GitHub This repository Search or type a command Explore Features Enterprise Blog Sign up Sign in

jschicht / **RawDir** Star 0 Fork 0

branch: master **RawDir / RawDir.au3**

jschicht 2 months ago Version 1.0.0.0

1 contributor

file 1996 lines (1920 sloc) 92.539 kb Open Edit Raw Blame History Delete

```
#RequireAdmin
#Region ;**** Directives created by AutoIt3Wrapper_GUI ****
#AutoIt3Wrapper_Change2CUI=y
#AutoIt3Wrapper_Res_Comment=Raw directory listing
#AutoIt3Wrapper_Res_Description=Low level dir command
#AutoIt3Wrapper_Res_Fileversion=1.0.0.0
#AutoIt3Wrapper_Res_LegalCopyright=Joakim Schicht
#AutoIt3Wrapper_Res_requestedExecutionLevel=asInvoker
#EndRegion ;**** Directives created by AutoIt3Wrapper_GUI ****
#Include <WinAPIEx.au3>
#include <Array.au3>
#Include <String.au3>
#Include <FileConstants.au3>

Global $DirArray,$NeedIndx=0, $ResidentIndx, $AttributesArr[18][4], $DoExtractMeta=False, $TargetFileName, $DATA_Name, $FN_
Global $TargetImageFile, $Entries, $InputFile, $IsShadowCopy=False, $IsPhysicalDrive=False, $IsImage=False, $hDisk, $sBuffe
Global $OutPutPath=@ScriptDir, $InitState = False, $DATA_Clusters, $AttributeOutFileName, $DATA_InitSize, $ImageOffset, $AD
Global $TargetDrive = "", $ALInnerCouner, $MFTSize, $TargetOffset, $SectorsPerCluster,$MFT_Record_Size,$BytesPerCluster,$By
Global $IsolatedAttributeList, $AttribListNonResident=0,$IsCompressed,$IsSparse, $_COMMON_KERNEL32DLL=DllOpen("kernel32.dll
Global $RUN_VCN[1],$RUN_Clusters[1],$MFT_RUN_Clusters[1],$MFT_RUN_VCN[1],$DataQ[1],$AttribX[1],$AttribXType[1],$AttribXCoun
Global $IndxEntryNumberArr[1],$IndxMFTReferenceArr[1],$IndxMFTRefSeqNoArr[1],$IndxIndexFlagsArr[1],$IndxMFTReferenceOfParen
Global $IRArr[12][2],$IndxArr[20][2]
Global $DateTimeFormat = 6 ; YYYY-MM-DD HH:MM:SS:MSMSMS:NSNSNSNS = 2007-08-18 08:15:37:733:1234
Global $tDelta = _WinTime_GetUTCToLocalFileTimeDelta()
Global Const $RecordSignature = '46494C45' ; FILE signature
Global Const $RecordSignatureBad = '44414142' ; BAAD signature
Global Const $STANDARD_INFORMATION = '10000000'
Global Const $ATTRIBUTE_LIST = '20000000'
Global Const $FILE_NAME = '30000000'
Global Const $OBJECT_ID = '40000000'
Global Const $SECURITY_DESCRIPTOR = '50000000'
Global Const $VOLUME_NAME = '60000000'

Global Const $VOLUME_INFORMATION = '70000000'
Global Const $DATA = '80000000'
Global Const $INDEX_ROOT = '90000000'
Global Const $INDEX_ALLOCATION = 'A0000000'
Global Const $BITMAP = 'B0000000'
Global Const $REPARSE_POINT = 'C0000000'
Global Const $EA_INFORMATION = 'D0000000'
Global Const $EA = 'E0000000'
Global Const $PROPERTY_SET = 'F0000000'
Global Const $LOGGED_UTILITY_STREAM = '00010000'
Global Const $ATTRIBUTE_END_MARKER = 'FFFFFFFF'
Global Const $FileInternalInformation = 6
Global Const $OBJ_CASE_INSENSITIVE = 0x00000040
Global Const $FILE_DIRECTORY_FILE = 0x00000002
Global Const $FILE_NON_DIRECTORY_FILE = 0x00000040
Global Const $FILE_RANDOM_ACCESS = 0x00000800
Global Const $tagIOSTATUSBLOCK = "dword Status;ptr Information"
Global Const $tagOBJECTATTRIBUTES = "ulong Length;hwnd RootDirectory;ptr ObjectName;ulong Attributes;ptr SecurityDescriptor
Global Const $tagUNICODESTRING = "ushort Length;ushort MaximumLength;ptr Buffer"
Global Const $tagFILEINTERNALINFORMATION = "int IndexNumber;"
Global $Timerstart = TimerInit()

ConsoleWrite("RawDir v1.0.0.0" & @CRLF)

```

```
If $cmdline[0] <> 2 Then
	ConsoleWrite("Error: Wrong input" & @CRLF)
	_Usage()
	Exit
ElseIf $cmdline[1] <> 1 And $cmdline[1] <> 2 Then
	ConsoleWrite("Error: Wrong param1: " & $cmdline[1] & @CRLF)
	_Usage()
	Exit
ElseIf  DriveGetFileSystem(StringMid($cmdline[2],1,2)&"\") <> "NTFS" Then
	ConsoleWrite("Error: Target drive is not valid or is not a NTFS volume: " & $cmdline[2] & @CRLF)
	Exit
EndIf
$DetailMode = $cmdline[1]
$StartStr = $cmdline[2]
If StringLen($StartStr)=2 Then $StartStr&="\"
$TargetDrive = StringMid($StartStr,1,2)
$ParentDir = _GenDirArray($StartStr)
Global $MftRefArray[$DirArray[0]+1]
_ReadBootSector($TargetDrive)
$BytesPerCluster = $SectorsPerCluster*$BytesPerSector
$MFTEntry = _FindMFT(0)
_DecodeMFTRecord($MFTEntry,0)
_DecodeDataQEntry($DataQ[1])
$MFTSize = $DATA_RealSize
Global $RUN_VCN[1], $RUN_Clusters[1]
_ExtractDataRuns()

$MFT_RUN_VCN = $RUN_VCN
$MFT_RUN_Clusters = $RUN_Clusters

$hDisk = _WinAPI_CreateFile("\\.\" & $TargetDrive,2,2,7)
If $hDisk = 0 Then
	ConsoleWrite("CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
	Exit
EndIf

$NextRef = 5
$MftRefArray[1]=$NextRef
$ResolvedPath = $DirArray[1]
For $i = 2 To $DirArray[0]
	Global $DataQ[1],$AttribX[1],$AttribXType[1],$AttribXCounter[1]
	$NewRecord = _FindFileMFTRecord($NextRef)
	_DecodeMFTRecord($NewRecord,1)
	$NextRef = _ParseIndex($DirArray[$i])
	$MftRefArray[$i]=$NextRef
	If @error Then
		Global $DataQ[1],$AttribX[1],$AttribXType[1],$AttribXCounter[1]
		$NewRecord = _FindFileMFTRecord($MftRefArray[$i-1])
		_DecodeMFTRecord($NewRecord,1)
		$LastCheck = _DisplayList($ResolvedPath)
	ElseIf $i=$DirArray[0] Then
		Global $DataQ[1],$AttribX[1],$AttribXType[1],$AttribXCounter[1]
		$NewRecord = _FindFileMFTRecord($MftRefArray[$i])
		_DecodeMFTRecord($NewRecord,1)
		$LastCheck = _DisplayList($ResolvedPath & "\" & $DirArray[$i])
		If @error Then ; In case last part was a file and not a directory
			Global $DataQ[1],$AttribX[1],$AttribXType[1],$AttribXCounter[1]
			$NewRecord = _FindFileMFTRecord($MftRefArray[$i-1])
			_DecodeMFTRecord($NewRecord,1)
			$LastCheck = _DisplayList($ResolvedPath)
		EndIf
	ElseIf StringIsDigit($NextRef) Then
		$ResolvedPath &= "\" & $DirArray[$i]
		ContinueLoop
	Else
		ConsoleWrite("Error: Something went wrong" & @CRLF)
		ExitLoop
	EndIf
Next
ConsoleWrite(@CRLF)
_End($Timerstart)
Exit

Func _DisplayList($DirListPath)
	If $DetailMode = 1 Then
```

```
        If $AttributesArr[10][2] = "TRUE" Then; $INDEX_ALLOCATION
            ConsoleWrite("Directory listing for: " & $DirListPath & @CRLF & @CRLF)
            For $j = 1 To Ubound($IndxFileNameArr)-1
                ConsoleWrite("Entry number: " & $IndxEntryNumberArr[$j] & @CRLF)
                ConsoleWrite("FileName: " & $IndxFileNameArr[$j] & @CRLF)
                ConsoleWrite("MFT Ref: " & $IndxMFTReferenceArr[$j] & @CRLF)
                ConsoleWrite("MFT Ref SeqNo: " & $IndxMFTRefSeqNoArr[$j] & @CRLF)
                ConsoleWrite("Parent MFT Ref: " & $IndxMFTReferenceOfParentArr[$j] & @CRLF)
                ConsoleWrite("Parent MFT Ref SeqNo: " & $IndxMFTParentRefSeqNoArr[$j] & @CRLF)
                ConsoleWrite("Flags: " & $IndxFileFlagsArr[$j] & @CRLF)
                ConsoleWrite("File Create Time: " & $IndxCTimeArr[$j] & @CRLF)
                ConsoleWrite("File Modified Time: " & $IndxATimeArr[$j] & @CRLF)
                ConsoleWrite("MFT Entry modified Time: " & $IndxMTimeArr[$j] & @CRLF)
                ConsoleWrite("File Last Access Time: " & $IndxRTimeArr[$j] & @CRLF)
                ConsoleWrite("Allocated Size: " & $IndxAllocSizeArr[$j] & @CRLF)
                ConsoleWrite("Real Size: " & $IndxRealSizeArr[$j] & @CRLF)
                ConsoleWrite("NameSpace: " & $IndxNameSpaceArr[$j] & @CRLF)
                ConsoleWrite("IndexFlags: " & $IndxIndexFlagsArr[$j] & @CRLF)
                ConsoleWrite("SubNodeVCN: " & $IndxSubNodeVCNArr[$j] & @CRLF)
                ConsoleWrite(@CRLF)
            Next
        ElseIf $AttributesArr[9][2] = "TRUE" Then ;And $ResidentIndx Then ; $INDEX_ROOT
            ConsoleWrite("Directory listing for: " & $DirListPath & @CRLF & @CRLF)
            For $j = 1 To Ubound($IndxFileNameArr)-1
                ConsoleWrite("Entry number: " & $IndxEntryNumberArr[$j] & @CRLF)
                ConsoleWrite("FileName: " & $IndxFileNameArr[$j] & @CRLF)
                ConsoleWrite("MFT Ref: " & $IndxMFTReferenceArr[$j] & @CRLF)
                ConsoleWrite("MFT Ref SeqNo: " & $IndxMFTRefSeqNoArr[$j] & @CRLF)
                ConsoleWrite("Parent MFT Ref: " & $IndxMFTReferenceOfParentArr[$j] & @CRLF)
                ConsoleWrite("Parent MFT Ref SeqNo: " & $IndxMFTParentRefSeqNoArr[$j] & @CRLF)
                ConsoleWrite("Flags: " & $IndxFileFlagsArr[$j] & @CRLF)
                ConsoleWrite("File Create Time: " & $IndxCTimeArr[$j] & @CRLF)
                ConsoleWrite("File Modified Time: " & $IndxATimeArr[$j] & @CRLF)
                ConsoleWrite("MFT Entry modified Time: " & $IndxMTimeArr[$j] & @CRLF)
                ConsoleWrite("File Last Access Time: " & $IndxRTimeArr[$j] & @CRLF)
                ConsoleWrite("Allocated Size: " & $IndxAllocSizeArr[$j] & @CRLF)
                ConsoleWrite("Real Size: " & $IndxRealSizeArr[$j] & @CRLF)
                ConsoleWrite("NameSpace: " & $IndxNameSpaceArr[$j] & @CRLF)
                ConsoleWrite("IndexFlags: " & $IndxIndexFlagsArr[$j] & @CRLF)
                ConsoleWrite("SubNodeVCN: " & $IndxSubNodeVCNArr[$j] & @CRLF)
                ConsoleWrite(@CRLF)
            Next
        Else
;           ConsoleWrite("Error: There was no index found for the parent folder." & @CRLF)
            Return SetError(1,0,0)
        EndIf
    ElseIf $DetailMode = 2 Then
        If $AttributesArr[10][2] = "TRUE" Then; $INDEX_ALLOCATION
            $HighestVal = _ArrayMax($IndxRealSizeArr,1)
            If @error then
                ConsoleWrite("Error: Unexpected error when resolving higest value in array: " & @error & @C
                Exit
            EndIf
            $HighestValLength = StringLen($HighestVal)
            $RealSizeStr = "RealSize"
            If StringLen($RealSizeStr) < $HighestValLength Then $RealSizeStr = _AlignString($RealSizeStr,$Highe
            ConsoleWrite("Directory listing for: " & $DirListPath & @CRLF & @CRLF)
            ConsoleWrite("           File Modified Time|   Type|"&$RealSizeStr&"| FileName" & @CRLF)
            For $j = 1 To Ubound($IndxFileNameArr)-1
                If StringInStr($IndxFileFlagsArr[$j],"directory") Then
                    $FileType = "<DIR>"
                Else
                    $FileType = "     "
                EndIf
                $AlignedSizeVal = _AlignString($IndxRealSizeArr[$j],$HighestValLength)
                $TextOut = $IndxATimeArr[$j] & " | " & $FileType & " | " & $AlignedSizeVal & " | " & $IndxF
                ConsoleWrite($TextOut & @CRLF)
            Next
        ElseIf $AttributesArr[9][2] = "TRUE" Then ;And $ResidentIndx Then ; $INDEX_ROOT
            $HighestVal = _ArrayMax($IndxRealSizeArr,1)
            If @error then
                ConsoleWrite("Error: Unexpected error when resolving higest value in array: " & @error & @C
                Exit
```

```
			EndIf
			$HighestValLength = StringLen($HighestVal)
			$RealSizeStr = "RealSize"
			If StringLen($RealSizeStr) < $HighestValLength Then $RealSizeStr = _AlignString($RealSizeStr,$Highe
			ConsoleWrite("Directory listing for: " & $DirListPath & @CRLF & @CRLF)
			ConsoleWrite("           File Modified Time|   Type|"&$RealSizeStr&"| FileName" & @CRLF)
			For $j = 1 To Ubound($IndxFileNameArr)-1
				If StringInStr($IndxFileFlagsArr[$j],"directory") Then
					$FileType = "<DIR>"
				Else
					$FileType = "     "
				EndIf
				$AlignedSizeVal = _AlignString($IndxRealSizeArr[$j],$HighestValLength)
				$TextOut = $IndxATimeArr[$j] & " | " & $FileType & " | " & $AlignedSizeVal & " | " & $IndxF
				ConsoleWrite($TextOut & @CRLF)
			Next
		Else
;			ConsoleWrite("Error: There was no index found for the parent folder." & @CRLF)
			Return SetError(1,0,0)
		EndIf
	EndIf
EndFunc

Func _ParseIndex($TestName)
	If $AttributesArr[10][2] = "TRUE" Then; $INDEX_ALLOCATION
		For $j = 1 To Ubound($IndxFileNameArr)-1
			If $IndxFileNameArr[$j] = $TestName Then
				Return $IndxMFTReferenceArr[$j]
			Else
;				Return SetError(1,0,0)
			EndIf
		Next
	ElseIf $AttributesArr[9][2] = "TRUE" Then ;And $ResidentIndx Then ; $INDEX_ROOT
		For $j = 1 To Ubound($IndxFileNameArr)-1
			If $IndxFileNameArr[$j] = $TestName Then
				Return $IndxMFTReferenceArr[$j]
			Else
;				Return SetError(1,0,0)
			EndIf
		Next
	Else
;		ConsoleWrite("Error: No index found for: " & $TestName & @CRLF)
		Return SetError(1,0,0)
	EndIf
EndFunc

Func _GenDirArray($InPath)
	Local $Reconstruct
	Global $DirArray = StringSplit($InPath,"\")
	For $i = 1 To $DirArray[0]-1
		$Reconstruct &= $DirArray[$i]&"\"
	Next
	$Reconstruct = StringTrimRight($Reconstruct,1)
	Return $Reconstruct
EndFunc

Func _ExtractSingleFile($MFTReferenceNumber)
	Global $DataQ[1],$AttribX[1],$AttribXType[1],$AttribXCounter[1]				;clear array
	$MFTRecord = _FindFileMFTRecord($MFTReferenceNumber)
	If $MFTRecord = "" Then
		ConsoleWrite("Target " & $MFTReferenceNumber & " not found" & @CRLF)
		;_DisplayInfo("Target " & $MFTReferenceNumber & " not found" & @CRLF)
		Return SetError(1,0,0)
	ElseIf StringMid($MFTRecord,3,8) <> $RecordSignature AND StringMid($MFTRecord,3,8) <> $RecordSignatureBad Then
		ConsoleWrite("Found record is not valid:" & @CRLF)
		;_DisplayInfo("Found record is not valid:" & @CRLF)
		ConsoleWrite(_HexEncode($MFTRecord) & @crlf)
		Return SetError(1,0,0)
	EndIf
	_DecodeMFTRecord($MFTRecord,1)
	Return
EndFunc

Func _DecodeAttrList($TargetFile, $AttrList)
```

```
        Local $offset, $length, $nBytes, $hFile, $LocalAttribID, $LocalName, $ALRecordLength, $ALNameLength, $ALNameOffset
        If StringMid($AttrList, 17, 2) = "00" Then              ;attribute list is in $AttrList
                $offset = Dec(_SwapEndian(StringMid($AttrList, 41, 4)))
                $List = StringMid($AttrList, $offset*2+1)
;               $IsolatedAttributeList = $list
        Else                    ;attribute list is found from data run in $AttrList
                $size = Dec(_SwapEndian(StringMid($AttrList, $offset*2 + 97, 16)))
                $offset = ($offset + Dec(_SwapEndian(StringMid($AttrList, $offset*2 + 65, 4))))*2
                $DataRun = StringMid($AttrList, $offset+1, StringLen($AttrList)-$offset)
;               ConsoleWrite("Attribute_List DataRun is " & $DataRun & @CRLF)
                Global $RUN_VCN[1], $RUN_Clusters[1]
                _ExtractDataRuns()
                $tBuffer = DllStructCreate("byte[" & $BytesPerCluster & "]")
                $hFile = _WinAPI_CreateFile("\\.\" & $TargetDrive, 2, 6, 6)
                If $hFile = 0 Then
                        ConsoleWrite("Error in function CreateFile when trying to locate Attribute List." & @CRLF)
                        ;_DisplayInfo("Error in function CreateFile when trying to locate Attribute List." & @CRLF)
                        _WinAPI_CloseHandle($hFile)
                        Return SetError(1,0,0)
                EndIf
                $List = ""
                For $r = 1 To Ubound($RUN_VCN)-1
                        _WinAPI_SetFilePointerEx($hFile, $RUN_VCN[$r]*$BytesPerCluster, $FILE_BEGIN)
                        For $i = 1 To $RUN_Clusters[$r]
                                _WinAPI_ReadFile($hFile, DllStructGetPtr($tBuffer), $BytesPerCluster, $nBytes)
                                $List &= StringTrimLeft(DllStructGetData($tBuffer, 1),2)
                        Next
                Next
;               _DebugOut("***AttrList New:",$List)
                _WinAPI_CloseHandle($hFile)
                $List = StringMid($List, 1, $size*2)
        EndIf
        $IsolatedAttributeList = $list
        $offset=0
        $str=""
        While StringLen($list) > $offset*2
                $type=StringMid($List, ($offset*2)+1, 8)
                $ALRecordLength = Dec(_SwapEndian(StringMid($List, $offset*2 + 9, 4)))
                $ALNameLength = Dec(_SwapEndian(StringMid($List, $offset*2 + 13, 2)))
                $ALNameOffset = Dec(_SwapEndian(StringMid($List, $offset*2 + 15, 2)))
                $TestVCN = Dec(_SwapEndian(StringMid($List, $offset*2 + 17, 16)))
                $ref=Dec(_SwapEndian(StringMid($List, $offset*2 + 33, 8)))
                $LocalAttribID = "0x" & StringMid($List, $offset*2 + 49, 2) & StringMid($List, $offset*2 + 51, 2)
                If $ALNameLength > 0 Then
                        $LocalName = StringMid($List, $offset*2 + 53, $ALNameLength*2*2)
                        $LocalName = _UnicodeHexToStr($LocalName)
                Else
                        $LocalName = ""
                EndIf
                If $ref <> $TargetFile Then             ;new attribute
                        If Not StringInStr($str, $ref) Then $str &= $ref & "-"
                EndIf
                If $type=$DATA Then
                        $DataInAttrlist=1
                        $IsolatedData=StringMid($List, ($offset*2)+1, $ALRecordLength*2)
                        If $TestVCN=0 Then $DataIsResident=1
                EndIf
                $offset += Dec(_SwapEndian(StringMid($List, $offset*2 + 9, 4)))
        WEnd
        If $str = "" Then
                ConsoleWrite("No extra MFT records found" & @CRLF)
                ;_DisplayInfo("No extra MFT records found" & @CRLF)
        Else
                $AttrQ = StringSplit(StringTrimRight($str,1), "-")
;               ConsoleWrite("Decode of $ATTRIBUTE_LIST reveiled extra MFT Records to be examined = " & _ArrayToString($Att
        EndIf
EndFunc

Func _StripMftRecord($MFTEntry)
        $UpdSeqArrOffset = Dec(_SwapEndian(StringMid($MFTEntry,11,4)))
        $UpdSeqArrSize = Dec(_SwapEndian(StringMid($MFTEntry,15,4)))
        $UpdSeqArr = StringMid($MFTEntry,3+($UpdSeqArrOffset*2),$UpdSeqArrSize*2*2)
        $UpdSeqArrPart0 = StringMid($UpdSeqArr,1,4)
        $UpdSeqArrPart1 = StringMid($UpdSeqArr,5,4)
        $UpdSeqArrPart2 = StringMid($UpdSeqArr,9,4)
        $RecordEnd1 = StringMid($MFTEntry,1023,4)
        $RecordEnd2 = StringMid($MFTEntry,2047,4)
        If $UpdSeqArrPart0 <> $RecordEnd1 OR $UpdSeqArrPart0 <> $RecordEnd2 Then
                ConsoleWrite("Error the $MFT record is corrupt" & @CRLF)
                ;_DisplayInfo("Error the $MFT record is corrupt" & @CRLF)
                Return SetError(1,0,0)
        Else
                $MFTEntry = StringMid($MFTEntry,1,1022) & $UpdSeqArrPart1 & StringMid($MFTEntry,1027,1020) & $UpdSeqArrPart
        EndIf
        $RecordSize = Dec(_SwapEndian(StringMid($MFTEntry,51,8)),2)
        $HeaderSize = Dec(_SwapEndian(StringMid($MFTEntry,43,4)),2)
        $MFTEntry = StringMid($MFTEntry,$HeaderSize*2+3,($RecordSize-$HeaderSize-8)*2)        ;strip "0x..." and "FFFFFFFF.
        Return $MFTEntry
EndFunc

Func _DecodeDataQEntry($attr)           ;processes data attribute
   $NonResidentFlag = StringMid($attr,17,2)
   $NameLength = Dec(StringMid($attr,19,2))
   $NameOffset = Dec(_SwapEndian(StringMid($attr,21,4)))
   If $NameLength > 0 Then              ;must be ADS
          $ADS_Name = _UnicodeHexToStr(StringMid($attr,$NameOffset*2 + 1,$NameLength*4))
          $ADS_Name = $FN_FileName & "[ADS_" & $ADS_Name & "]"
   Else
          $ADS_Name = $FN_FileName              ;need to preserve $FN_FileName
   EndIf
   $Flags = StringMid($attr,25,4)
   If BitAND($Flags,"0100") Then $IsCompressed = 1
   If BitAND($Flags,"0080") Then $IsSparse = 1
   If $NonResidentFlag = '01' Then
          $DATA_Clusters = Dec(_SwapEndian(StringMid($attr,49,16)),2) - Dec(_SwapEndian(StringMid($attr,33,16)),2) + 1

          $DATA_RealSize = Dec(_SwapEndian(StringMid($attr,97,16)),2)
          $DATA_InitSize = Dec(_SwapEndian(StringMid($attr,113,16)),2)
          $Offset = Dec(_SwapEndian(StringMid($attr,65,4)))
          $DataRun = StringMid($attr,$Offset*2+1,(StringLen($attr)-$Offset)*2)
   ElseIf $NonResidentFlag = '00' Then
          $DATA_LengthOfAttribute = Dec(_SwapEndian(StringMid($attr,33,8)),2)
          $Offset = Dec(_SwapEndian(StringMid($attr,41,4)))
          $DataRun = StringMid($attr,$Offset*2+1,$DATA_LengthOfAttribute*2)
   EndIf
EndFunc

Func _DecodeMFTRecord($MFTEntry,$MFTMode)
Global $IndxEntryNumberArr[1],$IndxMFTReferenceArr[1],$IndxIndexFlagsArr[1],$IndxMFTReferenceOfParentArr[1],$IndxCTimeArr[1
Local $MFTEntryOrig,$FN_Number,$DATA_Number,$SI_Number,$ATTRIBLIST_Number,$OBJID_Number,$SECURITY_Number,$VOLNAME_Number,$V
Local $INDEX_ROOT_ON="FALSE",$INDEX_ALLOCATION_ON="FALSE"
Global $IRArr[12][2],$IndxArr[20][2]
_SetArrays()
$HEADER_RecordRealSize = ""
$HEADER_MFTREcordNumber = ""
$UpdSeqArrOffset = Dec(_SwapEndian(StringMid($MFTEntry,11,4)))
$UpdSeqArrSize = Dec(_SwapEndian(StringMid($MFTEntry,15,4)))
$UpdSeqArr = StringMid($MFTEntry,3+($UpdSeqArrOffset*2),$UpdSeqArrSize*2*2)
$UpdSeqArrPart0 = StringMid($UpdSeqArr,1,4)
$UpdSeqArrPart1 = StringMid($UpdSeqArr,5,4)
$UpdSeqArrPart2 = StringMid($UpdSeqArr,9,4)
$RecordEnd1 = StringMid($MFTEntry,1023,4)
$RecordEnd2 = StringMid($MFTEntry,2047,4)
If $UpdSeqArrPart0 <> $RecordEnd1 OR $UpdSeqArrPart0 <> $RecordEnd2 Then
        ConsoleWrite("Error: the $MFT record is corrupt" & @CRLF)
        ;_DisplayInfo("Error: the $MFT record is corrupt" & @CRLF)
        Return SetError(1,0,0)
 Else
        $MFTEntry = StringMid($MFTEntry,1,1022) & $UpdSeqArrPart1 & StringMid($MFTEntry,1027,1020) & $UpdSeqArrPart2
EndIf
$HEADER_RecordRealSize = Dec(_SwapEndian(StringMid($MFTEntry,51,8)),2)
If $UpdSeqArrOffset = 48 Then
        $HEADER_MFTREcordNumber = Dec(_SwapEndian(StringMid($MFTEntry,91,8)),2)
Else
        $HEADER_MFTREcordNumber = "NT style"
EndIf
$AttributeOffset = (Dec(StringMid($MFTEntry,43,2))*2)+3

While 1
```

```
while 1
        $AttributeType = StringMid($MFTEntry,$AttributeOffset,8)
        $AttributeSize = StringMid($MFTEntry,$AttributeOffset+8,8)
        $AttributeSize = Dec(_SwapEndian($AttributeSize),2)
        Select
                Case $AttributeType = $STANDARD_INFORMATION
;                       $STANDARD_INFORMATION_ON = "TRUE"
                        $SI_Number += 1

                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $SI_Number)
                        EndIf
                Case $AttributeType = $ATTRIBUTE_LIST
;                       $ATTRIBUTE_LIST_ON = "TRUE"
                        $ATTRIBLIST_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $ATTRIBLIST_Number)
                        EndIf
                        $MFTEntryOrig = $MFTEntry
                        $AttrList = StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2)
                        _DecodeAttrList($HEADER_MFTRecordNumber, $AttrList)             ;produces $AttrQ - extra record lis
                        $str = ""
                        For $i = 1 To $AttrQ[0]
                                $record = _FindFileMFTRecord($AttrQ[$i])
                                $str &= _StripMftRecord($record)                ;no header or end marker
                        Next
                        $str &= "FFFFFFFF"              ;add end marker
                        $MFTEntry = StringMid($MFTEntry,1,($HEADER_RecordRealSize-8)*2+2) & $str       ;strip "FFFFFFFF..."
                Case $AttributeType = $FILE_NAME
;                       $FILE_NAME_ON = "TRUE"
                        $FN_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $FN_Number)
                        EndIf
                        $attr = StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2)
                        $NameSpace = StringMid($attr,179,2)
                        Select
                                Case $NameSpace = "00"  ;POSIX
                                        $NameQ[2] = $attr
                                Case $NameSpace = "01"  ;WIN32
                                        $NameQ[4] = $attr
                                Case $NameSpace = "02"  ;DOS
                                        $NameQ[1] = $attr
                                Case $NameSpace = "03"  ;DOS+WIN32
                                        $NameQ[3] = $attr
                        EndSelect
                Case $AttributeType = $OBJECT_ID
;                       $OBJECT_ID_ON = "TRUE"
                        $OBJID_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $OBJID_Number)

                        EndIf
                Case $AttributeType = $SECURITY_DESCRIPTOR
;                       $SECURITY_DESCRIPTOR_ON = "TRUE"
                        $SECURITY_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $SECURITY_Number)
                        EndIf
                Case $AttributeType = $VOLUME_NAME
;                       $VOLUME_NAME_ON = "TRUE"
                        $VOLNAME_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $VOLNAME_Number)
                                _ArrayAdd($AttribXCounter, $VOLNAME_Number)
                        EndIf
                Case $AttributeType = $VOLUME_INFORMATION
;                       $VOLUME_INFORMATION_ON = "TRUE"
                        $VOLINFO_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $VOLINFO_Number)
                        EndIf
                Case $AttributeType = $DATA
;                       $DATA_ON = "TRUE"
                        $DATA_Number += 1
                        _ArrayAdd($DataQ, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                Case $AttributeType = $INDEX_ROOT
                        $INDEX_ROOT_ON = "TRUE"
                        $INDEXROOT_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $INDEXROOT_Number)
                        EndIf
                        ReDim $IRArr[12][$INDEXROOT_Number+1]
                        $CoreIndexRoot = _GetAttributeEntry(StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                        $CoreIndexRootChunk = $CoreIndexRoot[0]
                        $CoreIndexRootName = $CoreIndexRoot[1]
                        If $CoreIndexRootName = "$I30" Then _Get_IndexRoot($CoreIndexRootChunk,$INDEXROOT_Number,$CoreIndex
                Case $AttributeType = $INDEX_ALLOCATION
                        $INDEX_ALLOCATION_ON = "TRUE"
                        $INDEXALLOC_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $INDEXALLOC_Number)
                        EndIf

                        $CoreIndexAllocation = _GetAttributeEntry(StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                        $CoreIndexAllocationChunk = $CoreIndexAllocation[0]
                        $CoreIndexAllocationName = $CoreIndexAllocation[1]
;                       _Arrayadd($HexDumpIndxRecord,$CoreIndexAllocationChunk)
                        If $CoreIndexAllocationName = "$I30" Then _Get_IndexAllocation($CoreIndexAllocationChunk,$INDEXALLO
                Case $AttributeType = $BITMAP
;                       $BITMAP_ON = "TRUE"
                        $BITMAP_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $BITMAP_Number)
                        EndIf
                Case $AttributeType = $REPARSE_POINT
;                       $REPARSE_POINT_ON = "TRUE"
                        $REPARSEPOINT_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $REPARSEPOINT_Number)
                        EndIf
                Case $AttributeType = $EA_INFORMATION
;                       $EA_INFORMATION_ON = "TRUE"
                        $EAINFO_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $EAINFO_Number)
                        EndIf
                Case $AttributeType = $EA
;                       $EA_ON = "TRUE"
                        $EA_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $EA_Number)
                        EndIf
                Case $AttributeType = $PROPERTY_SET
;                       $PROPERTY_SET_ON = "TRUE"
                        $PROPERTYSET_Number += 1
                        $PROPERTYSET_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))
                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $PROPERTYSET_Number)
                        EndIf
                Case $AttributeType = $LOGGED_UTILITY_STREAM
;                       $LOGGED_UTILITY_STREAM_ON = "TRUE"
                        $LOGGEDUTILSTREAM_Number += 1
                        If $MFTMode = 1 Then
                                _ArrayAdd($AttribX, StringMid($MFTEntry,$AttributeOffset,$AttributeSize*2))

                                _ArrayAdd($AttribXType, $AttributeType)
                                _ArrayAdd($AttribXCounter, $LOGGEDUTILSTREAM_Number)
                        EndIf
                Case $AttributeType = $ATTRIBUTE_END_MARKER
                        ExitLoop
        EndSelect
        $AttributeOffset += $AttributeSize*2
WEnd
$AttributesArr[9][2] = $INDEX_ROOT_ON
$AttributesArr[10][2] = $INDEX_ALLOCATION_ON
EndFunc

Func _ExtractDataRuns()
        $r=UBound($RUN_Clusters)
        $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
                        $RunListVCN = 0                 ;$RUN_VCN[$r-1]         ;0
                EndIf
                If (($RunListVCN=0) And ($RunListClusters>16) And (Mod($RunListClusters,16)>0)) Then
                 ;may be sparse section at end of Compression Signature
                        _ArrayAdd($RUN_Clusters,Mod($RunListClusters,16))
                        _ArrayAdd($RUN_VCN,$RunListVCN)
                        $RunListClusters -= Mod($RunListClusters,16)
                        $r += 1
                ElseIf (($RunListClusters>16) And (Mod($RunListClusters,16)>0)) Then
                 ;may be compressed data section at start of Compression Signature
                        _ArrayAdd($RUN_Clusters,$RunListClusters-Mod($RunListClusters,16))
                        _ArrayAdd($RUN_VCN,$RunListVCN)
                        $RunListVCN += $RUN_Clusters[$r]
                        $RunListClusters = Mod($RunListClusters,16)
                        $r += 1
                EndIf
          ;just normal or sparse data
                _ArrayAdd($RUN_Clusters,$RunListClusters)

                _ArrayAdd($RUN_VCN,$RunListVCN)
                $r += 1
                $i += $RunListVCNLength*2
        Until $i > StringLen($DataRun)
EndFunc

Func _FindFileMFTRecord($TargetFile)
        Local $nBytes, $TmpOffset, $Counter, $Counter2, $RecordJumper, $TargetFileDec, $RecordsTooMuch, $RetVal[2]
        $tBuffer = DllStructCreate("byte[" & $MFT_Record_Size & "]")
        $hFile = _WinAPI_CreateFile("\\.\" & $TargetDrive, 2, 6, 6)
        If $hFile = 0 Then
                ConsoleWrite("Error in function CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
                ; DisplayInfo("Error in function CreateFile: " & _WinAPI_GetLastErrorMessage() & @CRLF)
```

```
			_WinAPI_CloseHandle($hFile)
			Return SetError(1,0,0)
		EndIf
		$TargetFile = _DecToLittleEndian($TargetFile)
		$TargetFileDec = Dec(_SwapEndian($TargetFile),2)
		For $i = 1 To UBound($MFT_RUN_Clusters)-1
			$CurrentClusters = $MFT_RUN_Clusters[$i]
			$RecordsInCurrentRun = ($CurrentClusters*$SectorsPerCluster)/2
			$Counter+=$RecordsInCurrentRun
			If $Counter>$TargetFileDec Then
				ExitLoop
			EndIf
		Next
		$TryAt = $Counter-$RecordsInCurrentRun
		$TryAtArrIndex = $i
		$RecordsPerCluster = $SectorsPerCluster/2
		Do
			$RecordJumper+=$RecordsPerCluster
			$Counter2+=1
			$Final = $TryAt+$RecordJumper
		Until $Final>=$TargetFileDec
		$RecordsTooMuch = $Final-$TargetFileDec
		_WinAPI_SetFilePointerEx($hFile, $ImageOffset+$MFT_RUN_VCN[$i]*$BytesPerCluster+($Counter2*$BytesPerCluster)-($Reco
		_WinAPI_ReadFile($hFile, DllStructGetPtr($tBuffer), $MFT_Record_Size, $nBytes)
		$record = DllStructGetData($tBuffer, 1)
		If StringMid($record,91,8) = $TargetFile Then
			$TmpOffset = DllCall('kernel32.dll', 'int', 'SetFilePointerEx', 'ptr', $hFile, 'int64', 0, 'int64*', 0, 'dw
;			ConsoleWrite("Record number: " & Dec(_SwapEndian($TargetFile),2) & " found at disk offset: " & $TmpOffset[3
			;_DisplayInfo("Record number: " & Dec(_SwapEndian($TargetFile),2) & " found at disk offset: " & $TmpOffset[
			_WinAPI_CloseHandle($hFile)
;			$RetVal[0] = $TmpOffset[3]-1024
;			$RetVal[1] = $record
;			Return $RetVal
			Return $record
		Else
			_WinAPI_CloseHandle($hFile)
			Return ""

		EndIf
EndFunc

Func _FindMFT($TargetFile)
		Local $nBytes;, $MFT_Record_Size=1024
		$tBuffer = DllStructCreate("byte[" & $MFT_Record_Size & "]")
		$hFile = _WinAPI_CreateFile("\\.\" & $TargetDrive, 2, 2, 7)
		If $hFile = 0 Then
			ConsoleWrite("Error in function CreateFile when trying to locate MFT: " & _WinAPI_GetLastErrorMessage() & @
			;_DisplayInfo("Error in function CreateFile when trying to locate MFT: " & _WinAPI_GetLastErrorMessage() & (
			Return SetError(1,0,0)
		EndIf
;		ConsoleWrite("$MFT_Offset: " & $MFT_Offset & @CRLF)
		_WinAPI_SetFilePointerEx($hFile, $ImageOffset+$MFT_Offset, $FILE_BEGIN)
		_WinAPI_ReadFile($hFile, DllStructGetPtr($tBuffer), $MFT_Record_Size, $nBytes)
		_WinAPI_CloseHandle($hFile)
		$record = DllStructGetData($tBuffer, 1)
		If NOT StringMid($record,1,8) = '46494C45' Then
			ConsoleWrite("MFT record signature not found. "& @crlf)
			;_DisplayInfo("MFT record signature not found. "& @crlf)
			Return ""
		EndIf
		If StringMid($record,47,4) = "0100" AND Dec(_SwapEndian(StringMid($record,91,8))) = $TargetFile Then
;			ConsoleWrite("MFT record found" & @CRLF)
			Return $record		;returns record for MFT
		EndIf
		ConsoleWrite("MFT record not found" & @CRLF)
		;_DisplayInfo("MFT record not found" & @CRLF)
		Return ""
EndFunc

Func _DecToLittleEndian($DecimalInput)
		Return _SwapEndian(Hex($DecimalInput,8))
EndFunc

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

```
EndFunc

Func _UnicodeHexToStr($FileName)
        $str = ""
        For $i = 1 To StringLen($FileName) Step 4
                $str &= ChrW(Dec(_SwapEndian(StringMid($FileName, $i, 4))))
        Next
        Return $str
EndFunc

Func _DebugOut($text, $var)
        ConsoleWrite("Debug output for " & $text & @CRLF)
        For $i=1 To StringLen($var) Step 32
                $str=""
                For $n=0 To 15
                        $str &= StringMid($var, $i+$n*2, 2) & " "
                        if $n=7 then $str &= "- "
                Next
                ConsoleWrite($str & @CRLF)
        Next
EndFunc

Func _ReadBootSector($TargetDrive)
        Local $nbytes
        $tBuffer=DllStructCreate("byte[512]")
        $hFile = _WinAPI_CreateFile("\\.\" & $TargetDrive,2,2,7)
        If $hFile = 0 then
                ConsoleWrite("Error in function CreateFile: " & _WinAPI_GetLastErrorMessage() & " for: " & "\\.\" & $Target
                ;_DisplayInfo("Error in function CreateFile: " & _WinAPI_GetLastErrorMessage() & " for: " & "\\.\" & $Targe
                Return SetError(1,0,0)
        EndIf
        _WinAPI_SetFilePointerEx($hFile, $ImageOffset, $FILE_BEGIN)
        $read = _WinAPI_ReadFile($hFile, DllStructGetPtr($tBuffer), 512, $nBytes)
        If $read = 0 then
                ConsoleWrite("Error in function ReadFile: " & _WinAPI_GetLastErrorMessage() & " for: " & "\\.\" & $TargetDr
                ;_DisplayInfo("Error in function ReadFile: " & _WinAPI_GetLastErrorMessage() & " for: " & "\\.\" & $TargetD
                Return
        EndIf
        _WinAPI_CloseHandle($hFile)
   ; Good starting point from KaFu & trancexx at the AutoIt forum
        $tBootSectorSections = DllStructCreate("align 1;" & _
                                                                "byte Jump[3];" & _
                                                                "char SystemName[8];" & _
                                                                "ushort BytesPerSector;" & _
                                                                "ubyte SectorsPerCluster;" & _
                                                                "ushort ReservedSectors;" & _
                                                                "ubyte[3];" & _
                                                                "ushort;" & _
                                                                "ubyte MediaDescriptor;" & _
                                                                "ushort;" & _
                                                                "ushort SectorsPerTrack;" & _
                                                                "ushort NumberOfHeads;" & _
                                                                "dword HiddenSectors;" & _
                                                                "dword;" & _
                                                                "dword;" & _
                                                                "int64 TotalSectors;" & _
                                                                "int64 LogicalClusterNumberforthefileMFT;" & _
                                                                "int64 LogicalClusterNumberforthefileMFTMirr;" & _
                                                                "dword ClustersPerFileRecordSegment;" & _
                                                                "dword ClustersPerIndexBlock;" & _
                                                                "int64 NTFSVolumeSerialNumber;" & _
                                                                "dword Checksum", DllStructGetPtr($tBuffer))


        $BytesPerSector = DllStructGetData($tBootSectorSections, "BytesPerSector")
        $SectorsPerCluster = DllStructGetData($tBootSectorSections, "SectorsPerCluster")
        $BytesPerCluster = $BytesPerSector * $SectorsPerCluster
        $ClustersPerFileRecordSegment = DllStructGetData($tBootSectorSections, "ClustersPerFileRecordSegment")
        $LogicalClusterNumberforthefileMFT = DllStructGetData($tBootSectorSections, "LogicalClusterNumberforthefileMFT")
        $MFT_Offset = $BytesPerCluster * $LogicalClusterNumberforthefileMFT
        If $ClustersPerFileRecordSegment > 127 Then
                $MFT_Record_Size = 2 ^ (256 - $ClustersPerFileRecordSegment)
        Else
                $MFT_Record_Size = $BytesPerCluster * $ClustersPerFileRecordSegment
        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), _
            "dword", DllStructGetSize($tInput), _
            "dword", 11, _
            "ptr", DllStructGetPtr($tOut), _
            "dword*", $iSize)

    If @error Or Not $a_iCall[0] Then
        Return SetError(2, 0, "")
    EndIf

    Return SetError(0, 0, DllStructGetData($tOut, 1))

EndFunc  ;==>_HexEncode

Func _File_Attributes($FAInput)
        Local $FAOutput = ""
        If BitAND($FAInput, 0x0001) Then $FAOutput &= 'read_only+'
        If BitAND($FAInput, 0x0002) Then $FAOutput &= 'hidden+'
        If BitAND($FAInput, 0x0004) Then $FAOutput &= 'system+'

        If BitAND($FAInput, 0x0010) Then $FAOutput &= 'directory+'
        If BitAND($FAInput, 0x0020) Then $FAOutput &= 'archive+'
        If BitAND($FAInput, 0x0040) Then $FAOutput &= 'device+'
        If BitAND($FAInput, 0x0080) Then $FAOutput &= 'normal+'
        If BitAND($FAInput, 0x0100) Then $FAOutput &= 'temporary+'
        If BitAND($FAInput, 0x0200) Then $FAOutput &= 'sparse_file+'
        If BitAND($FAInput, 0x0400) Then $FAOutput &= 'reparse_point+'
        If BitAND($FAInput, 0x0800) Then $FAOutput &= 'compressed+'
        If BitAND($FAInput, 0x1000) Then $FAOutput &= 'offline+'
        If BitAND($FAInput, 0x2000) Then $FAOutput &= 'not_indexed+'
        If BitAND($FAInput, 0x4000) Then $FAOutput &= 'encrypted+'
        If BitAND($FAInput, 0x8000) Then $FAOutput &= 'integrity_stream+'
        If BitAND($FAInput, 0x10000) Then $FAOutput &= 'virtual+'
        If BitAND($FAInput, 0x20000) Then $FAOutput &= 'no_scrub_data+'
        If BitAND($FAInput, 0x10000000) Then $FAOutput &= 'directory+'
        If BitAND($FAInput, 0x20000000) Then $FAOutput &= 'index_view+'
        $FAOutput = StringTrimRight($FAOutput, 1)
        Return $FAOutput
EndFunc

Func _End($begin)
        Local $timerdiff = TimerDiff($begin)
        $timerdiff = Round(($timerdiff / 1000), 2)
        ConsoleWrite("Job took " & $timerdiff & " seconds" & @CRLF)
        ;_DisplayInfo("Job took " & $timerdiff & " seconds" & @CRLF)
;       Exit
EndFunc

Func _ExtractFile($record)
        $cBuffer = DllStructCreate("byte[" & $BytesPerCluster * 16 & "]")
    $zflag = 0
        $hFile = _WinAPI_CreateFile($AttributeOutFileName,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, $record)
                EndSelect
                If $DATA_RealSize > $DATA_InitSize Then
                        $FileSize = _WriteZeros($hfile, $DATA_RealSize - $DATA_InitSize)
                EndIf
                _WinAPI_CloseHandle($hFile)
                Return
        Else

                ConsoleWrite("Error creating output file: " & _WinAPI_GetLastErrorMessage() & @CRLF)
                ;_DisplayInfo("Error creating output file: " & _WinAPI_GetLastErrorMessage() & @CRLF)
        EndIf
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 _DoCompressed($hFile, $cBuffer, $record)
   Local $nBytes
   $r=1
   $FileSize = $DATA_InitSize
   $ProgressSize = $FileSize
   Do
          _WinAPI_SetFilePointerEx($hDisk, $ImageOffset+$RUN_VCN[$r]*$BytesPerCluster, $FILE_BEGIN)
          $i = $RUN_Clusters[$r]
          If (($RUN_VCN[$r+1]=0) And ($i+$RUN_Clusters[$r+1]=16) And $IsCompressed) Then
                 _WinAPI_ReadFile($hDisk, DllStructGetPtr($cBuffer), $BytesPerCluster * $i, $nBytes)
                 $Decompressed = _LZNTDecompress($cBuffer, $BytesPerCluster * $i)
                 If IsString($Decompressed) Then
                        If $r = 1 Then
                           _DebugOut("Decompression error for " & $ADS_Name, $record)
                        Else
                           _DebugOut("Decompression error (partial write) for " & $ADS_Name, $record)
                        EndIf
                        Return
                 Else           ;$Decompressed is an array
                        Local $dBuffer = DllStructCreate("byte[" & $Decompressed[1] & "]")
                        DllStructSetData($dBuffer, 1, $Decompressed[0])
                 EndIf
                 If $FileSize > $Decompressed[1] Then
                        _WinAPI_WriteFile($hFile, DllStructGetPtr($dBuffer), $Decompressed[1], $nBytes)
                        $FileSize -= $Decompressed[1]
                        $ProgressSize = $FileSize
                 Else
                        _WinAPI_WriteFile($hFile, DllStructGetPtr($dBuffer), $FileSize, $nBytes)
                 EndIf
                 $r += 1
          ElseIf $RUN_VCN[$r]=0 Then
                 $FileSize = _DoSparse($r, $hFile, $FileSize)

                 $ProgressSize = 0
          Else
                 $FileSize = _DoNormal($r, $hFile, $cBuffer, $FileSize)
                 $ProgressSize = 0
          EndIf
          $r += 1
   Until $r > UBound($RUN_VCN)-2
   If $r = UBound($RUN_VCN)-1 Then
          If $RUN_VCN[$r]=0 Then
                 $FileSize = _DoSparse($r, $hFile, $FileSize)
                 $ProgressSize = 0
          Else
                 $FileSize = _DoNormal($r, $hFile, $cBuffer, $FileSize)
                 $ProgressSize = 0
          EndIf
   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 _DoSparse($r,$hFile,$FileSize)
   Local $nBytes
   If Not IsDllStruct($sBuffer) Then _CreateSparseBuffer()
   $i = $RUN_Clusters[$r]
   While $i > 16 And $FileSize > $BytesPerCluster * 16
         _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $BytesPerCluster * 16, $nBytes)

         $i -= 16
         $FileSize -= $BytesPerCluster * 16
         $ProgressSize = $FileSize
   WEnd
   If $i <> 0 Then
         If $FileSize > $BytesPerCluster * $i Then
                _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $BytesPerCluster * $i, $nBytes)
                $FileSize -= $BytesPerCluster * $i
                $ProgressSize = $FileSize
         Else
                _WinAPI_WriteFile($hFile, DllStructGetPtr($sBuffer), $FileSize, $nBytes)
                $ProgressSize = 0
                Return 0
         EndIf
   EndIf
   Return $FileSize
EndFunc

Func _CreateSparseBuffer()
   Global $sBuffer = DllStructCreate("byte[" & $BytesPerCluster * 16 & "]")
   For $i = 1 To $BytesPerCluster * 16
          DllStructSetData ($sBuffer, $i, 0)
   Next
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 _ExtractResidentFile($Name, $Size, $record)
        Local $nBytes
        $xBuffer = DllStructCreate("byte[" & $Size & "]")
    DllStructSetData($xBuffer, 1, '0x' & $DataRun)
        $hFile = _WinAPI_CreateFile($Name,3,6,7)

        If $hFile Then
                _WinAPI_SetFilePointer($hFile, 0,$FILE_BEGIN)
                _WinAPI_WriteFile($hFile, DllStructGetPtr($xBuffer), $Size, $nBytes)
                _WinAPI_CloseHandle($hFile)
                Return
        Else
                ConsoleWrite("Error" & @CRLF)
        EndIf
EndFunc

Func _TranslateAttributeType($input)
        Local $RetVal
        Select
                Case $input = $STANDARD_INFORMATION
                        $RetVal = "$STANDARD_INFORMATION"
                Case $input = $ATTRIBUTE_LIST
                        $RetVal = "$ATTRIBUTE_LIST"
                Case $input = $FILE_NAME
                        $RetVal = "$FILE_NAME"
                Case $input = $OBJECT_ID
                        $RetVal = "$OBJECT_ID"
                Case $input = $SECURITY_DESCRIPTOR
                        $RetVal = "$SECURITY_DESCRIPTOR"
                Case $input = $VOLUME_NAME
                        $RetVal = "$VOLUME_NAME"
                Case $input = $VOLUME_INFORMATION
                        $RetVal = "$VOLUME_INFORMATION"
                Case $input = $DATA
                        $RetVal = "$DATA"
                Case $input = $INDEX_ROOT
                        $RetVal = "$INDEX_ROOT"
                Case $input = $INDEX_ALLOCATION
                        $RetVal = "$INDEX_ALLOCATION"
                Case $input = $BITMAP
                        $RetVal = "$BITMAP"
                Case $input = $REPARSE_POINT
                        $RetVal = "$REPARSE_POINT"
                Case $input = $EA_INFORMATION
                        $RetVal = "$EA_INFORMATION"
                Case $input = $EA
                        $RetVal = "$EA"
                Case $input = $PROPERTY_SET
                        $RetVal = "$PROPERTY_SET"
                Case $input = $LOGGED_UTILITY_STREAM
                        $RetVal = "$LOGGED_UTILITY_STREAM"
                Case $input = $ATTRIBUTE_END_MARKER
                        $RetVal = "$ATTRIBUTE_END_MARKER"
        EndSelect
        Return $RetVal
EndFunc


Func NT_SUCCESS($status)
    If 0 <= $status And $status <= 0x7FFFFFFF Then
        Return True
```

```
	Else
		Return False
	EndIf
EndFunc

Func _GetAttributeEntry($Entry)
		Local $CoreAttribute,$CoreAttributeTmp,$CoreAttributeArr[2]
		Local $ATTRIBUTE_HEADER_Length,$ATTRIBUTE_HEADER_NonResidentFlag,$ATTRIBUTE_HEADER_NameLength,$ATTRIBUTE_HEADER_Nam
		Local $ATTRIBUTE_HEADER_VCNs,$ATTRIBUTE_HEADER_OffsetToDataRuns,$ATTRIBUTE_HEADER_CompressionUnitSize,$ATTRIBUTE_HE
		Local $ATTRIBUTE_HEADER_LengthOfAttribute,$ATTRIBUTE_HEADER_OffsetToAttribute,$ATTRIBUTE_HEADER_IndexedFlag
		$ATTRIBUTE_HEADER_Length = StringMid($Entry,9,8)
		$ATTRIBUTE_HEADER_Length = Dec(StringMid($ATTRIBUTE_HEADER_Length,7,2) & StringMid($ATTRIBUTE_HEADER_Length,5,2) &
		$ATTRIBUTE_HEADER_NonResidentFlag = StringMid($Entry,17,2)
;		ConsoleWrite("$ATTRIBUTE_HEADER_NonResidentFlag = " & $ATTRIBUTE_HEADER_NonResidentFlag & @crlf)
		$ATTRIBUTE_HEADER_NameLength = Dec(StringMid($Entry,19,2))
;		ConsoleWrite("$ATTRIBUTE_HEADER_NameLength = " & $ATTRIBUTE_HEADER_NameLength & @crlf)
		$ATTRIBUTE_HEADER_NameRelativeOffset = StringMid($Entry,21,4)
;		ConsoleWrite("$ATTRIBUTE_HEADER_NameRelativeOffset = " & $ATTRIBUTE_HEADER_NameRelativeOffset & @crlf)
		$ATTRIBUTE_HEADER_NameRelativeOffset = Dec(_SwapEndian($ATTRIBUTE_HEADER_NameRelativeOffset))
;		ConsoleWrite("$ATTRIBUTE_HEADER_NameRelativeOffset = " & $ATTRIBUTE_HEADER_NameRelativeOffset & @crlf)
		If $ATTRIBUTE_HEADER_NameLength > 0 Then
			$ATTRIBUTE_HEADER_Name = _UnicodeHexToStr(StringMid($Entry,$ATTRIBUTE_HEADER_NameRelativeOffset*2 + 1,$ATTR
		Else
			$ATTRIBUTE_HEADER_Name = ""
		EndIf
		$ATTRIBUTE_HEADER_Flags = _SwapEndian(StringMid($Entry,25,4))
;		ConsoleWrite("$ATTRIBUTE_HEADER_Flags = " & $ATTRIBUTE_HEADER_Flags & @crlf)
		$Flags = ""
		If $ATTRIBUTE_HEADER_Flags = "0000" Then
			$Flags = "NORMAL"
		Else
			If BitAND($ATTRIBUTE_HEADER_Flags,"0001") Then
				$IsCompressed = 1
				$Flags &= "COMPRESSED+"
			EndIf
			If BitAND($ATTRIBUTE_HEADER_Flags,"4000") Then
				$IsEncrypted = 1
				$Flags &= "ENCRYPTED+"
			EndIf
			If BitAND($ATTRIBUTE_HEADER_Flags,"8000") Then
				$IsSparse = 1
				$Flags &= "SPARSE+"
			EndIf
			$Flags = StringTrimRight($Flags,1)
		EndIf
;		ConsoleWrite("File is " & $Flags & @CRLF)

		$ATTRIBUTE_HEADER_AttributeID = StringMid($Entry,29,4)
		$ATTRIBUTE_HEADER_AttributeID = StringMid($ATTRIBUTE_HEADER_AttributeID,3,2) & StringMid($ATTRIBUTE_HEADER_Attribut
		If $ATTRIBUTE_HEADER_NonResidentFlag = '01' Then
			$ATTRIBUTE_HEADER_StartVCN = StringMid($Entry,33,16)
;			ConsoleWrite("$ATTRIBUTE_HEADER_StartVCN = " & $ATTRIBUTE_HEADER_StartVCN & @crlf)
			$ATTRIBUTE_HEADER_StartVCN = Dec(_SwapEndian($ATTRIBUTE_HEADER_StartVCN),2)
;			ConsoleWrite("$ATTRIBUTE_HEADER_StartVCN = " & $ATTRIBUTE_HEADER_StartVCN & @crlf)
			$ATTRIBUTE_HEADER_LastVCN = StringMid($Entry,49,16)
;			ConsoleWrite("$ATTRIBUTE_HEADER_LastVCN = " & $ATTRIBUTE_HEADER_LastVCN & @crlf)
			$ATTRIBUTE_HEADER_LastVCN = Dec(_SwapEndian($ATTRIBUTE_HEADER_LastVCN),2)
;			ConsoleWrite("$ATTRIBUTE_HEADER_LastVCN = " & $ATTRIBUTE_HEADER_LastVCN & @crlf)
			$ATTRIBUTE_HEADER_VCNs = $ATTRIBUTE_HEADER_LastVCN - $ATTRIBUTE_HEADER_StartVCN
;			ConsoleWrite("$ATTRIBUTE_HEADER_VCNs = " & $ATTRIBUTE_HEADER_VCNs & @crlf)
			$ATTRIBUTE_HEADER_OffsetToDataRuns = StringMid($Entry,65,4)
			$ATTRIBUTE_HEADER_OffsetToDataRuns = Dec(StringMid($ATTRIBUTE_HEADER_OffsetToDataRuns,3,1) & StringMid($ATT
			$ATTRIBUTE_HEADER_CompressionUnitSize = Dec(_SwapEndian(StringMid($Entry,69,4)))
;			ConsoleWrite("$ATTRIBUTE_HEADER_CompressionUnitSize = " & $ATTRIBUTE_HEADER_CompressionUnitSize & @crlf)
			$IsCompressed = 0
			If $ATTRIBUTE_HEADER_CompressionUnitSize = 4 Then $IsCompressed = 1
			$ATTRIBUTE_HEADER_Padding = StringMid($Entry,73,8)
			$ATTRIBUTE_HEADER_Padding = StringMid($ATTRIBUTE_HEADER_Padding,7,2) & StringMid($ATTRIBUTE_HEADER_Padding,
			$ATTRIBUTE_HEADER_AllocatedSize = StringMid($Entry,81,16)
;			ConsoleWrite("$ATTRIBUTE_HEADER_AllocatedSize = " & $ATTRIBUTE_HEADER_AllocatedSize & @crlf)
			$ATTRIBUTE_HEADER_AllocatedSize = Dec(_SwapEndian($ATTRIBUTE_HEADER_AllocatedSize),2)
;			ConsoleWrite("$ATTRIBUTE_HEADER_AllocatedSize = " & $ATTRIBUTE_HEADER_AllocatedSize & @crlf)
			$ATTRIBUTE_HEADER_RealSize = StringMid($Entry,97,16)
;			ConsoleWrite("$ATTRIBUTE_HEADER_RealSize = " & $ATTRIBUTE_HEADER_RealSize & @crlf)
			$ATTRIBUTE_HEADER_RealSize = Dec(_SwapEndian($ATTRIBUTE_HEADER_RealSize),2)
```

```
;               ConsoleWrite("$ATTRIBUTE_HEADER_RealSize = " & $ATTRIBUTE_HEADER_RealSize & @crlf)
                $ATTRIBUTE_HEADER_InitializedStreamSize = StringMid($Entry,113,16)
;               ConsoleWrite("$ATTRIBUTE_HEADER_InitializedStreamSize = " & $ATTRIBUTE_HEADER_InitializedStreamSize & @crlf
                $ATTRIBUTE_HEADER_InitializedStreamSize = Dec(_SwapEndian($ATTRIBUTE_HEADER_InitializedStreamSize),2)
;               ConsoleWrite("$ATTRIBUTE_HEADER_InitializedStreamSize = " & $ATTRIBUTE_HEADER_InitializedStreamSize & @crlf
                $RunListOffset = StringMid($Entry,65,4)
;               ConsoleWrite("$RunListOffset = " & $RunListOffset & @crlf)
                $RunListOffset = Dec(_SwapEndian($RunListOffset))
;               ConsoleWrite("$RunListOffset = " & $RunListOffset & @crlf)
                If $IsCompressed AND $RunListOffset = 72 Then
                        $ATTRIBUTE_HEADER_CompressedSize = StringMid($Entry,129,16)
                        $ATTRIBUTE_HEADER_CompressedSize = Dec(_SwapEndian($ATTRIBUTE_HEADER_CompressedSize),2)
                EndIf
                $DataRun = StringMid($Entry,$RunListOffset*2+1,(StringLen($Entry)-$RunListOffset)*2)
;               ConsoleWrite("$DataRun = " & $DataRun & @crlf)
        ElseIf $ATTRIBUTE_HEADER_NonResidentFlag = '00' Then
                $ATTRIBUTE_HEADER_LengthOfAttribute = StringMid($Entry,33,8)
;               ConsoleWrite("$ATTRIBUTE_HEADER_LengthOfAttribute = " & $ATTRIBUTE_HEADER_LengthOfAttribute & @crlf)
                $ATTRIBUTE_HEADER_LengthOfAttribute = Dec(_SwapEndian($ATTRIBUTE_HEADER_LengthOfAttribute),2)
;               ConsoleWrite("$ATTRIBUTE_HEADER_LengthOfAttribute = " & $ATTRIBUTE_HEADER_LengthOfAttribute & @crlf)
;               $ATTRIBUTE_HEADER_OffsetToAttribute = StringMid($Entry,41,4)
;               $ATTRIBUTE_HEADER_OffsetToAttribute = Dec(StringMid($ATTRIBUTE_HEADER_OffsetToAttribute,3,2) & StringMid($A

                $ATTRIBUTE_HEADER_OffsetToAttribute = Dec(_SwapEndian(StringMid($Entry,41,4)))
;               ConsoleWrite("$ATTRIBUTE_HEADER_OffsetToAttribute = " & $ATTRIBUTE_HEADER_OffsetToAttribute & @crlf)
                $ATTRIBUTE_HEADER_IndexedFlag = Dec(StringMid($Entry,45,2))
                $ATTRIBUTE_HEADER_Padding = StringMid($Entry,47,2)
                $DataRun = StringMid($Entry,$ATTRIBUTE_HEADER_OffsetToAttribute*2+1,$ATTRIBUTE_HEADER_LengthOfAttribute*2)
;               ConsoleWrite("$DataRun = " & $DataRun & @crlf)
        EndIf
; Possible continuation
;       For $i = 1 To UBound($DataQ) - 1
        For $i = 1 To 1
;               _DecodeDataQEntry($DataQ[$i])
                If $ATTRIBUTE_HEADER_NonResidentFlag = '00' Then
;_ExtractResidentFile($DATA_Name, $DATA_LengthOfAttribute)
                        $CoreAttribute = $DataRun
                Else
                        Global $RUN_VCN[1], $RUN_Clusters[1]

                        $TotalClusters = $ATTRIBUTE_HEADER_LastVCN - $ATTRIBUTE_HEADER_StartVCN + 1
                        $Size = $ATTRIBUTE_HEADER_RealSize
;_ExtractDataRuns()
                        $r=UBound($RUN_Clusters)
                        $i=1
                        $RUN_VCN[0] = 0
                        $BaseVCN = $RUN_VCN[0]
                        If $DataRun = "" Then $DataRun = "00"
                        Do
                                $RunListID = StringMid($DataRun,$i,2)
                                If $RunListID = "00" Then ExitLoop
;                               ConsoleWrite("$RunListID = " & $RunListID & @crlf)
                                $i += 2
                                $RunListClustersLength = Dec(StringMid($RunListID,2,1))
;                               ConsoleWrite("$RunListClustersLength = " & $RunListClustersLength & @crlf)
                                $RunListVCNLength = Dec(StringMid($RunListID,1,1))
;                               ConsoleWrite("$RunListVCNLength = " & $RunListVCNLength & @crlf)
                                $RunListClusters = Dec(_SwapEndian(StringMid($DataRun,$i,$RunListClustersLength*2)),2)
;                               ConsoleWrite("$RunListClusters = " & $RunListClusters & @crlf)
                                $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(StringM
                                If $RunListVCN <> "" Then
                                        $RunListVCN = $BaseVCN
                                Else
                                        $RunListVCN = 0                 ;$RUN_VCN[$r-1]         ;0
                                EndIf
;                               ConsoleWrite("$RunListVCN = " & $RunListVCN & @crlf)
                                If (($RunListVCN=0) And ($RunListClusters>16) And (Mod($RunListClusters,16)>0)) Then
                                ;If (($RunListVCN=$RUN_VCN[$r-1]) And ($RunListClusters>16) And (Mod($RunListClusters,16)>0
                                ;may be sparse section at end of Compression Signature
                                        _ArrayAdd($RUN_Clusters,Mod($RunListClusters,16))

                                        _ArrayAdd($RUN_VCN,$RunListVCN)
```

```
					$RunListClusters -= Mod($RunListClusters,16)
					$r += 1
				ElseIf (($RunListClusters>16) And (Mod($RunListClusters,16)>0)) Then
				;may be compressed data section at start of Compression Signature
					_ArrayAdd($RUN_Clusters,$RunListClusters-Mod($RunListClusters,16))
					_ArrayAdd($RUN_VCN,$RunListVCN)
					$RunListVCN += $RUN_Clusters[$r]
					$RunListClusters = Mod($RunListClusters,16)
					$r += 1
				EndIf
			;just normal or sparse data
				_ArrayAdd($RUN_Clusters,$RunListClusters)
				_ArrayAdd($RUN_VCN,$RunListVCN)
				$r += 1
				$i += $RunListVCNLength*2
			Until $i > StringLen($DataRun)
;--------------------------------_ExtractDataRuns()
;			_ArrayDisplay($RUN_Clusters,"$RUN_Clusters")
;			_ArrayDisplay($RUN_VCN,"$RUN_VCN")
			If $TotalClusters * $BytesPerCluster >= $Size Then
;				ConsoleWrite(_ArrayToString($RUN_VCN) & @CRLF)
;				ConsoleWrite(_ArrayToString($RUN_Clusters) & @CRLF)
;ExtractFile
				Local $nBytes
				$hFile = _WinAPI_CreateFile("\\.\" & $TargetDrive, 2, 6, 6)
				If $hFile = 0 Then
					ConsoleWrite("Error in function _WinAPI_CreateFile when trying to open target drive
					_WinAPI_CloseHandle($hFile)
					Return
				EndIf
				$tBuffer = DllStructCreate("byte[" & $BytesPerCluster * 16 & "]")
				Select
					Case UBound($RUN_VCN) = 1		;no data, do nothing
					Case (UBound($RUN_VCN) = 2) Or (Not $IsCompressed)	;may be normal or sparse
						If $RUN_VCN[1] = $RUN_VCN[0] And $DATA_Name <> "$Boot" Then		;sp
;							_DoSparse($htest)
							ConsoleWrite("Error: Sparse attributes not supported!!!" & @CRLF)
						Else								;normal
;							_DoNormalAttribute($hFile, $tBuffer)
;							Local $nBytes
							$FileSize = $ATTRIBUTE_HEADER_RealSize
							For $s = 1 To UBound($RUN_VCN)-1
								_WinAPI_SetFilePointerEx($hFile, $RUN_VCN[$s]*$BytesPerClus
								$g = $RUN_Clusters[$s]
								While $g > 16 And $FileSize > $BytesPerCluster * 16
									_WinAPI_ReadFile($hFile, DllStructGetPtr($tBuffer),
;									_WinAPI_WriteFile($htest, DllStructGetPtr($tBuffer)
									$g -= 16
									$FileSize -= $BytesPerCluster * 16

									$CoreAttributeTmp = StringMid(DllStructGetData($tBu
									$CoreAttribute &= $CoreAttributeTmp
								WEnd
								If $g <> 0 Then
									_WinAPI_ReadFile($hFile, DllStructGetPtr($tBuffer),
;									$CoreAttributeTmp = StringMid(DllStructGetData($tBu
;									$CoreAttribute &= $CoreAttributeTmp
									If $FileSize > $BytesPerCluster * $g Then
;										_WinAPI_WriteFile($htest, DllStructGetPtr($
										$FileSize -= $BytesPerCluster * $g
										$CoreAttributeTmp = StringMid(DllStructGetD
										$CoreAttribute &= $CoreAttributeTmp
									Else
;										_WinAPI_WriteFile($htest, DllStructGetPtr($
;										Return
										$CoreAttributeTmp = StringMid(DllStructGetD
										$CoreAttribute &= $CoreAttributeTmp
									EndIf
								EndIf
							Next
;------------------_DoNormalAttribute()
						EndIf
					Case Else					;may be compressed
;						_DoCompressed($hFile, $htest, $tBuffer)
						ConsoleWrite("Error: Compressed attributes not supported!!!" & @CRLF)
				EndSelect
```

```
                                EndSelect
;------------------------ExtractFile
                        EndIf
;-------------------------
                EndIf
        Next
        $CoreAttributeArr[0] = $CoreAttribute
        $CoreAttributeArr[1] = $ATTRIBUTE_HEADER_Name
        Return $CoreAttributeArr
EndFunc

Func _Get_IndexRoot($Entry,$Current_Attrib_Number,$CurrentAttributeName)
        Local $LocalAttributeOffset = 1,$AttributeType,$CollationRule,$SizeOfIndexAllocationEntry,$ClustersPerIndexRoot,$IR
        $AttributeType = StringMid($Entry,$LocalAttributeOffset,8)
;       $AttributeType = _SwapEndian($AttributeType)
        $CollationRule = StringMid($Entry,$LocalAttributeOffset+8,8)
        $CollationRule = _SwapEndian($CollationRule)
        $SizeOfIndexAllocationEntry = StringMid($Entry,$LocalAttributeOffset+16,8)
        $SizeOfIndexAllocationEntry = Dec(_SwapEndian($SizeOfIndexAllocationEntry),2)
        $ClustersPerIndexRoot = Dec(StringMid($Entry,$LocalAttributeOffset+24,2))
;       $IRPadding = StringMid($Entry,$LocalAttributeOffset+26,6)
        $OffsetToFirstEntry = StringMid($Entry,$LocalAttributeOffset+32,8)
        $OffsetToFirstEntry = Dec(_SwapEndian($OffsetToFirstEntry),2)
        $TotalSizeOfEntries = StringMid($Entry,$LocalAttributeOffset+40,8)
        $TotalSizeOfEntries = Dec(_SwapEndian($TotalSizeOfEntries),2)
        $AllocatedSizeOfEntries = StringMid($Entry,$LocalAttributeOffset+48,8)
        $AllocatedSizeOfEntries = Dec(_SwapEndian($AllocatedSizeOfEntries),2)
        $Flags = StringMid($Entry,$LocalAttributeOffset+56,2)
        If $Flags = "01" Then
                $Flags = "01 (Index Allocation needed)"
                $ResidentIndx = 0
        Else
                $Flags = "00 (Fits in Index Root)"
                $ResidentIndx = 1
        EndIf
;       $IRPadding2 = StringMid($Entry,$LocalAttributeOffset+58,6)
        $IRArr[0][$Current_Attrib_Number] = "IndexRoot Number " & $Current_Attrib_Number
        $IRArr[1][$Current_Attrib_Number] = $CurrentAttributeName
        $IRArr[2][$Current_Attrib_Number] = $AttributeType
        $IRArr[3][$Current_Attrib_Number] = $CollationRule
        $IRArr[4][$Current_Attrib_Number] = $SizeOfIndexAllocationEntry
        $IRArr[5][$Current_Attrib_Number] = $ClustersPerIndexRoot
;       $IRArr[6][$Current_Attrib_Number] = $IRPadding
        $IRArr[7][$Current_Attrib_Number] = $OffsetToFirstEntry
        $IRArr[8][$Current_Attrib_Number] = $TotalSizeOfEntries
        $IRArr[9][$Current_Attrib_Number] = $AllocatedSizeOfEntries
        $IRArr[10][$Current_Attrib_Number] = $Flags
;       $IRArr[11][$Current_Attrib_Number] = $IRPadding2
        $TheResidentIndexEntry = StringMid($Entry,$LocalAttributeOffset+64)
        If $ResidentIndx And $AttributeType=$FILE_NAME Then
;               $TheResidentIndexEntry = StringMid($Entry,$LocalAttributeOffset+64)
                _DecodeIndxEntries($TheResidentIndexEntry,$Current_Attrib_Number,$CurrentAttributeName)
        ElseIf $ResidentIndx=0 And $AttributeType=$FILE_NAME Then
                _DecodeIndxEntries($TheResidentIndexEntry,$Current_Attrib_Number,$CurrentAttributeName)
        EndIf
EndFunc

Func _StripIndxRecord($Entry)
;       ConsoleWrite("Starting function _StripIndxRecord()" & @crlf)
        Local $LocalAttributeOffset = 1,$IndxHdrUpdateSeqArrOffset,$IndxHdrUpdateSeqArrSize,$IndxHdrUpdSeqArr,$IndxHdrUpdSe
        Local $IndxRecordEnd1,$IndxRecordEnd2,$IndxRecordEnd3,$IndxRecordEnd4,$IndxRecordEnd5,$IndxRecordEnd6,$IndxRecordEn
;       ConsoleWrite("Unfixed INDX record:" & @crlf)
;       ConsoleWrite(_HexEncode("0x"&$Entry) & @crlf)
;       ConsoleWrite(_HexEncode("0x" & StringMid($Entry,1,4096)) & @crlf)
        $IndxHdrUpdateSeqArrOffset = Dec(_SwapEndian(StringMid($Entry,$LocalAttributeOffset+8,4)))
;       ConsoleWrite("$IndxHdrUpdateSeqArrOffset = " & $IndxHdrUpdateSeqArrOffset & @crlf)
        $IndxHdrUpdateSeqArrSize = Dec(_SwapEndian(StringMid($Entry,$LocalAttributeOffset+12,4)))
;       ConsoleWrite("$IndxHdrUpdateSeqArrSize = " & $IndxHdrUpdateSeqArrSize & @crlf)
        $IndxHdrUpdSeqArr = StringMid($Entry,1+($IndxHdrUpdateSeqArrOffset*2),$IndxHdrUpdateSeqArrSize*2*2)
;       ConsoleWrite("$IndxHdrUpdSeqArr = " & $IndxHdrUpdSeqArr & @crlf)
        $IndxHdrUpdSeqArrPart0 = StringMid($IndxHdrUpdSeqArr,1,4)
        $IndxHdrUpdSeqArrPart1 = StringMid($IndxHdrUpdSeqArr,5,4)
        $IndxHdrUpdSeqArrPart2 = StringMid($IndxHdrUpdSeqArr,9,4)
        $IndxHdrUpdSeqArrPart3 = StringMid($IndxHdrUpdSeqArr,13,4)
        $IndxHdrUpdSeqArrPart4 = StringMid($IndxHdrUpdSeqArr,17,4)
        $IndxHdrUpdSeqArrPart4 = StringMid($IndxHdrUpdSeqArr,17,4)
        $IndxHdrUpdSeqArrPart5 = StringMid($IndxHdrUpdSeqArr,21,4)
        $IndxHdrUpdSeqArrPart6 = StringMid($IndxHdrUpdSeqArr,25,4)
        $IndxHdrUpdSeqArrPart7 = StringMid($IndxHdrUpdSeqArr,29,4)
        $IndxHdrUpdSeqArrPart8 = StringMid($IndxHdrUpdSeqArr,33,4)
        $IndxRecordEnd1 = StringMid($Entry,1021,4)
        $IndxRecordEnd2 = StringMid($Entry,2045,4)
        $IndxRecordEnd3 = StringMid($Entry,3069,4)
        $IndxRecordEnd4 = StringMid($Entry,4093,4)
        $IndxRecordEnd5 = StringMid($Entry,5117,4)
        $IndxRecordEnd6 = StringMid($Entry,6141,4)
        $IndxRecordEnd7 = StringMid($Entry,7165,4)
        $IndxRecordEnd8 = StringMid($Entry,8189,4)
        If $IndxHdrUpdSeqArrPart0 <> $IndxRecordEnd1 OR $IndxHdrUpdSeqArrPart0 <> $IndxRecordEnd2 OR $IndxHdrUpdSeqArrPart0
                ConsoleWrite("Error the INDX record is corrupt" & @CRLF)
                Return ; Not really correct because I think in theory chunks of 1024 bytes can be invalid and not just ever
        Else
                $Entry = StringMid($Entry,1,1020) & $IndxHdrUpdSeqArrPart1 & StringMid($Entry,1025,1020) & $IndxHdrUpdSeqAr
        EndIf
        $IndxRecordSize = Dec(_SwapEndian(StringMid($Entry,$LocalAttributeOffset+56,8)),2)
;       ConsoleWrite("$IndxRecordSize = " & $IndxRecordSize & @crlf)
        $IndxHeaderSize = Dec(_SwapEndian(StringMid($Entry,$LocalAttributeOffset+48,8)),2)
;       ConsoleWrite("$IndxHeaderSize = " & $IndxHeaderSize & @crlf)
        $IsNotLeafNode = StringMid($Entry,$LocalAttributeOffset+72,2) ;1 if not leaf node
        $Entry = StringMid($Entry,$LocalAttributeOffset+48+($IndxHeaderSize*2),($IndxRecordSize-$IndxHeaderSize-16)*2)
        If $IsNotLeafNode = "01" Then  ; This flag leads to the entry being 8 bytes of 00's longer than the others. Can be
                $Entry = StringTrimRight($Entry,16)
;               ConsoleWrite("Is not leaf node..." & @crlf)
        EndIf
        Return $Entry
EndFunc

Func _Get_IndexAllocation($Entry,$Current_Attrib_Number,$CurrentAttributeName)
;       ConsoleWrite("Starting function _Get_IndexAllocation()" & @crlf)
        Local $NextPosition = 1,$IndxHdrMagic,$IndxEntries,$TotalIndxEntries
;       ConsoleWrite("StringLen of chunk = " & StringLen($Entry) & @crlf)
;       ConsoleWrite("Expected records = " & StringLen($Entry)/8192 & @crlf)
        $NextPosition = 1
        Do
                $IndxHdrMagic = StringMid($Entry,$NextPosition,8)
;               ConsoleWrite("$IndxHdrMagic = " & $IndxHdrMagic & @crlf)
                $IndxHdrMagic = _HexToString($IndxHdrMagic)
;               ConsoleWrite("$IndxHdrMagic = " & $IndxHdrMagic & @crlf)
                If $IndxHdrMagic <> "INDX" Then
;                       ConsoleWrite("$IndxHdrMagic: " & $IndxHdrMagic & @crlf)
;                       ConsoleWrite("Error: Record is not of type INDX, and this was not expected.." & @crlf)
                        $NextPosition += 8192
                        ContinueLoop
                EndIf
                $IndxEntries = _StripIndxRecord(StringMid($Entry,$NextPosition,8192))
                $TotalIndxEntries &= $IndxEntries

                $NextPosition += 8192
        Until $NextPosition >= StringLen($Entry)+32
;       ConsoleWrite("INDX record:" & @crlf)
;       ConsoleWrite(_HexEncode("0x"& StringMid($Entry,1)) & @crlf)
;       ConsoleWrite("Total chunk of stripped INDX entries:" & @crlf)
;       ConsoleWrite(_HexEncode("0x"& StringMid($TotalIndxEntries,1)) & @crlf)
        _DecodeIndxEntries($TotalIndxEntries,$Current_Attrib_Number,$CurrentAttributeName)
EndFunc

Func _DecodeIndxEntries($Entry,$Current_Attrib_Number,$CurrentAttributeName)
;       ConsoleWrite("Starting function _DecodeIndxEntries()" & @crlf)
        Local $LocalAttributeOffset = 1,$NewLocalAttributeOffset,$IndxHdrMagic,$IndxHdrUpdateSeqArrOffset,$IndxHdrUpdateSeq
        Local $IndxHdrFlag,$IndxHdrPadding,$IndxHdrUpdateSequence,$IndxHdrUpdSeqArr,$IndxHdrUpdSeqArrPart0,$IndxHdrUpdSeqAr
        Local $FileReference,$IndexEntryLength,$StreamLength,$Flags,$Stream,$SubNodeVCN,$tmp0=0,$tmp1=0,$tmp2=0,$tmp3=0,$En
        $NewLocalAttributeOffset = 1
        $MFTReference = StringMid($Entry,$NewLocalAttributeOffset,12)
        $MFTReference = StringMid($MFTReference,7,2)&StringMid($MFTReference,5,2)&StringMid($MFTReference,3,2)&StringMid($M
        $MFTReference = Dec($MFTReference)
        $MFTReferenceSeqNo = StringMid($Entry,$NewLocalAttributeOffset+12,4)
        $MFTReferenceSeqNo = Dec(StringMid($MFTReferenceSeqNo,3,2)&StringMid($MFTReferenceSeqNo,1,2))
        $IndexEntryLength = StringMid($Entry,$NewLocalAttributeOffset+16,4)
        $IndexEntryLength = Dec(StringMid($IndexEntryLength,3,2)&StringMid($IndexEntryLength,3,2))
        $OffsetToFileName = StringMid($Entry,$NewLocalAttributeOffset+20,4)
        $OffsetToFileName = StringMid($Entry,$NewLocalAttributeOffset+20,4)
        $OffsetToFileName = Dec(StringMid($OffsetToFileName,3,2)&StringMid($OffsetToFileName,3,2))
        $IndexFlags = StringMid($Entry,$NewLocalAttributeOffset+24,4)
;       $Padding = StringMid($Entry,$NewLocalAttributeOffset+28,4)
        $MFTReferenceOfParent = StringMid($Entry,$NewLocalAttributeOffset+32,12)
        $MFTReferenceOfParent = StringMid($MFTReferenceOfParent,7,2)&StringMid($MFTReferenceOfParent,5,2)&StringMid($MFTRef
        $MFTReferenceOfParent = Dec($MFTReferenceOfParent)
        $MFTReferenceOfParentSeqNo = StringMid($Entry,$NewLocalAttributeOffset+44,4)
        $MFTReferenceOfParentSeqNo = Dec(StringMid($MFTReferenceOfParentSeqNo,3,2) & StringMid($MFTReferenceOfParentSeqNo,3
        $Indx_CTime = StringMid($Entry,$NewLocalAttributeOffset+48,16)
        $Indx_CTime = StringMid($Indx_CTime,15,2) & StringMid($Indx_CTime,13,2) & StringMid($Indx_CTime,11,2) & StringMid($
        $Indx_CTime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_CTime)
        $Indx_CTime = _WinTime_UTCFileTimeFormat(Dec($Indx_CTime)-$tDelta,$DateTimeFormat,2)
        If @error Then
;               $Indx_CTime = "-"
                $Indx_CTime = "1601-01-01 00:00:00:000:0000"
        Else
                $Indx_CTime = $Indx_CTime & ":" & _FillZero(StringRight($Indx_CTime_tmp,4))
        EndIf
        $Indx_ATime = StringMid($Entry,$NewLocalAttributeOffset+64,16)
        $Indx_ATime = StringMid($Indx_ATime,15,2) & StringMid($Indx_ATime,13,2) & StringMid($Indx_ATime,11,2) & StringMid($
        $Indx_ATime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_ATime)
        $Indx_ATime = _WinTime_UTCFileTimeFormat(Dec($Indx_ATime)-$tDelta,$DateTimeFormat,2)
        If @error Then
;               $Indx_ATime = "-"
                $Indx_ATime = "1601-01-01 00:00:00:000:0000"
        Else
                $Indx_ATime = $Indx_ATime & ":" & _FillZero(StringRight($Indx_ATime_tmp,4))

        EndIf
        $Indx_MTime = StringMid($Entry,$NewLocalAttributeOffset+80,16)
        $Indx_MTime = StringMid($Indx_MTime,15,2) & StringMid($Indx_MTime,13,2) & StringMid($Indx_MTime,11,2) & StringMid($
        $Indx_MTime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_MTime)
        $Indx_MTime = _WinTime_UTCFileTimeFormat(Dec($Indx_MTime)-$tDelta,$DateTimeFormat,2)
        If @error Then
;               $Indx_MTime = "-"
                $Indx_MTime = "1601-01-01 00:00:00:000:0000"
        Else
                $Indx_MTime = $Indx_MTime & ":" & _FillZero(StringRight($Indx_MTime_tmp,4))
        EndIf
        $Indx_RTime = StringMid($Entry,$NewLocalAttributeOffset+96,16)
        $Indx_RTime = StringMid($Indx_RTime,15,2) & StringMid($Indx_RTime,13,2) & StringMid($Indx_RTime,11,2) & StringMid($
        $Indx_RTime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_RTime)
        $Indx_RTime = _WinTime_UTCFileTimeFormat(Dec($Indx_RTime)-$tDelta,$DateTimeFormat,2)
        If @error Then
;               $Indx_RTime = "-"
                $Indx_RTime = "1601-01-01 00:00:00:000:0000"
        Else
                $Indx_RTime = $Indx_RTime & ":" & _FillZero(StringRight($Indx_RTime_tmp,4))
        EndIf
        $Indx_AllocSize = StringMid($Entry,$NewLocalAttributeOffset+112,16)
        $Indx_AllocSize = Dec(StringMid($Indx_AllocSize,15,2) & StringMid($Indx_AllocSize,13,2) & StringMid($Indx_AllocSize
        $Indx_RealSize = StringMid($Entry,$NewLocalAttributeOffset+128,16)
        $Indx_RealSize = Dec(StringMid($Indx_RealSize,15,2) & StringMid($Indx_RealSize,13,2) & StringMid($Indx_RealSize,11,
        $Indx_File_Flags = StringMid($Entry,$NewLocalAttributeOffset+144,16)
;       $Indx_File_Flags = StringMid($Indx_File_Flags,15,2) & StringMid($Indx_File_Flags,13,2) & StringMid($Indx_File_Flags
        $Indx_File_Flags = StringMid(_SwapEndian($Indx_File_Flags),9,8)
        $Indx_File_Flags = _File_Attributes("0x" & $Indx_File_Flags)
        $Indx_NameLength = StringMid($Entry,$NewLocalAttributeOffset+160,2)
        $Indx_NameLength = Dec($Indx_NameLength)
        $Indx_NameSpace = StringMid($Entry,$NewLocalAttributeOffset+162,2)
        Select
                Case $Indx_NameSpace = "00"     ;POSIX
                        $Indx_NameSpace = "POSIX"
                Case $Indx_NameSpace = "01"     ;WIN32
                        $Indx_NameSpace = "WIN32"
                Case $Indx_NameSpace = "02"     ;DOS
                        $Indx_NameSpace = "DOS"
                Case $Indx_NameSpace = "03"     ;DOS+WIN32
                        $Indx_NameSpace = "DOS+WIN32"
        EndSelect
        $Indx_FileName = StringMid($Entry,$NewLocalAttributeOffset+164,$Indx_NameLength*2*2)
        $Indx_FileName = _UnicodeHexToStr($Indx_FileName)
        $tmp1 = 164+($Indx_NameLength*2*2)
        Do ; Calculate the length of the padding - 8 byte aligned
                $tmp2 = $tmp1/16
                If Not IsInt($tmp2) Then
                        $tmp0 = 2
                        $tmp1 += $tmp0

                        $tmp3 += $tmp0
                EndIf
        Until IsInt($tmp2)
        $PaddingLength = $tmp3
;       $Padding2 = StringMid($Entry,$NewLocalAttributeOffset+164+($Indx_NameLength*2*2),$PaddingLength)
        If $IndexFlags <> "0000" Then
                $SubNodeVCN = StringMid($Entry,$NewLocalAttributeOffset+164+($Indx_NameLength*2*2)+$PaddingLength,16)
                $SubNodeVCNLength = 16
        Else
                $SubNodeVCN = "-"
                $SubNodeVCNLength = 0
        EndIf
        ReDim $IndxEntryNumberArr[1+$EntryCounter]
        ReDim $IndxMFTReferenceArr[1+$EntryCounter]
        ReDim $IndxMFTRefSeqNoArr[1+$EntryCounter]
        ReDim $IndxIndexFlagsArr[1+$EntryCounter]
        ReDim $IndxMFTReferenceOfParentArr[1+$EntryCounter]
        ReDim $IndxMFTParentRefSeqNoArr[1+$EntryCounter]
        ReDim $IndxCTimeArr[1+$EntryCounter]
        ReDim $IndxATimeArr[1+$EntryCounter]
        ReDim $IndxMTimeArr[1+$EntryCounter]
        ReDim $IndxRTimeArr[1+$EntryCounter]
        ReDim $IndxAllocSizeArr[1+$EntryCounter]
        ReDim $IndxRealSizeArr[1+$EntryCounter]
        ReDim $IndxFileFlagsArr[1+$EntryCounter]
        ReDim $IndxFileNameArr[1+$EntryCounter]
        ReDim $IndxNameSpaceArr[1+$EntryCounter]
        ReDim $IndxSubNodeVCNArr[1+$EntryCounter]
        $IndxEntryNumberArr[$EntryCounter] = $EntryCounter
        $IndxMFTReferenceArr[$EntryCounter] = $MFTReference
        $IndxMFTRefSeqNoArr[$EntryCounter] = $MFTReferenceSeqNo
        $IndxIndexFlagsArr[$EntryCounter] = $IndexFlags
        $IndxMFTReferenceOfParentArr[$EntryCounter] = $MFTReferenceOfParent
        $IndxMFTParentRefSeqNoArr[$EntryCounter] = $MFTReferenceOfParentSeqNo
        $IndxCTimeArr[$EntryCounter] = $Indx_CTime
        $IndxATimeArr[$EntryCounter] = $Indx_ATime
        $IndxMTimeArr[$EntryCounter] = $Indx_MTime
        $IndxRTimeArr[$EntryCounter] = $Indx_RTime
        $IndxAllocSizeArr[$EntryCounter] = $Indx_AllocSize
        $IndxRealSizeArr[$EntryCounter] = $Indx_RealSize
        $IndxFileFlagsArr[$EntryCounter] = $Indx_File_Flags
        $IndxFileNameArr[$EntryCounter] = $Indx_FileName
        $IndxNameSpaceArr[$EntryCounter] = $Indx_NameSpace
        $IndxSubNodeVCNArr[$EntryCounter] = $SubNodeVCN
; Work through the rest of the index entries
        $NextEntryOffset = $NewLocalAttributeOffset+164+($Indx_NameLength*2*2)+$PaddingLength+$SubNodeVCNLength
        If $NextEntryOffset+64 >= StringLen($Entry) Then Return
        Do
                $EntryCounter += 1
;               ConsoleWrite("$EntryCounter = " & $EntryCounter & @crlf)

                $MFTReference = StringMid($Entry,$NextEntryOffset,12)
;               ConsoleWrite("$MFTReference = " & $MFTReference & @crlf)
                $MFTReference = StringMid($MFTReference,7,2)&StringMid($MFTReference,5,2)&StringMid($MFTReference,3,2)&Stri
;               $MFTReference = StringMid($MFTReference,15,2)&StringMid($MFTReference,13,2)&StringMid($MFTReference,11,2)&S
;               ConsoleWrite("$MFTReference = " & $MFTReference & @crlf)
                $MFTReference = Dec($MFTReference)
                $MFTReferenceSeqNo = StringMid($Entry,$NextEntryOffset+12,4)
                $MFTReferenceSeqNo = Dec(StringMid($MFTReferenceSeqNo,3,2)&StringMid($MFTReferenceSeqNo,1,2))
                $IndexEntryLength = StringMid($Entry,$NextEntryOffset+16,4)
;               ConsoleWrite("$IndexEntryLength = " & $IndexEntryLength & @crlf)
                $IndexEntryLength = Dec(StringMid($IndexEntryLength,3,2)&StringMid($IndexEntryLength,3,2))
;               ConsoleWrite("$IndexEntryLength = " & $IndexEntryLength & @crlf)
                $OffsetToFileName = StringMid($Entry,$NextEntryOffset+20,4)
;               ConsoleWrite("$OffsetToFileName = " & $OffsetToFileName & @crlf)
                $OffsetToFileName = Dec(StringMid($OffsetToFileName,3,2)&StringMid($OffsetToFileName,3,2))
;               ConsoleWrite("$OffsetToFileName = " & $OffsetToFileName & @crlf)
                $IndexFlags = StringMid($Entry,$NextEntryOffset+24,4)
;               ConsoleWrite("$IndexFlags = " & $IndexFlags & @crlf)
                $Padding = StringMid($Entry,$NextEntryOffset+28,4)
;               ConsoleWrite("$Padding = " & $Padding & @crlf)
```

```
                $MFTReferenceOfParent = StringMid($Entry,$NextEntryOffset+32,12)
;               ConsoleWrite("$MFTReferenceOfParent = " & $MFTReferenceOfParent & @crlf)
                $MFTReferenceOfParent = StringMid($MFTReferenceOfParent,7,2)&StringMid($MFTReferenceOfParent,5,2)&StringMid
;               $MFTReferenceOfParent = StringMid($MFTReferenceOfParent,15,2)&StringMid($MFTReferenceOfParent,13,2)&StringM
;               ConsoleWrite("$MFTReferenceOfParent = " & $MFTReferenceOfParent & @crlf)
                $MFTReferenceOfParent = Dec($MFTReferenceOfParent)
                $MFTReferenceOfParentSeqNo = StringMid($Entry,$NextEntryOffset+44,4)
                $MFTReferenceOfParentSeqNo = Dec(StringMid($MFTReferenceOfParentSeqNo,3,2) & StringMid($MFTReferenceOfParen

                $Indx_CTime = StringMid($Entry,$NextEntryOffset+48,16)
                $Indx_CTime = StringMid($Indx_CTime,15,2) & StringMid($Indx_CTime,13,2) & StringMid($Indx_CTime,11,2) & Str
                $Indx_CTime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_CTime)
                $Indx_CTime = _WinTime_UTCFileTimeFormat(Dec($Indx_CTime)-$tDelta,$DateTimeFormat,2)
                If @error Then
;                       $Indx_CTime = "-"
                        $Indx_CTime = "1601-01-01 00:00:00:000:0000"
                Else
                        $Indx_CTime = $Indx_CTime & ":" & _FillZero(StringRight($Indx_CTime_tmp,4))
                EndIf
                $Indx_ATime = StringMid($Entry,$NextEntryOffset+64,16)
                $Indx_ATime = StringMid($Indx_ATime,15,2) & StringMid($Indx_ATime,13,2) & StringMid($Indx_ATime,11,2) & Str
                $Indx_ATime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_ATime)
                $Indx_ATime = _WinTime_UTCFileTimeFormat(Dec($Indx_ATime)-$tDelta,$DateTimeFormat,2)
                If @error Then
;                       $Indx_ATime = "-"
                        $Indx_ATime = "1601-01-01 00:00:00:000:0000"
                Else
                        $Indx_ATime = $Indx_ATime & ":" & _FillZero(StringRight($Indx_ATime_tmp,4))
                EndIf
                $Indx_MTime = StringMid($Entry,$NextEntryOffset+80,16)

                $Indx_MTime = StringMid($Indx_MTime,15,2) & StringMid($Indx_MTime,13,2) & StringMid($Indx_MTime,11,2) & Str
                $Indx_MTime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_MTime)
                $Indx_MTime = _WinTime_UTCFileTimeFormat(Dec($Indx_MTime)-$tDelta,$DateTimeFormat,2)
                If @error Then
;                       $Indx_MTime = "-"
                        $Indx_MTime = "1601-01-01 00:00:00:000:0000"
                Else
                        $Indx_MTime = $Indx_MTime & ":" & _FillZero(StringRight($Indx_MTime_tmp,4))
                EndIf
                $Indx_RTime = StringMid($Entry,$NextEntryOffset+96,16)
                $Indx_RTime = StringMid($Indx_RTime,15,2) & StringMid($Indx_RTime,13,2) & StringMid($Indx_RTime,11,2) & Str
                $Indx_RTime_tmp = _WinTime_UTCFileTimeToLocalFileTime("0x" & $Indx_RTime)
                $Indx_RTime = _WinTime_UTCFileTimeFormat(Dec($Indx_RTime)-$tDelta,$DateTimeFormat,2)
                If @error Then
;                       $Indx_RTime = "-"
                        $Indx_RTime = "1601-01-01 00:00:00:000:0000"
                Else
                        $Indx_RTime = $Indx_RTime & ":" & _FillZero(StringRight($Indx_RTime_tmp,4))
                EndIf
                $Indx_AllocSize = StringMid($Entry,$NextEntryOffset+112,16)
                $Indx_AllocSize = Dec(StringMid($Indx_AllocSize,15,2) & StringMid($Indx_AllocSize,13,2) & StringMid($Indx_A
;               ConsoleWrite("$Indx_AllocSize = " & $Indx_AllocSize & @crlf)
                $Indx_RealSize = StringMid($Entry,$NextEntryOffset+128,16)
                $Indx_RealSize = Dec(StringMid($Indx_RealSize,15,2) & StringMid($Indx_RealSize,13,2) & StringMid($Indx_Real
;               ConsoleWrite("$Indx_RealSize = " & $Indx_RealSize & @crlf)
                $Indx_File_Flags = StringMid($Entry,$NextEntryOffset+144,16)
;               ConsoleWrite("$Indx_File_Flags = " & $Indx_File_Flags & @crlf)
;               $Indx_File_Flags = StringMid($Indx_File_Flags,15,2) & StringMid($Indx_File_Flags,13,2) & StringMid($Indx_Fi
;               ConsoleWrite("$Indx_File_Flags = " & $Indx_File_Flags & @crlf)
                $Indx_File_Flags = StringMid(_SwapEndian($Indx_File_Flags),9,8)
                $Indx_File_Flags = _File_Attributes("0x" & $Indx_File_Flags)
;               ConsoleWrite("$Indx_File_Flags = " & $Indx_File_Flags & @crlf)
                $Indx_NameLength = StringMid($Entry,$NextEntryOffset+160,2)
                $Indx_NameLength = Dec($Indx_NameLength)
;               ConsoleWrite("$Indx_NameLength = " & $Indx_NameLength & @crlf)
                $Indx_NameSpace = StringMid($Entry,$NextEntryOffset+162,2)
;               ConsoleWrite("$Indx_NameSpace = " & $Indx_NameSpace & @crlf)
                Select
                        Case $Indx_NameSpace = "00"     ;POSIX
                                $Indx_NameSpace = "POSIX"
                        Case $Indx_NameSpace = "01"     ;WIN32
                                $Indx_NameSpace = "WIN32"
                        Case $Indx_NameSpace = "02"     ;DOS
                                $Indx_NameSpace = "DOS"
```

```
				Case $Indx_NameSpace = "03"	;DOS+WIN32
					$Indx_NameSpace = "DOS+WIN32"
			EndSelect
			$Indx_FileName = StringMid($Entry,$NextEntryOffset+164,$Indx_NameLength*2*2)
;			ConsoleWrite("$Indx_FileName = " & $Indx_FileName & @crlf)
			$Indx_FileName = _UnicodeHexToStr($Indx_FileName)

;			ConsoleWrite("$Indx_FileName = " & $Indx_FileName & @crlf)
			$tmp0 = 0
			$tmp2 = 0
			$tmp3 = 0
			$tmp1 = 164+($Indx_NameLength*2*2)
			Do ; Calculate the length of the padding - 8 byte aligned
				$tmp2 = $tmp1/16
				If Not IsInt($tmp2) Then
					$tmp0 = 2
					$tmp1 += $tmp0
					$tmp3 += $tmp0
				EndIf
			Until IsInt($tmp2)
			$PaddingLength = $tmp3
;			ConsoleWrite("$PaddingLength = " & $PaddingLength & @crlf)
			$Padding = StringMid($Entry,$NextEntryOffset+164+($Indx_NameLength*2*2),$PaddingLength)
;			ConsoleWrite("$Padding = " & $Padding & @crlf)
			If $IndexFlags <> "0000" Then
				$SubNodeVCN = StringMid($Entry,$NextEntryOffset+164+($Indx_NameLength*2*2)+$PaddingLength,16)
				$SubNodeVCNLength = 16
			Else
				$SubNodeVCN = "-"
				$SubNodeVCNLength = 0
			EndIf
;			ConsoleWrite("$SubNodeVCN = " & $SubNodeVCN & @crlf)
			$NextEntryOffset = $NextEntryOffset+164+($Indx_NameLength*2*2)+$PaddingLength+$SubNodeVCNLength
			ReDim $IndxEntryNumberArr[1+$EntryCounter]
			ReDim $IndxMFTReferenceArr[1+$EntryCounter]
			Redim $IndxMFTRefSeqNoArr[1+$EntryCounter]
			ReDim $IndxIndexFlagsArr[1+$EntryCounter]
			ReDim $IndxMFTReferenceOfParentArr[1+$EntryCounter]
			ReDim $IndxMFTParentRefSeqNoArr[1+$EntryCounter]
			ReDim $IndxCTimeArr[1+$EntryCounter]
			ReDim $IndxATimeArr[1+$EntryCounter]
			ReDim $IndxMTimeArr[1+$EntryCounter]
			ReDim $IndxRTimeArr[1+$EntryCounter]
			ReDim $IndxAllocSizeArr[1+$EntryCounter]
			ReDim $IndxRealSizeArr[1+$EntryCounter]
			ReDim $IndxFileFlagsArr[1+$EntryCounter]
			ReDim $IndxFileNameArr[1+$EntryCounter]
			ReDim $IndxNameSpaceArr[1+$EntryCounter]
			ReDim $IndxSubNodeVCNArr[1+$EntryCounter]
			$IndxEntryNumberArr[$EntryCounter] = $EntryCounter
			$IndxMFTReferenceArr[$EntryCounter] = $MFTReference
			$IndxMFTRefSeqNoArr[$EntryCounter] = $MFTReferenceSeqNo
			$IndxIndexFlagsArr[$EntryCounter] = $IndexFlags
			$IndxMFTReferenceOfParentArr[$EntryCounter] = $MFTReferenceOfParent
			$IndxMFTParentRefSeqNoArr[$EntryCounter] = $MFTReferenceOfParentSeqNo
			$IndxCTimeArr[$EntryCounter] = $Indx_CTime
			$IndxATimeArr[$EntryCounter] = $Indx_ATime

			$IndxMTimeArr[$EntryCounter] = $Indx_MTime
			$IndxRTimeArr[$EntryCounter] = $Indx_RTime
			$IndxAllocSizeArr[$EntryCounter] = $Indx_AllocSize
			$IndxRealSizeArr[$EntryCounter] = $Indx_RealSize
			$IndxFileFlagsArr[$EntryCounter] = $Indx_File_Flags
			$IndxFileNameArr[$EntryCounter] = $Indx_FileName
			$IndxNameSpaceArr[$EntryCounter] = $Indx_NameSpace
			$IndxSubNodeVCNArr[$EntryCounter] = $SubNodeVCN
;			_ArrayDisplay($IndxFileNameArr,"$IndxFileNameArr")
	Until $NextEntryOffset+32 >= StringLen($Entry)
EndFunc

Func _SetArrays()
	$IndxEntryNumberArr[0] = "Entry number"
	$IndxMFTReferenceArr[0] = "MFTReference"
	$IndxMFTRefSeqNoArr[0] = "MFTReference SeqNo"
	$IndxIndexFlagsArr[0] = "IndexFlags"
	$IndxMFTReferenceOfParentArr[0] = "Parent MFTReference"
	$IndxMFTParentRefSeqNoArr[0] = "Parent MFTReference SeqNo"
	$IndxCTimeArr[0] = "CTime"
	$IndxATimeArr[0] = "ATime"
	$IndxMTimeArr[0] = "MTime"
	$IndxRTimeArr[0] = "RTime"
	$IndxAllocSizeArr[0] = "AllocSize"
	$IndxRealSizeArr[0] = "RealSize"
	$IndxFileFlagsArr[0] = "File flags"
	$IndxFileNameArr[0] = "FileName"
	$IndxNameSpaceArr[0] = "NameSpace"
	$IndxSubNodeVCNArr[0] = "SubNodeVCN"
EndFunc

Func _FillZero($inp)
	Local $inplen, $out, $tmp = ""
	$inplen = StringLen($inp)
	For $i = 1 To 4-$inplen
		$tmp &= "0"
	Next
	$out = $tmp & $inp
	Return $out
EndFunc

; start: by Ascend4nt -----------------------------
Func _WinTime_GetUTCToLocalFileTimeDelta()
	Local $iUTCFileTime=864000000000		; exactly 24 hours from the origin (although 12 hours would be more
	$iLocalFileTime=_WinTime_UTCFileTimeToLocalFileTime($iUTCFileTime)
	If @error Then Return SetError(@error,@extended,-1)
	Return $iLocalFileTime-$iUTCFileTime	; /36000000000 = # hours delta (effectively giving the offset in hours from
EndFunc

Func _WinTime_UTCFileTimeToLocalFileTime($iUTCFileTime)

	If $iUTCFileTime<0 Then Return SetError(1,0,-1)
	Local $aRet=DllCall($_COMMON_KERNEL32DLL,"bool","FileTimeToLocalFileTime","uint64*",$iUTCFileTime,"uint64*",0)
	If @error Then Return SetError(2,@error,-1)
	If Not $aRet[0] Then Return SetError(3,0,-1)
	Return $aRet[2]
EndFunc

Func _WinTime_UTCFileTimeFormat($iUTCFileTime,$iFormat=4,$iPrecision=0,$bAMPMConversion=False)
;~	If $iUTCFileTime<0 Then Return SetError(1,0,"") ; checked in below call

	; First convert file time (UTC-based file time) to 'local file time'
	Local $iLocalFileTime=_WinTime_UTCFileTimeToLocalFileTime($iUTCFileTime)
	If @error Then Return SetError(@error,@extended,"")
	; Rare occassion: a filetime near the origin (January 1, 1601!!) is used,
	;	causing a negative result (for some timezones). Return as invalid param.
	If $iLocalFileTime<0 Then Return SetError(1,0,"")

	; Then convert file time to a system time array & format & return it
	Local $vReturn=_WinTime_LocalFileTimeFormat($iLocalFileTime,$iFormat,$iPrecision,$bAMPMConversion)
	Return SetError(@error,@extended,$vReturn)
EndFunc

Func _WinTime_LocalFileTimeFormat($iLocalFileTime,$iFormat=4,$iPrecision=0,$bAMPMConversion=False)
;~	If $iLocalFileTime<0 Then Return SetError(1,0,"")	; checked in below call

	; Convert file time to a system time array & return result
	Local $aSysTime=_WinTime_LocalFileTimeToSystemTime($iLocalFileTime)
	If @error Then Return SetError(@error,@extended,"")

	; Return only the SystemTime array?
	If $iFormat=0 Then Return $aSysTime

	Local $vReturn=_WinTime_FormatTime($aSysTime[0],$aSysTime[1],$aSysTime[2],$aSysTime[3], _
		$aSysTime[4],$aSysTime[5],$aSysTime[6],$aSysTime[7],$iFormat,$iPrecision,$bAMPMConversion)
	Return SetError(@error,@extended,$vReturn)
EndFunc

Func _WinTime_LocalFileTimeToSystemTime($iLocalFileTime)
	Local $aRet,$stSysTime,$aSysTime[8]=[-1,-1,-1,-1,-1,-1,-1,-1]

	; Negative values unacceptable
	If $iLocalFileTime<0 Then Return SetError(1,0,$aSysTime)

	; SYSTEMTIME structure [Year,Month,DayOfWeek,Day,Hour,Min,Sec,Milliseconds]
	$stSysTime=DllStructCreate("ushort[8]")

	$aRet=DllCall($_COMMON_KERNEL32DLL,"bool","FileTimeToSystemTime","uint64*",$iLocalFileTime,"ptr",DllStructGetPtr($s
	If @error Then Return SetError(2,@error,$aSysTime)
	If Not $aRet[0] Then Return SetError(3,0,$aSysTime)
	Dim $aSysTime[8]=[DllStructGetData($stSysTime,1,1),DllStructGetData($stSysTime,1,2),DllStructGetData($stSysTime,1,4

		DllStructGetData($stSysTime,1,6),DllStructGetData($stSysTime,1,7),DllStructGetData($stSysTime,1,8),DllStruc
	Return $aSysTime
EndFunc

Func _WinTime_FormatTime($iYear,$iMonth,$iDay,$iHour,$iMin,$iSec,$iMilSec,$iDayOfWeek,$iFormat=4,$iPrecision=0,$bAMPMConver
	Local Static $_WT_aMonths[12]=["January","February","March","April","May","June","July","August","September","Octob
	Local Static $_WT_aDays[7]=["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]

	If Not $iFormat Or $iMonth<1 Or $iMonth>12 Or $iDayOfWeek>6 Then Return SetError(1,0,"")

	; Pad MM,DD,HH,MM,SS,MSMSMSMS as necessary
	Local $sMM=StringRight(0&$iMonth,2),$sDD=StringRight(0&$iDay,2),$sMin=StringRight(0&$iMin,2)
	; $sYY = $iYear ; (no padding)
	;	[technically Year can be 1-x chars - but this is generally used for 4-digit years. And SystemTime only goes
	Local $sHH,$sSS,$sMS,$sAMPM

	; 'Extra precision 1': +SS (Seconds)
	If $iPrecision Then
		$sSS=StringRight(0&$iSec,2)
		; 'Extra precision 2': +MSMSMSMS (Milliseconds)
		If $iPrecision>1 Then
;			$sMS=StringRight('000'&$iMilSec,4)
			$sMS=StringRight('000'&$iMilSec,3);Fixed an erronous 0 in front of the milliseconds
		Else
			$sMS=""
		EndIf
	Else
		$sSS=""
		$sMS=""
	EndIf
	If $bAMPMConversion Then
		If $iHour>11 Then
			$sAMPM=" PM"
			; 12 PM will cause 12-12 to equal 0, so avoid the calculation:
			If $iHour=12 Then
				$sHH="12"
			Else
				$sHH=StringRight(0&($iHour-12),2)
			EndIf
		Else
			$sAMPM=" AM"
			If $iHour Then
				$sHH=StringRight(0&$iHour,2)
			Else
			; 00 military = 12 AM
				$sHH="12"
			EndIf
		EndIf
	Else
		$sAMPM=""

		$sHH=StringRight(0 & $iHour,2)
	EndIf

	Local $sDateTimeStr,$aReturnArray[3]

	; Return an array? [formatted string + "Month" + "DayOfWeek"]
	If BitAND($iFormat,0x10) Then
		$aReturnArray[1]=$_WT_aMonths[$iMonth-1]
		If $iDayOfWeek>=0 Then
			$aReturnArray[2]=$_WT_aDays[$iDayOfWeek]
		Else
			$aReturnArray[2]=""
		EndIf
		; Strip the 'array' bit off (array[1] will now indicate if an array is to be returned)
```

```
			$iFormat=BitAND($iFormat,0xF)
		Else
			; Signal to below that the array isn't to be returned
			$aReturnArray[1]=""
		EndIf

		; Prefix with "DayOfWeek "?
		If BitAND($iFormat,8) Then
			If $iDayOfWeek<0 Then Return SetError(1,0,"")	; invalid
			$sDateTimeStr=$_WT_aDays[$iDayOfWeek]&', '
			; Strip the 'DayOfWeek' bit off
			$iFormat=BitAND($iFormat,0x7)
		Else
			$sDateTimeStr=""
		EndIf

		If $iFormat<2 Then
			; Basic String format: YYYYMMDDHHMM[SS[MSMSMSMS[ AM/PM]]]
			$sDateTimeStr&=$iYear&$sMM&$sDD&$sHH&$sMin&$sSS&$sMS&$sAMPM
		Else
			; one of 4 formats which ends with " HH:MM[:SS[:MSMSMSMS[ AM/PM]]]"
			Switch $iFormat
				; /, : Format - MM/DD/YYYY
				Case 2
					$sDateTimeStr&=$sMM&'/'&$sDD&'/'
				; /, : alt. Format - DD/MM/YYYY
				Case 3
					$sDateTimeStr&=$sDD&'/'&$sMM&'/'
				; "Month DD, YYYY" format
				Case 4
					$sDateTimeStr&=$_WT_aMonths[$iMonth-1]&' '&$sDD&', '
				; "DD Month YYYY" format
				Case 5
					$sDateTimeStr&=$sDD&' '&$_WT_aMonths[$iMonth-1]&' '
				Case 6
					$sDateTimeStr&=$iYear&'-'&$sMM&'-'&$sDD

					$iYear=''
				Case Else
					Return SetError(1,0,"")
			EndSwitch
			$sDateTimeStr&=$iYear&' '&$sHH&':'&$sMin
			If $iPrecision Then
				$sDateTimeStr&=':'&$sSS
				If $iPrecision>1 Then $sDateTimeStr&=':'&$sMS
			EndIf
			$sDateTimeStr&=$sAMPM
		EndIf
		If $aReturnArray[1]<>"" Then
			$aReturnArray[0]=$sDateTimeStr
			Return $aReturnArray
		EndIf
		Return $sDateTimeStr
	EndFunc
	; end: by Ascend4nt ----------------------------

	Func _DecodeNameQ($NameQ)
		For $name = 1 To UBound($NameQ) - 1
			$NameString = $NameQ[$name]
			If $NameString = "" Then ContinueLoop
			$FN_AllocSize = Dec(_SwapEndian(StringMid($NameString,129,16)),2)
			$FN_RealSize = Dec(_SwapEndian(StringMid($NameString,145,16)),2)
			$FN_NameLength = Dec(StringMid($NameString,177,2))
			$FN_NameSpace = StringMid($NameString,179,2)
			Select
				Case $FN_NameSpace = '00'
					$FN_NameSpace = 'POSIX'
				Case $FN_NameSpace = '01'
					$FN_NameSpace = 'WIN32'
				Case $FN_NameSpace = '02'
					$FN_NameSpace = 'DOS'
				Case $FN_NameSpace = '03'
					$FN_NameSpace = 'DOS+WIN32'
				Case Else
					$FN_NameSpace = 'UNKNOWN'
```

```
                EndSelect
                $FN_FileName = StringMid($NameString,181,$FN_NameLength*4)
                $FN_FileName = _UnicodeHexToStr($FN_FileName)
                If StringLen($FN_FileName) <> $FN_NameLength Then $INVALID_FILENAME = 1
        Next
        Return
EndFunc

Func _Usage()
        ConsoleWrite("Usage:" & @CRLF)
        ConsoleWrite("RawDir.exe mode path" & @CRLF)
        ConsoleWrite("  mode can be 1 or 2. 1 is verbose output. 2 is more compact output." & @CRLF)

        ConsoleWrite("  path is the path to perform directory listing on." & @CRLF)
        ConsoleWrite("Example printing verbose output on the path C:\tmp" & @CRLF)
        ConsoleWrite("  RawDir.exe 1 C:\tmp" & @CRLF)
        ConsoleWrite("Example printing compact output on the root of the C: volume" & @CRLF)
        ConsoleWrite("  RawDir.exe 2 C:\" & @CRLF)
EndFunc

Func _AlignString($input,$length)
        While 1
                If StringLen($input)=$length Then ExitLoop
                $input = " "&$input
        WEnd
        Return $input
EndFunc
```