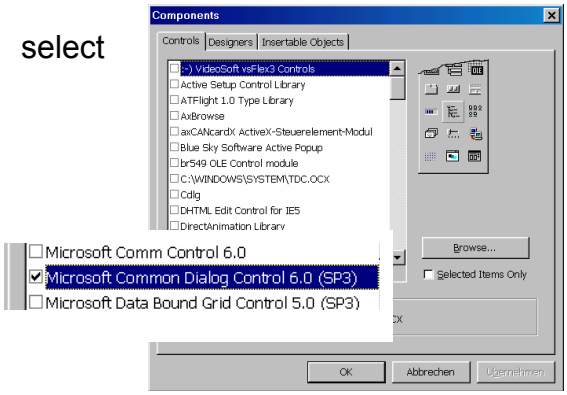


This application includes two Common Dialog controls for file selection. We need to make sure that you have access to those.

On the right you see what the Common Dialog symbol looks like. If you cannot find that on your Controls toolbar, you need to add it.

Right-click on the Controls toolbar and select “Components” from the context menu.

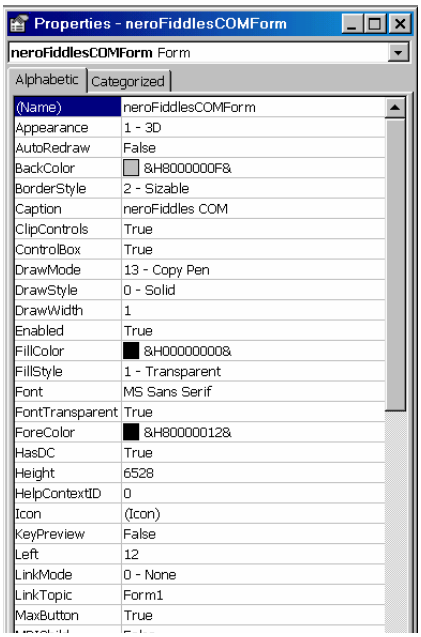
In the following dialog find an entry that looks like “Microsoft Common Dialog Control...”



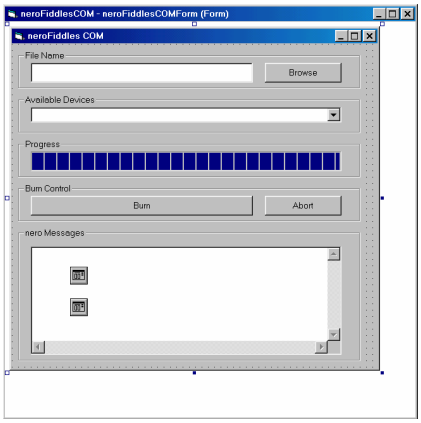
Activate it and click “OK”. Now the Common Dialog control should be available just like the other controls.

Select the main form, and change the “Name” property in the “Properties” dialog from “Form1” to “NeroFiddlesCOMForm”.

Whenever you’re requested to change a name or any other property for a control you will do that within this dialog window.



We need to add a few CommandButtons, TextBoxes and a progress bar.



- First create a TextBox and place it in the upper left corner of your form. Go to the properties dialog window and change the property “Name” from “Edit1” to “FileName”.
- Now create a CommandButton and name it “Browse”, also change the “Caption” property to “Browse”.
- Add a ComboBox control and name it “AvailableDevices”.
- Add a ProgressBar and name it “ProgressBar”
- Add a CommandButton and name it “Burn”, also change the caption to “Burn”.
- Add another CommandButton and name it “btnAbort”. Change the caption to “Abort.
- Now add a big TextBox, name it “edtMessages” and make it multi-line.
- Add two Common Dialog controls. Name the first “ImageFileDialog” and the second “SelectFileDialog”.
- Set the Filter property for SelectFileDialog to “All Files (*.*)|*.*|MP3 Files (*.mp3)|*.mp3”.
- Set the Filter property for ImageFileDialog to “All Files (*.*)|*.*|Nero Image Files (*.nrg)|*.nrg”.

7.4. Adding Functionality

We have created the required controls, named and altered them as needed and built the groundwork. Now it is time to do some real coding.

7.4.1. Public Definitions

First we need to define a few variables that will be used in all parts of the application.

We will define a Nero object, a drive object and a folder object and use them to initialise and access *NeroCOM*.

Right click on your the *NeroFiddlesCOM* Form and select “Code” from the context menu. At the very beginning of the file type this:

```
Public WithEvents nero As nero
Public drives As INeroDrives
```

```
Public WithEvents drive As NeroDrive
Public cnt As Integer
Public Folder As INeroFolder
Public strMessages As String
```

NameFromPath

This helper function will accept a full path a parameter, extract the file name, and return it to the caller of the function.

```
Function NameFromPath(strPath As String) As String
    Dim lngPos          As Long
    Dim strPart         As String
    Dim blnIncludesFile As Boolean

    lngPos = InStrRev(strPath, "\")
    blnIncludesFile = InStrRev(strPath, ".") > lngPos
    strPart = ""

    If lngPos > 0 Then
        If blnIncludesFile Then
            strPart = Right$(strPath, Len(strPath) - lngPos)
        End If
    End If

    NameFromPath = strPart
End Function
```

7.4.2. Form Initialization

First, a new Nero Object is created. Then its drive property is assigned to a drive object.

The system determines the number of available drives and fills their index number and names into the AvailableDevices List Box.

The virtual device is even present when there are no real devices with burning capability present in your system.

Change the Form_Initialize() function to make it look like this:

```
Private Sub Form_Initialize()
    Set nero = New nero
    ProgressBar.Value = 0
    strMessages = ""
    Dim drives As INeroDrives
    Set drives = nero.GetDrives(NERO_MEDIA_CDR)

    For myIndex = 0 To drives.Count - 1
        AvailableDevices.AddItem drives(myIndex).DeviceName, myIndex
    Next
```

```
    AvailableDevices.ListIndex = 0

ErrHandler:
    Exit Sub
End Sub
```

7.4.3. Command Button Event Handling

There are two types of events that we have to deal with in *NeroFiddlesCOM*.

One group of events is caused by the user pressing buttons, or selecting items. The other groups consists of events that are created by the *NeroCOM* interface.

We will create handlers for user events first.

Browse_Click

Double click on the “Browse” button. Visual Basic will add a new handler. We want a file select dialog to appear when we press that button, so that we can select the file that will be burned later.

Change the Browse handler to this:

```
Private Sub Browse_Click()
  SelectFileDialog.CancelError = True
  On Error GoTo ErrHandler
  SelectFileDialog.Flags = cdlOFNHideReadOnly
  SelectFileDialog.FilterIndex = 2
  SelectFileDialog.ShowOpen
  edtFileName.Text = SelectFileDialog.FileName
  Burn.Enabled = True
  Exit Sub

ErrHandler:
  Exit Sub
End Sub
```

btnAbort_Click

When burning is in progress, the user still has the option to cancel the process. This is done by simply calling the nero.Abort function. Double click on the “Abort” CommandButton and change the code to this:

```
Private Sub btnAbort_Click()
    nero.Abort
End Sub
```

Burn_Click

Whenever the “Burn” button is clicked, we want the following to happen:

- Abort CommandButton becomes enabled
- Browse CommandButton becomes disabled
- The drive is picked according to the current selection in the AvailableDevices ListBox

A new ISO track is generated and filled with the required data – destination file name, root folder. Then the drive function “BurnIsoAudioCD” is called, providing the ISO track object as a parameter.

Make the “Burn” event handler look like this:

```
Private Sub Burn_Click()
    btnAbort.Enabled = True
    Browse.Enabled = False
    Burn.Enabled = False

    Set Folder = New NeroFolder
    Dim drives As INeroDrives
    Set drives = nero.GetDrives(NERO_MEDIA_CDR)
    Set drive = drives(AvailableDevices.ListIndex)
    Dim isotrack As NeroISOTrack
    Set isotrack = New NeroISOTrack
    isotrack.Name = "TestTrack"
    isotrack.RootFolder = Folder
    Dim file As NeroFile
    Set file = New NeroFile
    Folder.Files.Add file
    file.Name = NameFromPath(edtFileName.Text)
    file.SourceFilePath = edtFileName.Text

    isotrack.BurnOptions = NERO_BURN_OPTION_CREATE_ISO_FS +
NERO_BURN_OPTION_USE_JOLIET

    drive.BurnIsoAudioCD "Pop Star", "Title", 0, isotrack, Nothing,
Nothing, NERO_BURN_FLAG_SIMULATE + NERO_BURN_FLAG_WRITE, 4,
NERO_MEDIA_CD
    GoTo quit

handle_error:
    strMessages = strMessages + Err.Description + Chr(13) + Chr(10) +
nero.LastError
    edtMessages = strMessages
quit:
End Sub
```

7.4.4. Handling NeroCOM Events

This is the second group of events we need to handle. These events are fired whenever *NeroCOM* needs information or wants to supply information about the current progress of operations.

drive_OnAddLogLine

Select “drive” from the “Object” ListBox in the code window.

Select “OnAddLogLine” from the “Procedure” ListBox.

Here we just want to display the text that *NeroCOM* offers us whenever a line is added to the log. Make your handler look like this:

```
Private Sub drive_OnAddLogLine(TextType As NEROLib.NERO_TEXT_TYPE,
                                                    Text As String)
    strMessages = strMessages + Text + Chr(13) + Chr(10)
    edtMessages = strMessages
End Sub
```

drive_OnDoneBurn

Now select “OnDoneBurn” from the “Procedure” ListBox.

Here we present the user with the results of the burn process, also presenting the complete error log that has been created during writing.

```
Private Sub drive_OnDoneBurn(StatusCode As NEROLib.NERO_BURN_ERROR)
    strMessages = strMessages + Chr(13) + Chr(10) + nero.ErrorLog +
Chr(13) + Chr(10)
    strMessages = strMessages + nero.LastError + Chr(13) + Chr(10)
    strMessages = strMessages + "Burn "
    If StatusCode <> NEROLib.NERO_BURN_OK Then
        strMessages = strMessages + "NOT (" & StatusCode & ") "
    End If
    strMessages = strMessages + "finished successfully!" + Chr(13) +
Chr(10)
    edtMessages = strMessages
    btnAbort.Enabled = False
    Browse.Enabled = True
    Burn.Enabled = True
    ProgressBar.Value = 0
End Sub
```

drive_OnAborted

OnAborted queries whether an operation shall be aborted. We always return false; if we choose to abort, we will explicitly call the Nero object’s abort method.

```
Private Sub drive_OnAborted(Abort As Boolean)
    Abort = False
End Sub
```

drive_OnProgress

Now we want to handle progress reports from *NeroCOM*.

Select “OnProgress” from the “Procedure” ListBox. Make it look like this:

```
Private Sub drive_OnProgress(ProgressInPercent As Long,
                                                    Abort As Boolean)
    Abort = False
    ProgressBar.Value = ProgressInPercent
End Sub
```

drive_OnDoneWaitForMedia

This event will be called when Nero has stopped waiting for the media.

```
Private Sub drive_OnDoneWaitForMedia(Success As Boolean)
    strMessages = strMessages + "Done waiting for media." + Chr(13) +
                                                                Chr(10)
    edtMessages = strMessages
End Sub
```

drive_OnSetPhase

We are also interested in finding out about the current phase *NeroCOM* is in – writing data, writing lead-in or lead-out parts of the CD-ROM and related information. Select “OnSetPhase” from the “Procedure” ListBox and change the procedure to make it look like this:

```
Private Sub drive_OnSetPhase(ByVal Text As String)
    strMessage = strMessage + Text
    edtMessages = strMessages
End Sub
```

nero_OnMegaFatal

This handler is called when a very critical error occurs.

```
Private Sub nero_OnMegaFatal()
    strMessage = strMessage + "A mega fatal error has occurred." +
                                                    Chr(13) + Chr(10)
    edtMessages = strMessages
End Sub
```

nero_OnFileSelImage

If the user decides to use the virtual device and not a real device more information is needed. The data that *NeroCOM* creates will be written to an image file. To provide the location of that file we need to handle the "OnFileSelImage" event.

Select "nero" from the Object ListBox in the code window. Select "OnFileSelImage" from the "Procedure" window and change it like this:

```
Private Sub nero_OnFileSelImage(FileName As String)
      ImageFileDialog.CancelError = True
      On Error GoTo ErrHandler
      ImageFileDialog.Flags = cdlOFNHideReadOnly
      ImageFileDialog.FilterIndex = 2
      ImageFileDialog.ShowOpen
      FileName = ImageFileDialog.FileName
      Exit Sub

ErrHandler:
      Exit Sub
End Sub
```

nero_OnNonEmptyCDRW

This event will be called whenever the user tries to write on a non-empty CD-RW. We merely return NERO_RETURN_EXIT, although of course more sophisticated ways of handling this message are possible, e.g. by asking the user if he wants to erase the CD-RW.

```
Private Sub nero_OnNonEmptyCDRW(Response As NEROLib.NERO_RESPONSE)
    strMessage = strMessage + "CD-RW not empty!" + Chr(13) + Chr(10)
    edtMessages = strMessages
    Response = NERO_RETURN_EXIT
End Sub
```

nero_OnWaitCD

This event will be called when Nero is waiting for a CD to be inserted into the drive. We add the string that is passed to the log window.

```
Private Sub nero_OnWaitCD(WaitCD As NEROLib.NERO_WAITCD_TYPE,
                                        WaitCDLocalizedText As String)
    strMessage = strMessage + WaitCDLocalizedText + Chr(13) + Chr(10)
    edtMessages = strMessages
End Sub
```

nero_OnWaitCDMediaInfo

This event will be fired when Nero is waiting for a particular type of media.

```
Private Sub nero_OnWaitCDMediaInfo(LastDetectedMedia As
NEROLib.NERO_MEDIA_TYPE, LastDetectedMediaName As String, RequestedMedia
As NEROLib.NERO_MEDIA_TYPE, RequestedMediaName As String)

    strMessage = strMessage + "Waiting for a particular media type:" +
                                                    Chr(13) + Chr(10)
    strMessage = strMessage + RequestedMediaName + Chr(13) + Chr(10)
    edtMessages = strMessages
End Sub
```

nero_OnWaitCDReminder

This event will occur when Nero has been waiting for the insertion of a CD for quite a while.

```
Private Sub nero_OnWaitCDReminder()
    strMessage = strMessage + "Still waiting for CD..." + Chr(13) +
                                                          Chr(10)
    edtMessages = strMessages
End Sub
```

7.5. Running The Application

That's it! Now the required functionality to pick an MP3 file, select a device for burning, select a file for image creation and actually burning it has been implemented. Choose "Run/Start" from the menu and see what happens.

8. NeroCOM Enumerations and Objects

8.1. Enumerations

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

8.1.1. 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).

8.1.2. 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.

8.1.3. 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.

8.1.4. 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 secs 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).

8.1.5. 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.

8.1.6. 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.

8.1.7. 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

Enumerator	Description
	burn process.
NERO_BURN_FLAG_DVDP_BURN_30MM_AT_LEAST	DVD+R/RW high compability 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 Filesystem after writing. Works for ISO only.
NERO_BURN_FLAG_WRITE	Really write at the end.

8.1.8. 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.
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.

8.1.9. 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.

Enumerator	Description
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.

8.1.10. 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.

8.1.11. 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.

8.1.12. 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 a other 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 occured/ drive is ready.

8.1.13. 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.

8.1.14. 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.

8.1.15. 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.

8.1.16. NERO_ENTRY_ERROR

This type is used by a 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 overriden function has tried to get an other 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.

8.1.17. 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 recommanded 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.

8.1.18. 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.

8.1.19. 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.

8.1.20. 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 filesystem may have been imported nevertheless.
NIDTR_NO_ERROR	No error.
NIDTR_READ_ERROR	Error while reading from the disc. Parts of the filesystem may have been imported nevertheless.

8.1.21. 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.

8.1.22. 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.
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.

Enumerator	Description
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.

8.1.23. 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 Packetwriting.
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.

Enumerator	Description
NERO_MEDIA_NO_DVD_P_RW	Exclude DVD+RW.
NERO_MEDIA_NONE	No media present.
NERO_MEDIA_PACKETW	A bit mask for packetwriting.
NERO_MEDIA_VPACKET	Variable Packetwriting.

8.1.24. 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 autoinsert 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.

8.1.25. 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.

8.1.26. 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.

8.1.27. 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.

8.1.28. 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.

8.1.29. 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!\n\nYour 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\ncontains 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!\n\nYour recorder requires this eject between simulation and burning. Please reinsert the\n 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."

Enumerator	Description
NERO_WAITCD_MULTISESSION_SIM	"To simulate this multisession compilation you need the disc, that 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)."
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.\n\nPlease insert original disc…"

8.1.30. NeroFSBlockAccessExtensionsType

Type of an extension.

Enumerator	Description
etBlankAreaControlExtension	
etDVDPRWFormatExtension	
etHDPartitionInfo	
etHDUsedBlockAccessExtention	
etIllegalBlockAccessExtension	
etMRWReadDefectiveManagementInfo	
etSectorMappingControlExtension	
etSectorPatchControlExtension	

8.1.31. 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.

8.1.32. NeroFSTrackType

Used by the PartitionType member of the NeroFSPartitionInfo object.

Enumerator	Description
vtAudio	Audio Track.
vtData	Data Track.

8.2. General Objects

8.2.1. 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		

8.2.2. 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
EnableOverburn	read descr.	Nero 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 NeroAPI's entire 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		
LastError	yes	Retrieve the last error from the NeroAPI.
Property LastError As String		

Property	read-only	Description
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 propterty 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()	
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	

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 autoinsert 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)	
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	Megafatal internal problem that cannot be fixed!
OnMegaFatal()	

Event	Description
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 an other CD.
Event OnNonEmptyCDRW(Response As NERO_RESPONSE)	
OnNoTrackFound	Problem because we do not have any track.
OnNoTrackFound()	
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()	
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)	
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 informations 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()	

8.2.3. 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 lenght 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 occured.
Event OnError(Error As Boolean)	
OnReadAudioBurn	Provide Audio Data for the burn engine.
Event OnReadAudioBurn(Length As Long, Data)	

8.2.4. 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)	

8.2.5. 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		

8.2.6. NeroCDStamp

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

8.2.7. 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.
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		
DeviceReady	yes	Indicates whether the drive is ready or not.
Property DeviceReady As Boolean		
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		

Property	read-only	Description
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		
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.
Property ReadSpeeds As NeroSpeeds		
SupportedAccessModes	yes	Supported access modes.
Property SupportedAccessModes As Long		
WriteSpeeds	yes	Available speeds for write access.
Property WriteSpeeds As NeroSpeeds		

Methods

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)	
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)	
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)	
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()	

Method	Description
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	
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)	
ImportIsoTrack	Import an ISO track from a previous session.
Sub ImportIsoTrack(TrackNumber As Long, dwFlags As NERO_IMPORT_ISO_TRACK_FLAGS)	
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)	
UpdateDeviceInfo	Force an update of the device information.
Function UpdateDeviceInfo(MediaType As NERO_MEDIA_TYPE) As NERO_DRIVE_ERROR	
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)	
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)	

Event	Description
OnDoneErase	Erasing CDRW 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)	
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)	

8.2.8. 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**	

8.2.9. NeroFreestyleTrack

This objects 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		
LengthInBlocks	no	The lenght 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)	

8.2.10. 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)	
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)	

8.2.11. 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		

8.2.12. 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, that 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		
VolumeSet	no	This name is used when multiple media are part of one logical unit.
Property VolumeSet As String		

8.2.13. 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)	

8.2.14. 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	

8.2.15. 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		
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		

8.2.16. 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**	

8.2.17. 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 pausse after this item.
Property PauseAfterItem As Long		
SourceFileName		MPG or JPG or AVI file name.
Property SourceFileName As String		

8.2.18. 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)	

8.3. Packet Writing Objects

8.3.1. 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 filesystem 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	

8.3.2. 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()	

Method	Description
ReadSectorsBuffered	The buffered reading method will use a cache to optimize filesystem 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	
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	

8.3.3. 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		

8.3.4. 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		

8.4. File System Objects

8.4.1. NeroDataInputStream

Used by the file producer object.

Methods

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

8.4.2. 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 directoy object when provided a name and a pritority. 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 fileentry order in a directory is defined by the directoryPriority parameter which is the primary sort criterium when arranging the files in a directory. (Note that this is only true for filesystems that do not require files to be sorted in the directory, e.g. UDF). If any of the priority specifiers 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 an 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	

8.4.3. 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 subheader.
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 filesystem.
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		

8.4.4. 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		

8.4.5. 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	

8.4.6. 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)	

8.4.7. 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)	

8.4.8. 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		

8.4.9. 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		

8.4.10. 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)	

9. Bibliography

9.1. COM Books

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

Don Box: Essential COM