branch: master ExtractFromDataRun / ExtractFromDatarun.au3

jschicht 2 months ago Version 1.0.0.2

1 contributor

file 625 lines (595 sloc) 25.031 kb Open Edit Raw Blame History Delete

```
#Region ;**** Directives created by AutoIt3Wrapper_GUI ****
#AutoIt3Wrapper_UseX64=y
#AutoIt3Wrapper_Res_Comment=Using dataruns to extract files from NTFS
#AutoIt3Wrapper_Res_Description=Using dataruns to extract files from NTFS
#AutoIt3Wrapper_Res_Fileversion=1.0.0.2
#AutoIt3Wrapper_Res_requestedExecutionLevel=asInvoker
#EndRegion ;**** Directives created by AutoIt3Wrapper_GUI ****
#include <GUIConstantsEx.au3>
#include <WindowsConstants.au3>
#include <StaticConstants.au3>
#include <EditConstants.au3>
#include <GuiEdit.au3>
#Include <WinAPIEx.au3>
#Include <FileConstants.au3>
;#Include <array.au3>

;Mostly code from NTFSFileExtractor

Global $RUN_VCN[1], $RUN_Clusters[1], $MFT_RUN_Clusters[1], $MFT_RUN_VCN[1], $DataQ[1], $AttrQ[1], $BytesPerCluster
Global $IsCompressed = False, $IsSparse = False
Global $outputpath=@ScriptDir, $hDisk, $sBuffer, $DataRun, $DATA_InitSize, $DATA_RealSize, $ImageOffset = 0, $ADS_Name
Global $TargetImageFile, $Entries, $IsImage=False, $IsPhysicalDrive=False, $ComboPhysicalDrives, $Combo, $IsShadowCopy=False

$Form = GUICreate("Extract from dataruns", 560, 280, -1, -1)
$ComboPhysicalDrives = GUICtrlCreateCombo("", 180, 5, 305, 20)
$buttonScanPhysicalDrives = GUICtrlCreateButton("Scan Physical", 5, 5, 80, 20)
$buttonScanShadowCopies = GUICtrlCreateButton("Scan Shadows", 90, 5, 80, 20)
$buttonTestPhysicalDrive = GUICtrlCreateButton("<-- Test it", 495, 5, 60, 20)
$Combo = GUICtrlCreateCombo("", 20, 40, 360, 20)
$buttonDrive = GUICtrlCreateButton("Rescan Mounted Drives", 425, 40, 130, 20)
$LabelDataRun = GUICtrlCreateLabel("DataRun:",20,70,80,20)
$InputDataRun = GUICtrlCreateInput("",100,70,400,20)

$LabelDataRealSize = GUICtrlCreateLabel("Real data size:",20,100,80,20)
$InputDataRealSize = GUICtrlCreateInput("0",100,100,100,20)
$LabelDataInitSize = GUICtrlCreateLabel("Init data size:",210,100,80,20)
$InputDataInitSize = GUICtrlCreateInput("0",290,100,100,20)
$LabelFileName = GUICtrlCreateLabel("Name of file:",20,130,80,20)
$InputFileName = GUICtrlCreateInput("file.ext",100,130,100,20)
$checkCompression = GUICtrlCreateCheckbox("IsCompressed", 210, 125, 95, 20)
$checkSparse = GUICtrlCreateCheckbox("IsSparse", 210, 145, 95, 20)
$ButtonOutput = GUICtrlCreateButton("Change Output", 400, 95, 100, 20)
$ButtonImage = GUICtrlCreateButton("Browse for image", 400, 125, 100, 20)
$ButtonStart = GUICtrlCreateButton("Start", 400, 150, 100, 20)
$myctredit = GUICtrlCreateEdit("Current output folder: " & $outputpath & @CRLF, 0, 180, 560, 100, $ES_AUTOVSCROLL + $WS_VSCR
_GUICtrlEdit_SetLimitText($myctredit, 128000)
;_GetPhysicalDrives()
_GetMountedDrivesInfo()
GUISetState(@SW_SHOW)

While 1
$nMsg = GUIGetMsg()
Select

        Case $nMsg = $ButtonImage
                _ProcessImage()
                $IsImage = True
```

```
                $IsShadowCopy = False
                $IsPhysicalDrive = False
        Case $nMsg = $ButtonOutput
                $newoutputpath = FileSelectFolder("Select output folder.", "",7,$outputpath)
                If Not @error then
                   _DisplayInfo("New output folder: " & $newoutputpath & @CRLF)
                   $outputpath = $newoutputpath
                EndIf
        Case $nMsg = $ButtonStart
                _Main()
        Case $nMsg = $buttonDrive
                _GetMountedDrivesInfo()
                $IsImage = False
                $IsShadowCopy = False
                $IsPhysicalDrive = False
        Case $nMsg = $GUI_EVENT_CLOSE
                 Exit
        Case $nMsg = $buttonScanPhysicalDrives
                _GetPhysicalDrives("PhysicalDrive")
                $IsShadowCopy = False
                $IsPhysicalDrive = True
                $IsImage = False
        Case $nMsg = $buttonScanShadowCopies
                _GetPhysicalDrives("GLOBALROOT\Device\HarddiskVolumeShadowCopy")
                $IsShadowCopy = True
                $IsPhysicalDrive = False

                $IsImage = False
        Case $nMsg = $buttonTestPhysicalDrive
                _TestPhysicalDrive()
EndSelect
WEnd

Func _Main()
        Global $RUN_VCN[1], $RUN_Clusters[1]
        $DATA_InitSize = GUICtrlRead($InputDataInitSize)
        $DATA_RealSize = GUICtrlRead($InputDataRealSize)
        If $DATA_InitSize="" Or $DATA_RealSize="" Or Not StringIsDigit($DATA_InitSize) Or Not StringIsDigit($DATA_RealSize)
                _DisplayInfo("Error: Invalid value for data real size or data init size" & @CRLF)
                Return
        EndIf
        $DataRun = GUICtrlRead($InputDataRun)
        $DataRun = StringStripWS($DataRun,8)
        If $DataRun = "" Or Not StringIsXDigit($DataRun) Then
                _DisplayInfo("Error: Datarun input not valid: " & $DataRun & @CRLF)
                Return
        EndIf
        $TargetFileName = GUICtrlRead($InputFileName)
        If $TargetFileName="" Then
                _DisplayInfo("Error: Need to set something for filename to extract to" & @CRLF)
                Return
        EndIf
        $TargetFileName = StringMid($TargetFileName,StringInStr($TargetFileName,"\",0,-1)+1)
        Select
                Case $IsImage = True
                        $TargetDrive = "Img"
                        $ImageOffset = Int(StringMid(GUICtrlRead($Combo),10),2)
                        _DisplayInfo(@CRLF & "Target is: " & GUICtrlRead($Combo) & @CRLF)
                        _DisplayInfo("Target is: " & $TargetImageFile & @CRLF)
                        _DisplayInfo("Volume at offset: " & $ImageOffset & @CRLF)
                        $hDisk = _WinAPI_CreateFile($TargetImageFile,2,2,7)
                        If $hDisk = 0 Then _DisplayInfo("CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
                Case $IsPhysicalDrive = True
                        $TargetDrive = "PD"&StringMid($TargetImageFile,18)
                        $ImageOffset = Int(StringMid(GUICtrlRead($Combo),10),2)
                        _DisplayInfo("Target drive is: " & $TargetImageFile & @CRLF)
                        _DisplayInfo("Volume at offset: " & $ImageOffset & @CRLF)
                        $hDisk = _WinAPI_CreateFile($TargetImageFile,2,2,7)
                        If $hDisk = 0 Then _DisplayInfo("CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
                Case $IsShadowCopy = True
                        $TargetDrive = "SC"&StringMid($TargetImageFile,47)
                        $ImageOffset = Int(StringMid(GUICtrlRead($Combo),10),2)
                        _DisplayInfo("Target drive is: " & $TargetImageFile & @CRLF)
                        _DisplayInfo("Volume at offset: " & $ImageOffset & @CRLF)
                        $hDisk = _WinAPI_CreateFile($TargetImageFile,2,2,7)
```

```
                        If $hDisk = 0 Then _DisplayInfo("CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
                Case Else

                        $TargetDrive = StringMid(GUICtrlRead($Combo),1,1)
                        $hDisk = _WinAPI_CreateFile("\\.\" & $TargetDrive&":",2,2,7)
                        If $hDisk = 0 Then _DisplayInfo("CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
        EndSelect

#cs
        If $IsImage Then
                $ImageOffset = Int(StringMid(GUICtrlRead($Combo),10),2)
                $hDisk = _WinAPI_CreateFile("\\.\" & $TargetImageFile,2,2,7)
                If $hDisk = 0 Then
                        _DisplayInfo("CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
                        Return
                EndIf
        ElseIf $IsPhysicalDrive=False Then
                $TargetDrive = StringMid(GUICtrlRead($Combo),1,2)
                $hDisk = _WinAPI_CreateFile("\\.\" & $TargetDrive,2,2,7)
                If $hDisk = 0 Then
                        _DisplayInfo("CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
                        Return
                EndIf
        ElseIf $IsPhysicalDrive=True Then
                $TargetDrive = StringMid($TargetImageFile,18)
                $ImageOffset = Int(StringMid(GUICtrlRead($Combo),10),2)
                _DisplayInfo("Target drive is: " & $TargetImageFile)
                _DisplayInfo("Volume at offset: " & $ImageOffset)
                $hDisk = _WinAPI_CreateFile($TargetImageFile,2,2,7)
                If $hDisk = 0 Then _DisplayInfo("CreateFile: " & _WinAPI_GetLastErrorMessage())
        EndIf
        _DisplayInfo(GUICtrlRead($Combo) & @CRLF)
#ce
        If GUICtrlRead($checkCompression)=1 Then $IsCompressed=True
        If GUICtrlRead($checkSparse)=1 Then $IsSparse=True
        _WinAPI_SetFilePointerEx($hDisk, $ImageOffset, $FILE_BEGIN)
        $BootRecord = _GetDiskConstants()
        If $BootRecord = "" Then
                _DisplayInfo("Error: Unable to read Boot Sector" & @CRLF)
                Return
        EndIf
        _ExtractDataRuns()
        _ExtractFile($TargetFileName)
EndFunc

Func _ExtractDataRuns()
        $r=UBound($RUN_Clusters)
        ReDim $RUN_Clusters[$r + 400], $RUN_VCN[$r + 400]
        $i=1
        $RUN_VCN[0] = 0
        $BaseVCN = $RUN_VCN[0]
        If $DataRun = "" Then $DataRun = "00"
        Do

                $RunListID = StringMid($DataRun,$i,2)
                If $RunListID = "00" Then ExitLoop
                $i += 2
                $RunListClustersLength = Dec(StringMid($RunListID,2,1))
                $RunListVCNLength = Dec(StringMid($RunListID,1,1))
                $RunListClusters = Dec(_SwapEndian(StringMid($DataRun,$i,$RunListClustersLength*2)),2)
                $i += $RunListClustersLength*2
                $RunListVCN = _SwapEndian(StringMid($DataRun, $i, $RunListVCNLength*2))
          ;next line handles positive or negative move
                $BaseVCN += Dec($RunListVCN,2)-(($r>1) And (Dec(StringMid($RunListVCN,1,1))>7))*Dec(StringMid("1000000000000
                If $RunListVCN <> "" Then
                        $RunListVCN = $BaseVCN
                Else
                        $RunListVCN = 0
                EndIf
                If (($RunListVCN=0) And ($RunListClusters>16) And (Mod($RunListClusters,16)>0)) Then
                        ;may be sparse section at end of Compression Signature
                        $RUN_Clusters[$r] = Mod($RunListClusters,16)
                        $RUN_VCN[$r] = $RunListVCN
                        $RunListClusters -= Mod($RunListClusters,16)
                        $r += 1
```

```
                ElseIf (($RunListClusters>16) And (Mod($RunListClusters,16)>0)) Then
                        ;may be compressed data section at start of Compression Signature
                        $RUN_Clusters[$r] = $RunListClusters-Mod($RunListClusters,16)
                        $RUN_VCN[$r] = $RunListVCN
                        $RunListVCN += $RUN_Clusters[$r]
                        $RunListClusters = Mod($RunListClusters,16)
                        $r += 1
                EndIf
                ;just normal or sparse data
                $RUN_Clusters[$r] = $RunListClusters
                $RUN_VCN[$r] = $RunListVCN
                $r += 1
                $i += $RunListVCNLength*2
        Until $i > StringLen($DataRun)
        ReDim $RUN_Clusters[$r], $RUN_VCN[$r]
EndFunc

Func _ExtractFile($ADS_Name)
    $cBuffer = DllStructCreate("byte[" & $BytesPerCluster * 16 & "]")
    $zflag = 0
        If FileExists($ADS_Name) Then FileDelete($ADS_Name)
        $ADS_Name = "\\.\"&$outputpath&"\"&$ADS_Name
        _DisplayInfo("Output: " & $ADS_Name & @CRLF)
        $hFile = _WinAPI_CreateFile($ADS_Name,3,6,7)
        If $hFile Then
                Select
                        Case UBound($RUN_VCN) = 1               ;no data, do nothing
                        Case UBound($RUN_VCN) = 2       ;may be normal or sparse
                                If $RUN_VCN[1] = 0 And $IsSparse Then           ;sparse

                                        $FileSize = _DoSparse(1, $hFile, $DATA_InitSize)
                                Else                                                                    ;normal
                                        $FileSize = _DoNormal(1, $hFile, $cBuffer, $DATA_InitSize)
                                EndIf
                    Case Else                                   ;may be compressed
                                _DoCompressed($hFile, $cBuffer, "")
                EndSelect
                If $DATA_RealSize > $DATA_InitSize Then
                    $FileSize = _WriteZeros($hfile, $DATA_RealSize - $DATA_InitSize)
                EndIf
                _WinAPI_CloseHandle($hFile)
                Return
        Else
                _DisplayInfo("Error: CreateFile returned: " & _WinAPI_GetLastErrorMessage() & @CRLF)
        EndIf
EndFunc

Func _DoNormal($r, $hFile, $cBuffer, $FileSize)
        Local $nBytes
        _WinAPI_SetFilePointerEx($hDisk, $ImageOffset+$RUN_VCN[$r]*$BytesPerCluster, $FILE_BEGIN)
        $i = $RUN_Clusters[$r]
        While $i > 16 And $FileSize > $BytesPerCluster * 16
                _WinAPI_ReadFile($hDisk, DllStructGetPtr($cBuffer), $BytesPerCluster * 16, $nBytes)
                _WinAPI_WriteFile($hFile, DllStructGetPtr($cBuffer), $BytesPerCluster * 16, $nBytes)
                $i -= 16
                $FileSize -= $BytesPerCluster * 16
                $ProgressSize = $FileSize
        WEnd
        If $i = 0 Or $FileSize = 0 Then Return $FileSize
        If $i > 16 Then $i = 16
        _WinAPI_ReadFile($hDisk, DllStructGetPtr($cBuffer), $BytesPerCluster * $i, $nBytes)
        If $FileSize > $BytesPerCluster * $i Then
                _WinAPI_WriteFile($hFile, DllStructGetPtr($cBuffer), $BytesPerCluster * $i, $nBytes)
                $FileSize -= $BytesPerCluster * $i
                $ProgressSize = $FileSize
                Return $FileSize
        Else
                _WinAPI_WriteFile($hFile, DllStructGetPtr($cBuffer), $FileSize, $nBytes)
                $ProgressSize = 0
                Return 0
        EndIf
EndFunc

Func _DoCompressed($hFile, $cBuffer, $record)
        Local $nBytes
```

```
279         $r=1
280         $FileSize = $DATA_InitSize
281         $ProgressSize = $FileSize
282         Do
283                 _WinAPI_SetFilePointerEx($hDisk, $ImageOffset+$RUN_VCN[$r]*$BytesPerCluster, $FILE_BEGIN)
284                 $i = $RUN_Clusters[$r]
285                 If (($RUN_VCN[$r+1]=0) And ($i+$RUN_Clusters[$r+1]=16) And $IsCompressed) Then
286                         _WinAPI_ReadFile($hDisk, DllStructGetPtr($cBuffer), $BytesPerCluster * $i, $nBytes)
287                         ConsoleWrite(_HexEncode(DllStructGetData($cBuffer,1)) & @CRLF)
288                         $Decompressed = _LZNTDecompress($cBuffer, $BytesPerCluster * $i)
289                         If IsString($Decompressed) Then
290                                 If $r = 1 Then
291                                         _DisplayInfo("Error: Decompression error" & @CRLF)
292                                 Else
293                                         _DisplayInfo("Error: Decompression error (partial write)" & @CRLF)
294                                 EndIf
295                                 Return
296                         Else            ;$Decompressed is an array
297                                 Local $dBuffer = DllStructCreate("byte[" & $Decompressed[1] & "]")
298                                 DllStructSetData($dBuffer, 1, $Decompressed[0])
299                         EndIf
300                         If $FileSize > $Decompressed[1] Then
301                                 _WinAPI_WriteFile($hFile, DllStructGetPtr($dBuffer), $Decompressed[1], $nBytes)
302                                 $FileSize -= $Decompressed[1]
303                                 $ProgressSize = $FileSize
304                         Else
305                                 _WinAPI_WriteFile($hFile, DllStructGetPtr($dBuffer), $FileSize, $nBytes)
306                         EndIf
307                         $r += 1
308                 ElseIf $RUN_VCN[$r]=0 Then
309                         $FileSize = _DoSparse($r, $hFile, $FileSize)
310                         $ProgressSize = 0
311                 Else
312                         $FileSize = _DoNormal($r, $hFile, $cBuffer, $FileSize)
313                         $ProgressSize = 0
314                 EndIf
315                 $r += 1
316         Until $r > UBound($RUN_VCN)-2
317         If $r = UBound($RUN_VCN)-1 Then
318                 If $RUN_VCN[$r]=0 Then
319                         $FileSize = _DoSparse($r, $hFile, $FileSize)
320                         $ProgressSize = 0
321                 Else
322                         $FileSize = _DoNormal($r, $hFile, $cBuffer, $FileSize)
323                         $ProgressSize = 0
324                 EndIf
325         EndIf
326 EndFunc
327
328 Func _DoSparse($r,$hFile,$FileSize)
329         MsgBox(0,"Info","_DoSparse()")
330         Local $nBytes
331         If Not IsDllStruct($sBuffer) Then _CreateSparseBuffer()
332         $i = $RUN_Clusters[$r]
333         While $i > 16 And $FileSize > $BytesPerCluster * 16
334                 _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $BytesPerCluster * 16, $nBytes)
335                 $i -= 16
336                 $FileSize -= $BytesPerCluster * 16
337                 $ProgressSize = $FileSize
338         WEnd
339         If $i <> 0 Then
340                 If $FileSize > $BytesPerCluster * $i Then
341                         _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $BytesPerCluster * $i, $nBytes)
342                         $FileSize -= $BytesPerCluster * $i
343                         $ProgressSize = $FileSize
344                 Else
345                         _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $FileSize, $nBytes)
346                         $ProgressSize = 0
347                         Return 0
348                 EndIf
349         EndIf
350         Return $FileSize
351 EndFunc
```

```
Func _CreateSparseBuffer()
        Global $sBuffer = DllStructCreate("byte[" & $BytesPerCluster * 16 & "]")
        For $i = 1 To $BytesPerCluster * 16
                DllStructSetData ($sBuffer, $i, 0)
        Next
EndFunc

Func _WriteZeros($hfile, $count)
        Local $nBytes
        If Not IsDllStruct($sBuffer) Then _CreateSparseBuffer()
        While $count > $BytesPerCluster * 16
                _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $BytesPerCluster * 16, $nBytes)
                $count -= $BytesPerCluster * 16
                $ProgressSize = $DATA_RealSize - $count
        WEnd
        If $count <> 0 Then _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $count, $nBytes)
        $ProgressSize = $DATA_RealSize
        Return 0
EndFunc

Func _LZNTDecompress($tInput, $Size)    ;note function returns a null string if error, or an array if no error
        Local $tOutput[2]
        Local $cBuffer = DllStructCreate("byte[" & $BytesPerCluster*16 & "]")
    Local $a_Call = DllCall("ntdll.dll", "int", "RtlDecompressBuffer", _
            "ushort", 2, _
            "ptr", DllStructGetPtr($cBuffer), _
            "dword", DllStructGetSize($cBuffer), _
            "ptr", DllStructGetPtr($tInput), _
            "dword", $Size, _
            "dword*", 0)


    If @error Or $a_Call[0] Then        ;if $a_Call[0]=0 then output size is in $a_Call[6], otherwise $a_Call[6] is invalid
        Return SetError(1, 0, "") ; error decompressing
    EndIf
    Local $Decompressed = DllStructCreate("byte[" & $a_Call[6] & "]", DllStructGetPtr($cBuffer))
        $tOutput[0] = DllStructGetData($Decompressed, 1)
        $tOutput[1] = $a_Call[6]
    Return SetError(0, 0, $tOutput)
EndFunc

Func _SwapEndian($iHex)
        Return StringMid(Binary(Dec($iHex,2)),3, StringLen($iHex))
EndFunc

Func _GetDiskConstants()
        Local $nbytes
        $tBuffer = DllStructCreate("byte[512]")
        $read = _WinAPI_ReadFile($hDisk, DllStructGetPtr($tBuffer), 512, $nBytes)
        If $read = 0 Then Return ""
        $record = DllStructGetData($tBuffer, 1)
        $BytesPerSector = Dec(_SwapEndian(StringMid($record,25,4)),2)
        $SectorsPerCluster = Dec(_SwapEndian(StringMid($record,29,2)),2)
        $BytesPerCluster = $BytesPerSector * $SectorsPerCluster
        $LogicalClusterNumberforthefileMFT = Dec(_SwapEndian(StringMid($record,99,8)),2)
        $MFT_Offset = $BytesPerCluster * $LogicalClusterNumberforthefileMFT
        $ClustersPerFileRecordSegment = Dec(_SwapEndian(StringMid($record,131,8)),2)
        If $ClustersPerFileRecordSegment > 127 Then
                $MFT_Record_Size = 2 ^ (256 - $ClustersPerFileRecordSegment)
        Else
                $MFT_Record_Size = $BytesPerCluster * $ClustersPerFileRecordSegment
        EndIf
        Return $record
EndFunc

Func _DisplayInfo($DebugInfo)
        GUICtrlSetData($myctredit, $DebugInfo, 1)
EndFunc

Func _ProcessImage()
        $TargetImageFile = FileOpenDialog("Select image file",@ScriptDir,"All (*.*)")
        If @error then Return
        $TargetImageFile = "\\.\"&$TargetImageFile
        _DisplayInfo("Selected disk image file: " & $TargetImageFile & @CRLF)
```

```
        GUICtrlSetData($Combo,"","")
        $Entries = ''
        _CheckMBR()
        GUICtrlSetData($Combo,$Entries,StringMid($Entries, 1, StringInStr($Entries, "|") -1))
        If $Entries = "" Then _DisplayInfo("Sorry, no NTFS volume found in that file." & @CRLF)
EndFunc   ;==>_ProcessImage

Func _CheckMBR()

        Local $nbytes, $PartitionNumber, $PartitionEntry,$FilesystemDescriptor
        Local $StartingSector,$NumberOfSectors
        Local $hImage = _WinAPI_CreateFile($TargetImageFile,2,2,7)
        $tBuffer = DllStructCreate("byte[512]")
        Local $read = _WinAPI_ReadFile($hImage, DllStructGetPtr($tBuffer), 512, $nBytes)
        If $read = 0 Then Return ""
        Local $sector = DllStructGetData($tBuffer, 1)
        For $PartitionNumber = 0 To 3
                $PartitionEntry = StringMid($sector,($PartitionNumber*32)+3+892,32)
                If $PartitionEntry = "00000000000000000000000000000000" Then ExitLoop ; No more entries
                $FilesystemDescriptor = StringMid($PartitionEntry,9,2)
                $StartingSector = Dec(_SwapEndian(StringMid($PartitionEntry,17,8)),2)
                $NumberOfSectors = Dec(_SwapEndian(StringMid($PartitionEntry,25,8)),2)
                If ($FilesystemDescriptor = "EE" and $StartingSector = 1 and $NumberOfSectors = 4294967295) Then ; A typical
                        _CheckGPT($hImage)
                ElseIf $FilesystemDescriptor = "05" Or $FilesystemDescriptor = "0F" Then ;Extended partition
                        _CheckExtendedPartition($StartingSector, $hImage)
                ElseIf $FilesystemDescriptor = "07" Then ;Marked as NTFS
                        $Entries &= _GenComboDescription($StartingSector,$NumberOfSectors)
                EndIf
    Next
        If $Entries = "" Then ;Also check if pure partition image (without mbr)
                $NtfsVolumeSize = _TestNTFS($hImage, 0)
                If $NtfsVolumeSize Then $Entries = _GenComboDescription(0,$NtfsVolumeSize)
        EndIf
        _WinAPI_CloseHandle($hImage)
EndFunc   ;==>_CheckMBR

Func _CheckGPT($hImage) ; Assume GPT to be present at sector 1, which is not fool proof
   ;Actually it is. While LBA1 may not be at sector 1 on the disk, it will always be there in an image.
        Local $nbytes,$read,$sector,$GPTSignature,$StartLBA,$Processed=0,$FirstLBA,$LastLBA
        $tBuffer = DllStructCreate("byte[512]")
        $read = _WinAPI_ReadFile($hImage, DllStructGetPtr($tBuffer), 512, $nBytes)              ;read second sector
        If $read = 0 Then Return ""
        $sector = DllStructGetData($tBuffer, 1)
        $GPTSignature = StringMid($sector,3,16)
        If $GPTSignature <> "4546492050415254" Then
                _DisplayInfo("Error: Could not find GPT signature" & @CRLF)
                Return
        EndIf
        $StartLBA = Dec(_SwapEndian(StringMid($sector,147,16)),2)
        $PartitionsInArray = Dec(_SwapEndian(StringMid($sector,163,8)),2)
        $PartitionEntrySize = Dec(_SwapEndian(StringMid($sector,171,8)),2)
        _WinAPI_SetFilePointerEx($hImage, $StartLBA*512, $FILE_BEGIN)
        $SizeNeeded = $PartitionsInArray*$PartitionEntrySize ;Set buffer size -> maximum number of partition entries that ca
        $tBuffer = DllStructCreate("byte[" & $SizeNeeded & "]")
        $read = _WinAPI_ReadFile($hImage, DllStructGetPtr($tBuffer), $SizeNeeded, $nBytes)
        If $read = 0 Then Return ""
        $sector = DllStructGetData($tBuffer, 1)
        Do

                $FirstLBA = Dec(_SwapEndian(StringMid($sector,67+($Processed*2),16)),2)
                $LastLBA = Dec(_SwapEndian(StringMid($sector,83+($Processed*2),16)),2)
                If $FirstLBA = 0 And $LastLBA = 0 Then ExitLoop ; No more entries
                $Processed += $PartitionEntrySize
                If Not _TestNTFS($hImage, $FirstLBA) Then ContinueLoop ;Continue the loop if filesystem not NTFS
                $Entries &= _GenComboDescription($FirstLBA,$LastLBA-$FirstLBA)
        Until $Processed >= $SizeNeeded
EndFunc   ;==>_CheckGPT

Func _CheckExtendedPartition($StartSector, $hImage)     ;Extended partitions can only contain Logical Drives, but can be mor
        Local $nbytes,$read,$sector,$NextEntry=0,$StartingSector,$NumberOfSectors,$PartitionTable,$FilesystemDescriptor
        $tBuffer = DllStructCreate("byte[512]")
        While 1
                _WinAPI_SetFilePointerEx($hImage, ($StartSector + $NextEntry) * 512, $FILE_BEGIN)
                $read = _WinAPI_ReadFile($hImage, DllStructGetPtr($tBuffer), 512, $nBytes)
                If $read = 0 Then Return ""
```

```
                If $read = 0 Then Return
                $sector = DllStructGetData($tBuffer, 1)
                $PartitionTable = StringMid($sector,3+892,64)
                $FilesystemDescriptor = StringMid($PartitionTable,9,2)
                $StartingSector = $StartSector+$NextEntry+Dec(_SwapEndian(StringMid($PartitionTable,17,8)),2)
                $NumberOfSectors = Dec(_SwapEndian(StringMid($PartitionTable,25,8)),2)
                If $FilesystemDescriptor = "07" Then $Entries &= _GenComboDescription($StartingSector,$NumberOfSectors)
                If StringMid($PartitionTable,33) = "00000000000000000000000000000000" Then ExitLoop ; No more entries
                $NextEntry = Dec(_SwapEndian(StringMid($PartitionTable,49,8)),2)
        WEnd
EndFunc   ;==>_CheckExtendedPartition

Func _TestNTFS($hImage, $PartitionStartSector)
        Local $nbytes, $TotalSectors
        If $PartitionStartSector <> 0 Then
                _WinAPI_SetFilePointerEx($hImage, $PartitionStartSector*512, $FILE_BEGIN)
        Else
                _WinAPI_CloseHandle($hImage)
                $hImage = _WinAPI_CreateFile($TargetImageFile,2,2,7)
        EndIf
        $tBuffer = DllStructCreate("byte[512]")
        $read = _WinAPI_ReadFile($hImage, DllStructGetPtr($tBuffer), 512, $nBytes)
        If $read = 0 Then Return ""
        $sector = DllStructGetData($tBuffer, 1)
        $TestSig = StringMid($sector,9,8)
        $TotalSectors = Dec(_SwapEndian(StringMid($sector,83,8)),2)
        If $TestSig = "4E544653" Then Return $TotalSectors              ; Volume is NTFS
        _DisplayInfo("Could not find NTFS on that volume" & @CRLF)              ; Volume is not NTFS
    Return 0
EndFunc   ;==>_TestNTFS   ;==>_TestNTFS

Func _GenComboDescription($StartSector,$SectorNumber)
        Return "Offset = " & $StartSector*512 & ": Volume size = " & Round(($SectorNumber*512)/1024/1024/1024,2) & " GB|"
EndFunc   ;==>_GenComboDescription


Func _GetMountedDrivesInfo()
        GUICtrlSetData($Combo,"","")
        Local $menu = '', $Drive = DriveGetDrive('All')
        If @error Then
                _DisplayInfo("Error - something went wrong in Func _GetPhysicalDriveInfo" & @CRLF)
                Return
        EndIf
        For $i = 1 to $Drive[0]
                $DriveType = DriveGetType($Drive[$i])
                $DriveCapacity = Round(DriveSpaceTotal($Drive[$i]),0)
                If DriveGetFileSystem($Drive[$i]) = 'NTFS' Then
                        $menu &=  StringUpper($Drive[$i]) & "  (" & $DriveType & ")  - " & $DriveCapacity & " MB  - NTFS|"
                EndIf
        Next
        If $menu Then
;               _DisplayInfo("NTFS drives detected" & @CRLF)
                GUICtrlSetData($Combo, $menu, StringMid($menu, 1, StringInStr($menu, "|") -1))
                $IsImage = False
        Else
                _DisplayInfo("No NTFS drives detected" & @CRLF)
        EndIf
EndFunc

Func _HexEncode($bInput)
    Local $tInput = DllStructCreate("byte[" & BinaryLen($bInput) & "]")
    DllStructSetData($tInput, 1, $bInput)
    Local $a_iCall = DllCall("crypt32.dll", "int", "CryptBinaryToString", _
            "ptr", DllStructGetPtr($tInput), _
            "dword", DllStructGetSize($tInput), _
            "dword", 11, _
            "ptr", 0, _
            "dword*", 0)

    If @error Or Not $a_iCall[0] Then
        Return SetError(1, 0, "")
    EndIf
    Local $iSize = $a_iCall[5]
    Local $tOut = DllStructCreate("char[" & $iSize & "]")
    $a_iCall = DllCall("crypt32.dll", "int", "CryptBinaryToString", _
            "ptr", DllStructGetPtr($tInput)
```

```
574             "dword", DllStructGetSize($tInput), _
575             "dword", 11, _
576             "ptr", DllStructGetPtr($tOut), _
577             "dword*", $iSize)
578     If @error Or Not $a_iCall[0] Then
579         Return SetError(2, 0, "")
580     EndIf
581     Return SetError(0, 0, DllStructGetData($tOut, 1))
582 EndFunc
583
584 Func _GetPhysicalDrives($InputDevice)
585         Local $PhysicalDriveString, $hFile0
586         If StringLeft($InputDevice,10) = "GLOBALROOT" Then ; Shadow copies starts at 1 whereas physical drive starts at 0
587                 $i=1
588         Else
589                 $i=0
590         EndIf
591         GUICtrlSetData($Combo,"","")
592         $Entries = ''
593         GUICtrlSetData($ComboPhysicalDrives,"","")
594         $sDrivePath = '\\.\'&$InputDevice
595         ConsoleWrite("$sDrivePath: " & $sDrivePath & @CRLF)
596         Do
597                 $hFile0 = _WinAPI_CreateFile($sDrivePath & $i,2,2,2)
598                 If $hFile0 <> 0 Then
599                         ConsoleWrite("Found: " & $sDrivePath & $i & @CRLF)
600                         _WinAPI_CloseHandle($hFile0)
601                         $PhysicalDriveString &= $sDrivePath&$i&"|"
602                 EndIf
603                 $i+=1
604         Until $hFile0=0
605         GUICtrlSetData($ComboPhysicalDrives, $PhysicalDriveString, StringMid($PhysicalDriveString, 1, StringInStr($PhysicalD
606 EndFunc
607
608 Func _TestPhysicalDrive()
609         $TargetImageFile = GUICtrlRead($ComboPhysicalDrives)
610         If @error then Return
611         _DisplayInfo("Target is " & $TargetImageFile & @CRLF)
612         GUICtrlSetData($Combo,"","")
613         $Entries = ''
614         _CheckMBR()
615         GUICtrlSetData($Combo,$Entries,StringMid($Entries, 1, StringInStr($Entries, "|") -1))
616         If $Entries = "" Then _DisplayInfo("Sorry, no NTFS volume found" & @CRLF)
617         If StringInStr($TargetImageFile,"GLOBALROOT") Then
618                 $IsShadowCopy=True
619                 $IsPhysicalDrive=False
620                 $IsImage=False
621         ElseIf StringInStr($TargetImageFile,"PhysicalDrive") Then
622                 $IsShadowCopy=False
623                 $IsPhysicalDrive=True
624                 $IsImage=False
625         EndIf
    EndFunc
```