



# STARPRINT LIMITED

## Source code exported to PDF

<b>Project Name</b>	VBZip 1.0.6
<b>Student Name</b>	Joginder S Nahil
<b>Lecturer's Name</b>	<b>Dr. Aaron Mann</b>
<b>Short Project Filename</b>	pVBZip6.vbp
<b>General Date</b>	07/10/2007 12:31:16
<b>Long Date</b>	07 October 2007
<b>Short Date</b>	07/10/2007
<b>Long Time</b>	12:31:16
<b>Short Time</b>	12:31

```

00001 Option Explicit
00002
00003 Private WithEvents m_cZ As cZip

```

---

```

00004
00005 Private Sub cmdRecurse_Click()
00006     With m_cZ
00007         .ZipFile = App.Path & "\Test_Rec.zip"
00008         .Encrypt = (chkEncrypt.Value = vbChecked)
00009         .AddComment = (chkAddComment.Value = vbChecked)
00010         .BasePath = App.Path
00011         .ClearFileSpecs
00012         .AddFileSpec "*.fr*"
00013         .AddFileSpec "*.cls"
00014         .AddFileSpec "*.bas"
00015         .StoreFolderNames = True
00016         .RecurseSubDirs = True
00017         .Zip
00018
00019         If (.Success) Then
00020             MsgBox "Zipped files." _
00021                 & vbCrLf & vbCrLf & _
00022                 "   Source: files matching *.fr*;*.cls;*.bas from " & .BasePath & vbCrLf & _
00023                 "   To: " & .ZipFile, vbInformation
00024         Else
00025             MsgBox "Zipping failed.", vbExclamation
00026         End If
00027     End With
00028 End Sub

```

---

```

00029
00030 Private Sub cmdSingleFile_Click()
00031     Dim cc As New GCommonDialog
00032     Dim sFile As String
00033
00034     ' Get the file to zip:
00035     If (cc.VBGetOpenFileName(sFile, , , , , "All Files (*.*)|*.*", , , "Choose File to
> Zip", , Me.hwnd)) Then
00036         With m_cZ
00037             .Encrypt = (chkEncrypt.Value = vbChecked)
00038             .AddComment = (chkAddComment.Value = vbChecked)
00039             .ZipFile = App.Path & "\Test_One.zip"
00040             .StoreFolderNames = False
00041             .RecurseSubDirs = False
00042             .ClearFileSpecs
00043             .AddFileSpec sFile
00044             .Zip
00045
00046             If (.Success) Then
00047                 MsgBox "Zipped one file." _
00048                     & vbCrLf & vbCrLf & _
00049                     "   Source: " & .FileSpec(1) & vbCrLf & _
00050                     "   To: " & .ZipFile, vbInformation
00051             Else
00052                 MsgBox "Zip Failed.", vbExclamation
00053             End If
00054         End With
00055     End If
00056 End Sub
00057

```

---

```

00058
00059 Private Sub Form_Load()
00060     Set m_cZ = New cZip
00061 End Sub

```

---

```

00062
00063 Private Sub m_cZ_CommentRequest(sComment As String, bCancel As Boolean)
00064     '
00065     Dim sComm As String
00066     sComm = InputBox("Enter comment:", App.EXENAME)
00067     sComm = Trim(sComm)
00068     If Len(sComm) = 0 Then
00069         bCancel = True
1 2

```

```
00070 1 2 { Else
00071      sComment = sComm
00072  } End If
00073      '
00074  End Sub

00075
00076  Private Sub m_cZ_PasswordRequest(sPassword As String, ByVal lMaxPasswordLength As Long,
»   ByVal bConfirm As Boolean, bCancel As Boolean)
00077      '
00078      Dim sPass As String
00079      Dim sMsg As String
00080      If (bConfirm) Then
00081          sMsg = "Confirm password:"
00082      Else
00083          sMsg = "Enter password:"
00084      End If
00085      sPass = InputBox(sMsg, App.EXENAME)
00086      sPass = Trim(sPass)
00087      If (Len(sPass) = 0) Then
00088          bCancel = True
00089      Else
00090          sPassword = sPass
00091      End If
00092      '
00093  End Sub

00094
00095
00096  Private Sub m_cZ_Progress(ByVal lCount As Long, ByVal sMsg As String)
00097      sbrMain.Panels(1).Text = sMsg
00098      lstLog.AddItem sMsg
00099      lstLog.ListIndex = lstLog.NewIndex
00100  End Sub
```

```

00101 Option Explicit
00102
00103 '
» =====

00104 ' Name:      mzip
00105 ' Author:   Steve McMahon (steve@vbaccelerator.com)
00106 ' Date:    1 January 2000
00107 '
00108 ' Requires: Info -Zip 's Zip32.DLL v2.32, renamed
00109 '           - Without encryption, this DLL is renamed to vbzip10.dll
00110 '           - With encryption, this DLL is renamed to vbzip11.dll
00111 '           This code is written for vbzip11.dll.
00112 '           cUnzip.cls
00113 '
00114 ' Copyright © 2000-2003 Steve McMahon for vbAccelerator
00115 '
» -----

00116 ' Visit vbAccelerator - advanced free source code for VB programmers
00117 ' http://vbaccelerator.com
00118 '
» -----

00119 '
00120 ' Part of the implementation of cUnzip.cls, a class which gives a
00121 ' simple interface to Info-ZIP's excellent, free zipping library
00122 ' (Zip32.DLL).
00123 '
00124 ' This sample uses decompression code by the Info-ZIP group. The
00125 ' original Info-Zip sources are freely available from their website
00126 ' at
00127 '   http://www.cdrcom.com/pubs/infozip/
00128 '
00129 ' Please ensure you visit the site and read their free source licensing
00130 ' information and requirements before using their code in your own
00131 ' application.
00132 '
00133 '
» =====

00134
00135
00136 Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
00137     lpvDest As Any, lpvSource As Any, ByVal cbCopy As Long)
00138
00139 ' argv
00140 Private Type ZIPnames
00141     s(0 To 1023) As String
00142 End Type
00143
00144 ' Callback large "string" (sic)
00145 Private Type CBChar
00146     ch(0 To 4096) As Byte
00147 End Type
00148
00149 ' Callback small "string" (sic)
00150 Private Type CBCh
00151     ch(0 To 255) As Byte
00152 End Type
00153
00154 ' Store the callback functions
00155 Private Type ZIPUSERFUNCTIONS
00156     lpPtrPrint As Long           ' Pointer to application's print routine
00157     lpPtrComment As Long        ' Pointer to application's comment routine
00158     lpPtrPassword As Long       ' Pointer to application's password routine.
00159     lpPtrService As Long        ' callback function designed to be used for allowing the
00160     ' app to process Windows messages, or cancelling the operation
00161     ' as well as giving option of progress. If this function returns
00162     ' non-zero, it will terminate what it is doing. It provides the app
00163     ' with the name of the archive member it has just processed, as well
00164     ' as the original size.

```

```

00165 End Type
00166
00167 Public Type ZPOPT
00168     Date As String ' US Date (8 Bytes Long) "12/31/98"?
00169     szRootDir As String ' Root Directory Pathname (Up To 256 Bytes Long)
00170     szTempDir As String ' Temp Directory Pathname (Up To 256 Bytes Long)
00171     fTemp As Long ' 1 If Temp dir Wanted, Else 0
00172     fSuffix As Long ' Include Suffixes (Not Yet Implemented!)
00173     fEncrypt As Long ' 1 If Encryption Wanted, Else 0
00174     fSystem As Long ' 1 To Include System/Hidden Files, Else 0
00175     fVolume As Long ' 1 If Storing Volume Label, Else 0
00176     fExtra As Long ' 1 If Excluding Extra Attributes, Else 0
00177     fNoDirEntries As Long ' 1 If Ignoring Directory Entries, Else 0
00178     fExcludeDate As Long ' 1 If Excluding Files Earlier Than Specified Date, Else 0
00179     fIncludeDate As Long ' 1 If Including Files Earlier Than Specified Date, Else 0
00180     fVerbose As Long ' 1 If Full Messages Wanted, Else 0
00181     fQuiet As Long ' 1 If Minimum Messages Wanted, Else 0
00182     fCRLF_LF As Long ' 1 If Translate CR/LF To LF, Else 0
00183     fLF_CRLF As Long ' 1 If Translate LF To CR/LF, Else 0
00184     fJunkDir As Long ' 1 If Junking Directory Names, Else 0
00185     fGrow As Long ' 1 If Allow Appending To Zip File, Else 0
00186     fForce As Long ' 1 If Making Entries Using DOS File Names, Else 0
00187     fMove As Long ' 1 If Deleting Files Added Or Updated, Else 0
00188     fDeleteEntries As Long ' 1 If Files Passed Have To Be Deleted, Else 0
00189     fUpdate As Long ' 1 If Updating Zip File-Overwrite Only If Newer, Else 0
00190     fFreshen As Long ' 1 If Freshing Zip File-Overwrite Only, Else 0
00191     fJunkSFX As Long ' 1 If Junking SFX Prefix, Else 0
00192     fLatestTime As Long ' 1 If Setting Zip File Time To Time Of Latest File In
    » Archive, Else 0
00193     fComment As Long ' 1 If Putting Comment In Zip File, Else 0
00194     fOffsets As Long ' 1 If Updating Archive Offsets For SFX Files, Else 0
00195     fPrivilege As Long ' 1 If Not Saving Privileges, Else 0
00196     fEncryption As Long ' Read Only Property!!!
00197     fRecurse As Long ' 1 (-r), 2 (-R) If Recursing Into Sub-Directories, Else 0
00198     fRepair As Long ' 1 = Fix Archive, 2 = Try Harder To Fix, Else 0
00199     flevel As Byte ' Compression Level - 0 = Stored 6 = Default 9 = Max
00200 End Type
00201
00202 'This assumes zip32.dll is in your \windows\system directory!
00203 Private Declare Function ZpInit Lib "vbzip11.dll" (ByRef tUserFn As ZIPUSERFUNCTIONS)
    » As Long ' Set Zip Callbacks
00204 Private Declare Function ZpSetOptions Lib "vbzip11.dll" (ByRef tOpts As ZPOPT) As Long
    » ' Set Zip options
00205 Private Declare Function ZpGetOptions Lib "vbzip11.dll" () As ZPOPT ' used to check
    » encryption flag only
00206 Private Declare Function ZpArchive Lib "vbzip11.dll" (ByVal argc As Long, ByVal funame
    » As String, ByRef argv As ZIPnames) As Long ' Real zipping action
00207
00208 ' Object for callbacks:
00209 Private m_cZip As cZip
00210 Private m_bCancel As Boolean
00211 Private m_bPasswordConfirm As Boolean
00212
00213 Private Function pAddressOf(ByVal lPtr As Long) As Long
    ' VB Bug workaround fn
00214     pAddressOf = lPtr
00215 End Function
00216
00217
00218 Public Function VBZip( _
00219     cZipObject As cZip, _
00220     tZPOPT As ZPOPT, _
00221     sFileSpecs() As String, _
00222     iFileCount As Long _
00223 ) As Long
00224     Dim tUser As ZIPUSERFUNCTIONS
00225     Dim lR As Long
00226     Dim i As Long
00227     Dim sZipFile As String
00228     Dim tZipName As ZIPnames
00229
00230     m_bCancel = False

```

```

00231     m_bPasswordConfirm = False
00232     Set m_cZip = cZipObject
00233
00234     { If Not Len(Trim$(m_cZip.BasePath)) = 0 Then
00235         ChDir m_cZip.BasePath
00236     End If
00237
00238     ' Set address of callback functions
00239     tUser.lPtrPrint = plAddressOf(AddressOf ZipPrintCallback)
00240     tUser.lPtrPassword = plAddressOf(AddressOf ZipPasswordCallback)
00241     tUser.lPtrComment = plAddressOf(AddressOf ZipCommentCallback)
00242     tUser.lPtrService = plAddressOf(AddressOf ZipServiceCallback)
00243     lR = ZpInit(tUser)
00244
00245     ' Set options
00246     lR = ZpSetOptions(tZPOPT)
00247
00248     ' Go for it!
00249     { For i = 1 To iFileCount
00250         tZipName.s(i - 1) = sFileSpecs(i)
00251     Next i
00252     tZipName.s(i) = vbNullChar
00253
00254     sZipFile = cZipObject.ZipFile
00255     lR = ZpArchive(iFileCount, sZipFile, tZipName)
00256
00257     Debug.Print lR
00258     VBZip = lR
00259
00260 End Function

```

```

00261
00262 Private Function ZipServiceCallback( _
00263     ByRef mname As CChar, _
00264     ByVal x As Long _
00265 ) As Long
00266     Dim iPos As Long
00267     Dim sInfo As String
00268     Dim bCancel As Boolean
00269
00270     '-- Always Put This In Callback Routines!
00271     On Error Resume Next
00272
00273     ' Check we've got a message:
00274     { If x > 1 And x < 1024 Then
00275         ' If so, then get the readable portion of it:
00276         ReDim b(0 To x) As Byte
00277         CopyMemory b(0), mname, x
00278         ' Convert to VB string:
00279         sInfo = StrConv(b, vbUnicode)
00280         iPos = InStr(sInfo, vbNullChar)
00281         { If iPos > 0 Then
00282             sInfo = Left$(sInfo, iPos - 1)
00283         End If
00284         m_cZip.Service sInfo, bCancel
00285         { If bCancel Then
00286             m_bCancel = True
00287             ZipServiceCallback = 1
00288         Else
00289             ZipServiceCallback = 0
00290         End If
00291     End If
00292
00293 End Function

```

```

00294
00295 Private Function ZipPrintCallback( _
00296     ByRef fname As CChar, _
00297     ByVal x As Long _
00298 ) As Long
00299     Dim iPos As Long
00300     Dim sFile As String
00301     On Error Resume Next

```

```

00302
00303 ' Check we've got a message:
00304 If x > 1 And x < 1024 Then
00305 ' If so, then get the readable portion of it:
00306 ReDim b(0 To x) As Byte
00307 CopyMemory b(0), fname, x
00308 ' Convert to VB string:
00309 sFile = StrConv(b, vbUnicode)
00310 {
00311     sFile = Left$(sFile, iPos - 1)
00312 }
00313 End If
00314 ' Fix up backslashes:
00315 ReplaceSection sFile, "/", "\"
00316
00317 ' Tell the caller about it
00318 m_cZip.ProgressReport sFile
00319 End If
00320 ZipPrintCallback = 0
00321
00322 End Function

```

```

00323
00324 Private Function ZipCommentCallback( _
00325     ByRef comm As CbChar _
00326 ) As Integer
00327 Dim bCancel As Boolean
00328 Dim sComment As String
00329 Dim b() As Byte
00330 Dim lSize As Long
00331 Dim i As Long
00332
00333 On Error Resume Next
00334 ' not supported always return \0
00335 comm.ch(0) = vbNullString
00336 ZipCommentCallback = 1
00337
00338 {
00339     If m_bCancel Then
00340         Exit Function
00341     }
00342 End If
00343
00344 ' Ask for a comment:
00345 m_cZip.CommentRequest sComment, bCancel
00346
00347 sComment = Trim$(sComment)
00348
00349 ' Cancel out if no useful comment:
00350 {
00351     If bCancel Or Len(sComment) = 0 Then
00352         m_bCancel = True
00353         Exit Function
00354     }
00355 End If
00356
00357 ' Put password into return parameter:
00358 lSize = Len(sComment)
00359 b = StrConv(sComment, vbFromUnicode)
00360 i = 0
00361 lSize = UBound(b)
00362 If (lSize > 254) Then lSize = 254
00363 For i = 0 To lSize
00364     comm.ch(i) = b(i)
00365 }
00366 Next i
00367 comm.ch(lSize + 1) = 0
00368 ZipCommentCallback = 0
00369
00370 End Function

```

```

00366
00367 Private Function ZipPasswordCallback( _
00368     ByRef pwd As CbCh, _
00369     ByVal maxPasswordLength As Long, _
00370     ByRef s2 As CbCh, _
00371     ByRef Name As CbCh _
00372 ) As Integer

```

```

00373
00374     Dim bCancel As Boolean
00375     Dim sPassword As String
00376     Dim b() As Byte
00377     Dim lSize As Long
00378     Dim i As Long
00379
00380     On Error Resume Next
00381
00382     ' The default:
00383     ZipPasswordCallback = 1
00384
00385     If m_bCancel Then
00386         Exit Function
00387     End If
00388
00389     ' Ask for password:
00390     m_cZip.PasswordRequest sPassword, maxPasswordLength, m_bPasswordConfirm, bCancel
00391
00392     sPassword = Trim$(sPassword)
00393
00394     ' Cancel out if no useful password:
00395     If bCancel Or Len(sPassword) = 0 Then
00396         m_bCancel = True
00397         Exit Function
00398     End If
00399
00400     ' Put password into return parameter:
00401     lSize = Len(sPassword)
00402     b = StrConv(sPassword, vbFromUnicode)
00403     For i = 0 To maxPasswordLength - 1
00404         If (i <= UBound(b)) Then
00405             pwd.ch(i) = b(i)
00406         Else
00407             pwd.ch(i) = 0
00408         End If
00409     Next i
00410
00411     ' Ask UnZip to process it:
00412     m_bPasswordConfirm = True
00413     ZipPasswordCallback = 0
00414
00415 End Function

```

---

```

00416
00417 Private Function ReplaceSection(ByRef sString As String, ByVal sToReplace As String,
»   ByVal sReplaceWith As String) As Long
00418     Dim iPos As Long
00419     Dim iLastPos As Long
00420     iLastPos = 1
00421     Do
00422         iPos = InStr(iLastPos, sString, "/" )
00423         If (iPos > 1) Then
00424             Mid$(sString, iPos, 1) = "\"
00425             iLastPos = iPos + 1
00426         End If
00427     Loop While Not (iPos = 0)
00428     ReplaceSection = iLastPos
00429
00430 End Function

```

```

00431 Option Explicit
00432
00433 '
» =====

00434 ' Name:      vbAccelerator cUnzip class
00435 ' Author:    Steve McMahon (steve@vbaccelerator.com)
00436 ' Date:      1 January 2000
00437 '
00438 ' Requires:  Info-ZIP's Zip32.DLL v2.32, renamed to vbzip10.dll
00439 '            mZip.bas
00440 '
00441 ' Copyright © 2000 Steve McMahon for vbAccelerator
00442 '
» -----

00443 ' Visit vbAccelerator - advanced free source code for VB programmers
00444 ' http://vbaccelerator.com
00445 '
» -----

00446 '
00447 ' Part of the implementation of cUnzip.cls, a class which gives a
00448 ' simple interface to Info-ZIP's excellent, free zipping library
00449 ' (Zip32.DLL).
00450 '
00451 '
» =====

00452
00453
00454 Public Enum EZPMsgLevel
00455     ezpAllMessages = 0
00456     ezpPartialMessages = 1
00457     ezpNoMessages = 2
00458 End Enum
00459
00460 Public Event Cancel(ByVal sMsg As String, ByRef bCancel As Boolean)
00461 Public Event PasswordRequest(ByRef sPassword As String, ByVal lMaxPasswordLength As
» Long, ByVal bConfirm As Boolean, ByRef bCancel As Boolean)
00462 Public Event CommentRequest(ByRef sComment As String, ByRef bCancel As Boolean)
00463 Public Event Progress(ByVal lCount As Long, ByVal sMsg As String)
00464
00465
00466 Private m_tZPOPT As ZPOPT
00467 Private m_sFileName As String
00468 Private m_sFileSpecs() As String
00469 Private m_iCount As Long
00470 Private m_lR As Long

```

---

```

00471
00472 ' Set zip options
00473 ' m_tZPOPT.fSuffix = 0           ' include suffixes (not yet implemented)
00474 ' m_tZPOPT.fExtra = 0           ' 1 if including extra attributes
00475 ' m_tZPOPT.date = vbNullString ' "12/31/79"? US Date?
00476 ' m_tZPOPT.fExcludeDate = 0    ' 1 if excluding files earlier than a specified date
00477 ' m_tZPOPT.fIncludeDate = 0    ' 1 if including files earlier than a specified date
00478 ' m_tZPOPT.fJunkSFX = 0        ' 1 if junking sfx prefix
00479 ' m_tZPOPT.fOffsets = 0        ' 1 if updating archive offsets for sfx Files
00480
00481 ' m_tZPOPT.fComment = 0        ' 1 if putting comment in zip file
00482
00483 ' m_tZPOPT.fGrow = 0           ' 1 if allow appending to zip file
00484 ' m_tZPOPT.fForce = 0          ' 1 if making entries using DOS names
00485 ' m_tZPOPT.fMove = 0           ' 1 if deleting files added or updated
00486 ' m_tZPOPT.fDeleteEntries = 0 ' 1 if files passed have to be deleted
00487 ' m_tZPOPT.fLatestTime = 0    ' 1 if setting zip file time to time of latest file in
» archive
00488 ' m_tZPOPT.fPrivilege = 0      ' 1 if not saving privileges
00489 ' m_tZPOPT.fEncryption = 0    ' Read only property!
00490 ' m_tZPOPT.fRepair = 0        ' 1=> fix archive, 2=> try harder to fix
00491 ' m_tZPOPT.flevel = 0         ' compression level - should be 0!!!

```

```

00492
00493
00494
00495 { Public Property Get ZipFile() As String
00496     ZipFile = m_sFileName
00497 End Property
-----
00498 { Public Property Let ZipFile(ByVal sFileName As String)
00499     m_sFileName = sFileName
00500 End Property
-----
00501 { Public Property Get BasePath() As String
00502     BasePath = m_tZPOPT.szRootDir
00503 End Property
-----
00504 { Public Property Let BasePath(ByVal sBasePath As String)
00505     m_tZPOPT.szRootDir = sBasePath
00506 End Property
-----
00507 { Public Property Get Encrypt() As Boolean
00508     Encrypt = Not (m_tZPOPT.fEncrypt = 0)
00509 End Property
-----
00510 { Public Property Let Encrypt(ByVal bState As Boolean)
00511     m_tZPOPT.fEncrypt = Abs(bState)
00512 End Property
-----
00513 { Public Property Get AddComment() As Boolean
00514     Encrypt = Not (m_tZPOPT.fComment = 0)
00515 End Property
-----
00516 { Public Property Let AddComment(ByVal bState As Boolean)
00517     m_tZPOPT.fComment = Abs(bState)
00518 End Property
-----
00519 { Public Property Get IncludeSystemAndHiddenFiles() As Boolean
00520     IncludeSystemAndHiddenFiles = Not (m_tZPOPT.fSystem = 0) ' 1 to include
    system/hidden files
00521 End Property
-----
00522 { Public Property Let IncludeSystemAndHiddenFiles(ByVal bState As Boolean)
00523     m_tZPOPT.fSystem = Abs(bState) ' 1 to include system/hidden files
00524 End Property
-----
00525 { Public Property Get StoreVolumeLabel() As Boolean
00526     StoreVolumeLabel = Not (m_tZPOPT.fVolume = 0) ' 1 if storing volume label
00527 End Property
-----
00528 { Public Property Let StoreVolumeLabel(ByVal bState As Boolean)
00529     m_tZPOPT.fVolume = Abs(bState)
00530 End Property
-----
00531 { Public Property Get StoreDirectories() As Boolean
00532     StoreDirectories = Not (m_tZPOPT.fNoDirEntries = 0) ' 1 if ignoring directory entries
00533 End Property
-----
00534 { Public Property Let StoreDirectories(ByVal bState As Boolean)
00535     m_tZPOPT.fNoDirEntries = Abs(Not (bState))
00536 End Property
-----
00537 { Public Property Get StoreFolderNames() As Boolean
00538     StoreFolderNames = (m_tZPOPT.fJunkDir = 0)
00539 End Property
-----
00540 { Public Property Let StoreFolderNames(ByVal bState As Boolean)
00541     m_tZPOPT.fJunkDir = Abs(Not (bState))
00542 End Property
-----
00543 { Public Property Get RecurseSubDirs() As Boolean
00544     RecurseSubDirs = Not (m_tZPOPT.fRecurse = 0) ' 1 if recursing into subdirectories
00545 End Property
-----
00546 { Public Property Let RecurseSubDirs(ByVal bState As Boolean)
00547     If bState Then
00548         m_tZPOPT.fRecurse = 2
00549     Else
00550         m_tZPOPT.fRecurse = 0
00551     End If

```

```

00552 } End Property
00553
00554 { Public Property Get UpdateOnlyIfNewer() As Boolean
00555     UpdateOnlyIfNewer = Not (m_tZPOPT.fUpdate = 0) ' 1 if updating zip file--overwrite
00556     } only if newer
00557 } End Property
00558 { Public Property Let UpdateOnlyIfNewer(ByVal bState As Boolean)
00559     m_tZPOPT.fUpdate = Abs(bState) ' 1 if updating zip file--overwrite only if newer
00560 } End Property
00561 { Public Property Get FreshenFiles() As Boolean
00562     FreshenFiles = Not (m_tZPOPT.fFreshen = 0) ' 1 if freshening zip file--overwrite only
00563 } End Property
00564 { Public Property Let FreshenFiles(ByVal bState As Boolean)
00565     m_tZPOPT.fUpdate = Abs(bState) ' 1 if updating zip file--overwrite only if newer
00566 } End Property
00567 { Public Property Get MessageLevel() As EZPMsgLevel
00568     If Not (m_tZPOPT.fVerbose = 0) Then
00569         MessageLevel = ezpAllMessages
00570     ElseIf Not (m_tZPOPT.fQuiet = 0) Then
00571         MessageLevel = ezpPartialMessages
00572     Else
00573         MessageLevel = ezpNoMessages
00574     End If
00575 } End Property
00576 { Public Property Let MessageLevel(ByVal eLevel As EZPMsgLevel)
00577     Select Case eLevel
00578     Case ezpPartialMessages
00579         m_tZPOPT.fQuiet = 1
00580         m_tZPOPT.fVerbose = 0
00581     Case ezpNoMessages
00582         m_tZPOPT.fQuiet = 0
00583         m_tZPOPT.fVerbose = 0
00584     Case ezpAllMessages
00585         m_tZPOPT.fQuiet = 0
00586         m_tZPOPT.fVerbose = 1
00587     End Select
00588 } End Property
00589 { Public Property Get ConvertCRLFtoLF() As Boolean
00590     ConvertCRLFtoLF = (m_tZPOPT.fCRLF_LF <> 0)
00591 } End Property
00592 { Public Property Let ConvertCRLFtoLF(ByVal bState As Boolean)
00593     m_tZPOPT.fCRLF_LF = Abs(bState)
00594 } End Property
00595 { Public Property Get ConvertLFtoCRLF() As Boolean
00596     ConvertLFtoCRLF = (m_tZPOPT.fLF_CRLF <> 0)
00597 } End Property
00598 { Public Property Let ConvertLFtoCRLF(ByVal bState As Boolean)
00599     m_tZPOPT.fLF_CRLF = Abs(bState)
00600 } End Property
00601 { Friend Sub ProgressReport( _
00602     ByVal sMsg As String _
00603     )
00604     RaiseEvent Progress(1, sMsg)
00605 } End Sub
00606 { Friend Sub PasswordRequest( _
00607     ByRef sPassword As String, _
00608     ByVal lMaxPasswordLength As Long, _
00609     ByVal bPasswordConfirm As Boolean, _
00610     ByRef bCancel As Boolean _
00611     )
00612     RaiseEvent PasswordRequest(sPassword, lMaxPasswordLength, bPasswordConfirm, bCancel)
00613 } End Sub

```

```
00614 Friend Sub CommentRequest( _
00615     ByRef sComment As String, _
00616     ByRef bCancel As Boolean _
00617 )
00618     RaiseEvent CommentRequest(sComment, bCancel)
00619 End Sub

00620 Friend Sub Service( _
00621     ByVal sMsg As String, _
00622     ByRef bCancel As Boolean _
00623 )
00624     RaiseEvent Cancel(sMsg, bCancel)
00625 End Sub

00626 Public Sub ClearFileSpecs()
00627     m_iCount = 0
00628     Erase m_sFileSpecs()
00629 End Sub

00630 Public Function AddFileSpec(ByVal sSpec As String) As Long
00631     m_iCount = m_iCount + 1
00632     ReDim Preserve m_sFileSpecs(1 To m_iCount) As String
00633     m_sFileSpecs(m_iCount) = sSpec
00634 End Function

00635 Public Property Get FileSpecCount() As Long
00636     FileSpecCount = m_iCount
00637 End Property

00638 Public Property Get FileSpec(ByVal nIndex As Long)
00639     FileSpec = m_sFileSpecs(nIndex)
00640 End Property

00641 Public Property Get AllowAppend() As Boolean
00642     AllowAppend = (m_tZPOPT.fGrow = 1)
00643 End Property

00644 Public Property Let AllowAppend(ByVal bState As Boolean)
00645     m_tZPOPT.fGrow = Abs(bState)
00646 End Property

00647 Public Sub Zip()
00648     m_lR = 0
00649     m_lR = mZip.VBZip(Me, m_tZPOPT, m_sFileSpecs(), m_iCount)
00650 End Sub

00651 Public Property Get Success() As Boolean
00652     Success = (m_lR = 0)
00653 End Property

00654 Public Sub Delete()
00655     ' Deletes the entries specified by the file specs:
00656     m_tZPOPT.fDeleteEntries = 1
00657     mZip.VBZip Me, m_tZPOPT, m_sFileSpecs(), m_iCount
00658     m_tZPOPT.fDeleteEntries = 0
00659 End Sub

00660
00661 Private Sub Class_Initialize()
00662     StoreDirectories = False
00663     StoreFolderNames = False
00664     RecurseSubDirs = False
00665 End Sub
```

```

00666 Option Explicit
00667
00668 ' =====
00669 ' Class:      GCommonDialog
00670 ' Filename:  GCommonDialog.cls
00671 ' Author:    Steve McMahon, based on original by Bruce McKinney
00672 ' Date:      24 May 1998
00673 ' =====
00674
00675
00676 ' =====
00677 ' API declares:
00678 ' =====
00679 Public Enum EErrorCommonDialog
00680     eeBaseCommonDialog = 13450 ' CommonDialog
00681 End Enum
00682
00683 Private Declare Function lstrlen Lib "kernel32" Alias "lstrlenA" (ByVal lpString As
» String) As Long
00684 Private Declare Function GlobalAlloc Lib "kernel32" (ByVal wFlags As Long, ByVal
» dwBytes As Long) As Long
00685 Private Declare Function GlobalCompact Lib "kernel32" (ByVal dwMinFree As Long) As Long
00686
00686 Private Declare Function GlobalFree Lib "kernel32" (ByVal hMem As Long) As Long
00687 Private Declare Function GlobalLock Lib "kernel32" (ByVal hMem As Long) As Long
00688 Private Declare Function GlobalReAlloc Lib "kernel32" (ByVal hMem As Long, ByVal
» dwBytes As Long, ByVal wFlags As Long) As Long
00689 Private Declare Function GlobalSize Lib "kernel32" (ByVal hMem As Long) As Long
00690 Private Declare Function GlobalUnlock Lib "kernel32" (ByVal hMem As Long) As Long
00691 Private Declare Sub CopyMemoryStr Lib "kernel32" Alias "RtlMoveMemory" ( _
00692     lpvDest As Any, ByVal lpvSource As String, ByVal cbCopy As Long)
00693
00694 Private Const MAX_PATH = 260
00695 Private Const MAX_FILE = 260
00696
00697 Private Type OPENFILENAME
00698     lStructSize As Long           ' Filled with UDT size
00699     hWndOwner As Long            ' Tied to Owner
00700     hInstance As Long           ' Ignored (used only by templates)
00701     lpstrFilter As String        ' Tied to Filter
00702     lpstrCustomFilter As String  ' Ignored (exercise for reader)
00703     nMaxCustFilter As Long       ' Ignored (exercise for reader)
00704     nFilterIndex As Long        ' Tied to FilterIndex
00705     lpstrFile As String         ' Tied to FileName
00706     nMaxFile As Long            ' Handled internally
00707     lpstrFileName As String     ' Tied to FileName
00708     nMaxFileName As Long       ' Handled internally
00709     lpstrInitialDir As String   ' Tied to InitDir
00710     lpstrTitle As String        ' Tied to DlgTitle
00711     flags As Long              ' Tied to Flags
00712     nFileOffset As Integer      ' Ignored (exercise for reader)
00713     nFileExtension As Integer   ' Ignored (exercise for reader)
00714     lpstrDefExt As String       ' Tied to DefaultExt
00715     lCustData As Long          ' Ignored (needed for hooks)
00716     lpfnHook As Long           ' Ignored (good luck with hooks)
00717     lpTemplateName As Long     ' Ignored (good luck with templates)
00718 End Type
00719
00720 Private Declare Function GetOpenFileName Lib "COMDLG32" _
00721     Alias "GetOpenFileNameA" (file As OPENFILENAME) As Long
00722 Private Declare Function GetSaveFileName Lib "COMDLG32" _
00723     Alias "GetSaveFileNameA" (file As OPENFILENAME) As Long
00724 Private Declare Function GetFileTitle Lib "COMDLG32" _
00725     Alias "GetFileTitleA" (ByVal szFile As String, _
00726     ByVal szTitle As String, ByVal cbBuf As Long) As Long
00727
00728 Public Enum EOpenFile
00729     OFN_READONLY = &H1
00730     OFN_OVERWRITEPROMPT = &H2
00731     OFN_HIDEREADONLY = &H4
00732     OFN_NOCHANGEDIR = &H8
00733     OFN_SHOWHELP = &H10

```

```

00734     OFN_ENABLEHOOK = &H20
00735     OFN_ENABLETEMPLATE = &H40
00736     OFN_ENABLETEMPLATEHANDLE = &H80
00737     OFN_NOVALIDATE = &H100
00738     OFN_ALLOWMULTISELECT = &H200
00739     OFN_EXTENSIONDIFFERENT = &H400
00740     OFN_PATHMUSTEXIST = &H800
00741     OFN_FILEMUSTEXIST = &H1000
00742     OFN_CREATEPROMPT = &H2000
00743     OFN_SHAREAWARE = &H4000
00744     OFN_NOREADONLYRETURN = &H8000
00745     OFN_NOTESTFILECREATE = &H10000
00746     OFN_NONETWORKBUTTON = &H20000
00747     OFN_NOLONGNAMES = &H40000
00748     OFN_EXPLORER = &H80000
00749     OFN_NODEREFERENCELINKS = &H100000
00750     OFN_LONGNAMES = &H200000
00751 End Enum
00752
00753 Private Type TCHOOSECOLOR
00754     lStructSize As Long
00755     hWndOwner As Long
00756     hInstance As Long
00757     rgbResult As Long
00758     lpCustColors As Long
00759     flags As Long
00760     lCustData As Long
00761     lpfnHook As Long
00762     lpTemplateName As Long
00763 End Type
00764
00765 Private Declare Function ChooseColor Lib "COMDLG32.DLL" _
00766     Alias "ChooseColorA" (Color As TCHOOSECOLOR) As Long
00767
00768 Public Enum EChooseColor
00769     CC_RGBInit = &H1
00770     CC_FullOpen = &H2
00771     CC_PreventFullOpen = &H4
00772     CC_ColorShowHelp = &H8
00773     ' Win95 only
00774     CC_SolidColor = &H80
00775     CC_AnyColor = &H100
00776     ' End Win95 only
00777     CC_ENABLEHOOK = &H10
00778     CC_ENABLETEMPLATE = &H20
00779     CC_EnableTemplateHandle = &H40
00780 End Enum
00781 Private Declare Function GetSysColor Lib "user32" (ByVal nIndex As Long) As Long
00782
00783 Private Type TCHOOSEFONT
00784     lStructSize As Long           ' Filled with UDT size
00785     hWndOwner As Long           ' Caller's window handle
00786     hdc As Long                 ' Printer DC/IC or NULL
00787     lpLogFont As Long           ' Pointer to LOGFONT
00788     iPointSize As Long          ' 10 * size in points of font
00789     flags As Long               ' Type flags
00790     rgbColors As Long           ' Returned text color
00791     lCustData As Long           ' Data passed to hook function
00792     lpfnHook As Long           ' Pointer to hook function
00793     lpTemplateName As Long      ' Custom template name
00794     hInstance As Long           ' Instance handle for template
00795     lpszStyle As String         ' Return style field
00796     nFontType As Integer        ' Font type bits
00797     iAlign As Integer           ' Filler
00798     nSizeMin As Long            ' Minimum point size allowed
00799     nSizeMax As Long            ' Maximum point size allowed
00800 End Type
00801 Private Declare Function ChooseFont Lib "COMDLG32" _
00802     Alias "ChooseFontA" (chfont As TCHOOSEFONT) As Long
00803
00804 Private Const LF_FACESIZE = 32
00805 Private Type LOGFONT

```

```

00806 | lfHeight As Long
00807 | lfWidth As Long
00808 | lfEscapement As Long
00809 | lfOrientation As Long
00810 | lfWeight As Long
00811 | lfItalic As Byte
00812 | lfUnderline As Byte
00813 | lfStrikeOut As Byte
00814 | lfCharSet As Byte
00815 | lfOutPrecision As Byte
00816 | lfClipPrecision As Byte
00817 | lfQuality As Byte
00818 | lfPitchAndFamily As Byte
00819 | lfFaceName(LF_FACESIZE) As Byte
00820 | End Type
00821 |
00822 | Public Enum EChooseFont
00823 |     CF_ScreenFonts = &H1
00824 |     CF_PrinterFonts = &H2
00825 |     CF_BOTH = &H3
00826 |     CF_FontShowHelp = &H4
00827 |     CF_UseStyle = &H80
00828 |     CF_EFFECTS = &H100
00829 |     CF_AnsiOnly = &H400
00830 |     CF_NoVectorFonts = &H800
00831 |     CF_NoOemFonts = CF_NoVectorFonts
00832 |     CF_NoSimulations = &H1000
00833 |     CF_LimitSize = &H2000
00834 |     CF_FixedPitchOnly = &H4000
00835 |     CF_WYSIWYG = &H8000 ' Must also have ScreenFonts And PrinterFonts
00836 |     CF_ForceFontExist = &H10000
00837 |     CF_ScalableOnly = &H20000
00838 |     CF_TTOnly = &H40000
00839 |     CF_NoFaceSel = &H80000
00840 |     CF_NoStyleSel = &H100000
00841 |     CF_NoSizeSel = &H200000
00842 |     ' Win95 only
00843 |     CF_SelectScript = &H400000
00844 |     CF_NoScriptSel = &H800000
00845 |     CF_NoVertFonts = &H1000000
00846 |
00847 |     CF_InitToLogFontStruct = &H40
00848 |     CF_Apply = &H200
00849 |     CF_EnableHook = &H8
00850 |     CF_EnableTemplate = &H10
00851 |     CF_EnableTemplateHandle = &H20
00852 |     CF_FontNotSupported = &H238
00853 | End Enum
00854 |
00855 | ' These are extra nFontType bits that are added to what is returned to the
00856 | ' EnumFonts callback routine
00857 |
00858 | Public Enum EFontType
00859 |     Simulated_FontType = &H8000
00860 |     Printer_FontType = &H4000
00861 |     Screen_FontType = &H2000
00862 |     Bold_FontType = &H100
00863 |     Italic_FontType = &H200
00864 |     Regular_FontType = &H400
00865 | End Enum
00866 |
00867 | Private Type TPRINTDLG
00868 |     lStructSize As Long
00869 |     hWndOwner As Long
00870 |     hDevMode As Long
00871 |     hDevNames As Long
00872 |     hdc As Long
00873 |     flags As Long
00874 |     nFromPage As Integer
00875 |     nToPage As Integer
00876 |     nMinPage As Integer
00877 |     nMaxPage As Integer

```

```

00878 | nCopies As Integer
00879 | hInstance As Long
00880 | lCustData As Long
00881 | lpfnPrintHook As Long
00882 | lpfnSetupHook As Long
00883 | lpPrintTemplateName As Long
00884 | lpSetupTemplateName As Long
00885 | hPrintTemplate As Long
00886 | hSetupTemplate As Long
00887 | End Type
00888
00889 | ' DEVMODE collation selections
00890 | Private Const DMCOLLATE_FALSE = 0
00891 | Private Const DMCOLLATE_TRUE = 1
00892
00893 | Private Declare Function PrintDlg Lib "COMDLG32.DLL" _
00894 |     Alias "PrintDlgA" (prtdlg As TPRINTDLG) As Integer
00895
00896 | Public Enum EPrintDialog
00897 |     PD_ALLPAGES = &H0
00898 |     PD_SELECTION = &H1
00899 |     PD_PAGENUMS = &H2
00900 |     PD_NOSELECTION = &H4
00901 |     PD_NOPAGENUMS = &H8
00902 |     PD_COLLATE = &H10
00903 |     PD_PRINTTOFILE = &H20
00904 |     PD_PRINTSETUP = &H40
00905 |     PD_NOWARNING = &H80
00906 |     PD_RETURNDC = &H100
00907 |     PD_RETURNIC = &H200
00908 |     PD_RETURNDEFAULT = &H400
00909 |     PD_SHOWHELP = &H800
00910 |     PD_ENABLEPRINTHOOK = &H1000
00911 |     PD_ENABLESETUPHOOK = &H2000
00912 |     PD_ENABLEPRINTTEMPLATE = &H4000
00913 |     PD_ENABLESETUPTEMPLATE = &H8000
00914 |     PD_ENABLEPRINTTEMPLATEHANDLE = &H10000
00915 |     PD_ENABLESETUPTEMPLATEHANDLE = &H20000
00916 |     PD_USEDEVMODECOPIES = &H40000
00917 |     PD_USEDEVMODECOPIESANDCOLLATE = &H40000
00918 |     PD_DISABLEPRINTTOFILE = &H80000
00919 |     PD_HIDEPRINTTOFILE = &H100000
00920 |     PD_NONETWORKBUTTON = &H200000
00921 | End Enum
00922
00923 | Private Type DEVNAMES
00924 |     wDriverOffset As Integer
00925 |     wDeviceOffset As Integer
00926 |     wOutputOffset As Integer
00927 |     wDefault As Integer
00928 | End Type
00929
00930 | Private Const CCHDEVICENAME = 32
00931 | Private Const CCHFORMNAME = 32
00932 | Private Type DevMode
00933 |     dmDeviceName As String * CCHDEVICENAME
00934 |     dmSpecVersion As Integer
00935 |     dmDriverVersion As Integer
00936 |     dmSize As Integer
00937 |     dmDriverExtra As Integer
00938 |     dmFields As Long
00939 |     dmOrientation As Integer
00940 |     dmPaperSize As Integer
00941 |     dmPaperLength As Integer
00942 |     dmPaperWidth As Integer
00943 |     dmScale As Integer
00944 |     dmCopies As Integer
00945 |     dmDefaultSource As Integer
00946 |     dmPrintQuality As Integer
00947 |     dmColor As Integer
00948 |     dmDuplex As Integer
00949 |     dmYResolution As Integer

```

```

00950 | dmTTOption As Integer
00951 | dmCollate As Integer
00952 | dmFormName As String * CCHFORMNAME
00953 | dmUnusedPadding As Integer
00954 | dmBitsPerPel As Integer
00955 | dmPelsWidth As Long
00956 | dmPelsHeight As Long
00957 | dmDisplayFlags As Long
00958 | dmDisplayFrequency As Long
00959 | End Type
00960 |
00961 | ' New Win95 Page Setup dialogs are up to you
00962 | Private Type POINTL
00963 |     x As Long
00964 |     y As Long
00965 | End Type
00966 | Private Type RECT
00967 |     Left As Long
00968 |     Top As Long
00969 |     Right As Long
00970 |     Bottom As Long
00971 | End Type
00972 |
00973 |
00974 | Private Type TPAGESETUPDLG
00975 |     lStructSize           As Long
00976 |     hWndOwner             As Long
00977 |     hDevMode              As Long
00978 |     hDevNames             As Long
00979 |     flags                 As Long
00980 |     ptPaperSize          As POINTL
00981 |     rtMinMargin          As RECT
00982 |     rtMargin              As RECT
00983 |     hInstance            As Long
00984 |     lCustData             As Long
00985 |     lpfnPageSetupHook    As Long
00986 |     lpfnPagePaintHook    As Long
00987 |     lpPageSetupTemplateName As Long
00988 |     hPageSetupTemplate   As Long
00989 | End Type
00990 |
00991 | ' EPaperSize constants same as vbPRPS constants
00992 | Public Enum EPaperSize
00993 |     epsLetter = 1 ' Letter, 8 1/2 x 11 in.
00994 |     epsLetterSmall ' Letter Small, 8 1/2 x 11 in.
00995 |     epsTabloid ' Tabloid, 11 x 17 in.
00996 |     epsLedger ' Ledger, 17 x 11 in.
00997 |     epsLegal ' Legal, 8 1/2 x 14 in.
00998 |     epsStatement ' Statement, 5 1/2 x 8 1/2 in.
00999 |     epsExecutive ' Executive, 7 1/2 x 10 1/2 in.
01000 |     epsA3 ' A3, 297 x 420 mm
01001 |     epsA4 ' A4, 210 x 297 mm
01002 |     epsA4Small ' A4 Small, 210 x 297 mm
01003 |     epsA5 ' A5, 148 x 210 mm
01004 |     epsB4 ' B4, 250 x 354 mm
01005 |     epsB5 ' B5, 182 x 257 mm
01006 |     epsFolio ' Folio, 8 1/2 x 13 in.
01007 |     epsQuarto ' Quarto, 215 x 275 mm
01008 |     eps10x14 ' 10 x 14 in.
01009 |     eps11x17 ' 11 x 17 in.
01010 |     epsNote ' Note, 8 1/2 x 11 in.
01011 |     epsEnv9 ' Envelope #9, 3 7/8 x 8 7/8 in.
01012 |     epsEnv10 ' Envelope #10, 4 1/8 x 9 1/2 in.
01013 |     epsEnv11 ' Envelope #11, 4 1/2 x 10 3/8 in.
01014 |     epsEnv12 ' Envelope #12, 4 1/2 x 11 in.
01015 |     epsEnv14 ' Envelope #14, 5 x 11 1/2 in.
01016 |     epsCSheet ' C size sheet
01017 |     epsDSheet ' D size sheet
01018 |     epsESheet ' E size sheet
01019 |     epsEnvDL ' Envelope DL, 110 x 220 mm
01020 |     epsEnvC3 ' Envelope C3, 324 x 458 mm
01021 |     epsEnvC4 ' Envelope C4, 229 x 324 mm

```

```

01022 | epsEnvC5           ' Envelope C5, 162 x 229 mm
01023 | epsEnvC6           ' Envelope C6, 114 x 162 mm
01024 | epsEnvC65          ' Envelope C65, 114 x 229 mm
01025 | epsEnvB4           ' Envelope B4, 250 x 353 mm
01026 | epsEnvB5           ' Envelope B5, 176 x 250 mm
01027 | epsEnvB6           ' Envelope B6, 176 x 125 mm
01028 | epsEnvItaly        ' Envelope, 110 x 230 mm
01029 | epsenvmonarch     ' Envelope Monarch, 3 7/8 x 7 1/2 in.
01030 | epsEnvPersonal    ' Envelope, 3 5/8 x 6 1/2 in.
01031 | epsFanfoldUS      ' U.S. Standard Fanfold, 14 7/8 x 11 in.
01032 | epsFanfoldStdGerman ' German Standard Fanfold, 8 1/2 x 12 in.
01033 | epsFanfoldLglGerman ' German Legal Fanfold, 8 1/2 x 13 in.
01034 | epsUser = 256     ' User-defined
01035 | End Enum
01036 |
01037 | ' EPrintQuality constants same as vbPRPQ constants
01038 | Public Enum EPrintQuality
01039 |     epqDraft = -1
01040 |     epqLow = -2
01041 |     epqMedium = -3
01042 |     epqHigh = -4
01043 | End Enum
01044 |
01045 | Public Enum EOrientation
01046 |     eoPortrait = 1
01047 |     eoLandscape
01048 | End Enum
01049 |
01050 | Private Declare Function PageSetupDlg Lib "COMDLG32" _
01051 |     Alias "PageSetupDlgA" (lppage As TPAGESETUPDLG) As Boolean
01052 |
01053 | Public Enum EPageSetup
01054 |     PSD_Defaultminmargins = &H0 ' Default (printer's)
01055 |     PSD_InWinIniIntlMeasure = &H0
01056 |     PSD_MINMARGINS = &H1
01057 |     PSD_MARGINS = &H2
01058 |     PSD_INTHOUSANDTHSOFINCHES = &H4
01059 |     PSD_INHUNDREDDTHSOFMILLIMETERS = &H8
01060 |     PSD_DISABLEMARGINS = &H10
01061 |     PSD_DISABLEPRINTER = &H20
01062 |     PSD_NoWarning = &H80
01063 |     PSD_DISABLEORIENTATION = &H100
01064 |     PSD_ReturnDefault = &H400
01065 |     PSD_DISABLEPAPER = &H200
01066 |     PSD_ShowHelp = &H800
01067 |     PSD_EnablePageSetupHook = &H2000
01068 |     PSD_EnablePageSetupTemplate = &H8000
01069 |     PSD_EnablePageSetupTemplateHandle = &H20000
01070 |     PSD_EnablePagePaintHook = &H40000
01071 |     PSD_DisablePagePainting = &H80000
01072 | End Enum
01073 |
01074 |
01075 | Public Enum EPageSetupUnits
01076 |     epsuInches
01077 |     epsuMillimeters
01078 | End Enum
01079 |
01080 | ' Common dialog errors
01081 |
01082 | Private Declare Function CommDlgExtendedError Lib "COMDLG32" () As Long
01083 |
01084 | Public Enum EDialogError
01085 |     CDERR_DIALOGFAILURE = &HFFFF
01086 |
01087 |     CDERR_GENERALCODES = &H0&
01088 |     CDERR_STRUCTSIZE = &H1&
01089 |     CDERR_INITIALIZATION = &H2&
01090 |     CDERR_NOTEMPLATE = &H3&
01091 |     CDERR_NOINSTANCE = &H4&
01092 |     CDERR_LOADSTRFAILURE = &H5&
01093 |     CDERR_FINDRESFAILURE = &H6&

```

```

01094 CDERR_LOADRESFAILURE = &H7&
01095 CDERR_LOCKRESFAILURE = &H8&
01096 CDERR_MEMALLOCFailure = &H9&
01097 CDERR_MEMLOCKFAILURE = &HA&
01098 CDERR_NOHOOK = &HB&
01099 CDERR_REGISTERMSGFAIL = &HC&
01100
01101 PDERR_PRINTERCODES = &H1000&
01102 PDERR_SETUPFAILURE = &H1001&
01103 PDERR_PARSEFAILURE = &H1002&
01104 PDERR_RETDEFFFAILURE = &H1003&
01105 PDERR_LOADDRVFAILURE = &H1004&
01106 PDERR_GETDEVMODEFAIL = &H1005&
01107 PDERR_INITFAILURE = &H1006&
01108 PDERR_NODEVICES = &H1007&
01109 PDERR_NODEFAULTPRN = &H1008&
01110 PDERR_DNDMMISMATCH = &H1009&
01111 PDERR_CREATEICFAILURE = &H100A&
01112 PDERR_PRINTERNOTFOUND = &H100B&
01113 PDERR_DEFAULTDIFFERENT = &H100C&
01114
01115 CFERR_CHOOSEFONTCODES = &H2000&
01116 CFERR_NOFONTS = &H2001&
01117 CFERR_MAXLESSTHANMIN = &H2002&
01118
01119 FNERR_FILENAMECODES = &H3000&
01120 FNERR_SUBCLASSFAILURE = &H3001&
01121 FNERR_INVALIDFILENAME = &H3002&
01122 FNERR_BUFFERTOOSMALL = &H3003&
01123
01124 CCERR_CHOOSECOLORCODES = &H5000&
01125 End Enum
01126
01127 ' Hook and notification support:
01128 Private Type NMHDR
01129     hwndFrom As Long
01130     idFrom As Long
01131     code As Long
01132 End Type
01133 '// Structure used for all file based OpenFileName notifications
01134 Private Type OFNOTIFY
01135     hdr As NMHDR
01136     lpOFN As Long           ' Long pointer to OFN structure
01137     pszFile As String ' ; // May be NULL
01138 End Type
01139
01140 '// Structure used for all object based OpenFileName notifications
01141 Private Type OFNOTIFYEX
01142     hdr As NMHDR
01143     lpOFN As Long           ' Long pointer to OFN structure
01144     psf As Long
01145     LPVOID As Long         '// May be NULL
01146 End Type
01147
01148 Private Type OFNOTIFYshort
01149     hdr As NMHDR
01150     lpOFN As Long
01151 End Type
01152
01153 ' Messages:
01154 Private Const WM_INITDIALOG = &H110
01155 Private Const WM_NOTIFY = &H4E
01156 Private Const WM_USER = &H400
01157 Private Const WM_GETDLGCODE = &H87
01158 Private Const WM_NCDESTROY = &H82
01159
01160
01161 ' Notification codes:
01162 Private Const H_MAX As Long = &HFFFF + 1
01163 Private Const CDN_FIRST = (H_MAX - 601)
01164 Private Const CDN_LAST = (H_MAX - 699)
01165

```

```

01166  '// Notifications when Open or Save dialog status changes
01167 Private Const CDN_INITDONE = (CDN_FIRST - &H0)
01168 Private Const CDN_SELCHANGE = (CDN_FIRST - &H1)
01169 Private Const CDN_FOLDERCHANGE = (CDN_FIRST - &H2)
01170 Private Const CDN_SHAREVIOLATION = (CDN_FIRST - &H3)
01171 Private Const CDN_HELP = (CDN_FIRST - &H4)
01172 Private Const CDN_FILEOK = (CDN_FIRST - &H5)
01173 Private Const CDN_TYPECHANGE = (CDN_FIRST - &H6)
01174 Private Const CDN_INCLUDEITEM = (CDN_FIRST - &H7)
01175
01176 Private Const CDM_FIRST = (WM_USER + 100)
01177 Private Const CDM_LAST = (WM_USER + 200)
01178
01179 Private Const DWL_MSGRESULT = 0
01180 Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd
» As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
01181
01182 Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
01183     lpvDest As Any, lpvSource As Any, ByVal cbCopy As Long)
01184
01185
01186 ' =====
01187 ' Implementation:
01188 ' =====
01189
01190 ' Array of custom colors lasts for life of app
01191 Private alCustom(0 To 15) As Long, fNotFirst As Boolean
01192 Public Enum EPrintRange
01193     eprAll
01194     eprPageNumbers
01195     eprSelection
01196 End Enum
01197 Private m_lApiReturn As Long
01198 Private m_lExtendedError As Long
01199 Private m_dvmode As DevMode
01200 Private m_oEventSink As Object
01201
01202 Public Function DialogHook( _
01203     ByVal hDlg As Long, _
01204     ByVal msg As Long, _
01205     ByVal wParam As Long, _
01206     ByVal lParam As Long _
01207 )
01208     Dim tNMH As NMHDR
01209     Dim tOFNs As OFNOTIFYshort
01210     Dim tOF As OPENFILENAME
01211
01212     If Not (m_oEventSink Is Nothing) Then
01213         Select Case msg
01214             Case WM_INITDIALOG
01215                 DialogHook = m_oEventSink.InitDialog(hDlg)
01216             Case WM_NOTIFY
01217                 CopyMemory tNMH, ByVal lParam, Len(tNMH)
01218                 Select Case tNMH.code
01219                     Case CDN_SELCHANGE
01220                         ' Changed selected file:
01221                         DialogHook = m_oEventSink.FileChange(hDlg)
01222                     Case CDN_FOLDERCHANGE
01223                         ' Changed folder:
01224                         DialogHook = m_oEventSink.FolderChange(hDlg)
01225                     Case CDN_FILEOK
01226                         ' Clicked OK:
01227                         If Not m_oEventSink.ConfirmOK() Then
01228                             SetWindowLong hDlg, DWL_MSGRESULT, 1
01229                             DialogHook = 1
01230                         Else
01231                             SetWindowLong hDlg, DWL_MSGRESULT, 0
01232                         End If
01233                     Case CDN_HELP
01234                         ' Help clicked
01235                     Case CDN_TYPECHANGE
01236     1 2 3 4 DialogHook = m_oEventSink.TypeChange(hDlg)

```

```

01237 1 2 3 4 } Case CDN_INCLUDEITEM
01238      ' Hummm
01239      End Select
01240      } Case WM_NCDESTROY
01241          m_oEventSink.DialogClose
01242      End Select
01243  End If
01244 End Function

01245
01246
01247 { Public Property Get APIReturn() As Long
01248     'return object's APIReturn property
01249     APIReturn = m_lApiReturn
01250 End Property

01251 { Public Property Get ExtendedError() As Long
01252     'return object's ExtendedError property
01253     ExtendedError = m_lExtendedError
01254 End Property

01255
01256 { Private Sub Class_Initialize()
01257     #If fComponent Then
01258         InitColors
01259     #End If
01260 End Sub

01261
01262 { Function VBGetOpenFileName(Filename As String, _
01263     Optional FileTitle As String, _
01264     Optional FileMustExist As Boolean = True, _
01265     Optional MultiSelect As Boolean = False, _
01266     Optional ReadOnly As Boolean = False, _
01267     Optional HideReadOnly As Boolean = False, _
01268     Optional Filter As String = "All (*.*)| *.*", _
01269     Optional FilterIndex As Long = 1, _
01270     Optional InitDir As String, _
01271     Optional DlgTitle As String, _
01272     Optional DefaultExt As String, _
01273     Optional Owner As Long = -1, _
01274     Optional flags As Long = 0, _
01275     Optional Hook As Boolean = False, _
01276     Optional EventSink As Object _
01277     ) As Boolean
01278
01279     Dim opfile As OPENFILENAME, s As String, afFlags As Long
01280
01281     m_lApiReturn = 0
01282     m_lExtendedError = 0
01283
01284     With opfile
01285         .lStructSize = Len(opfile)
01286
01287         ' Add in specific flags and strip out non-VB flags
01288
01289         .flags = (-FileMustExist * OFN_FILEMUSTEXIST) Or _
01290             (-MultiSelect * OFN_ALLOWMULTISELECT) Or _
01291             (-ReadOnly * OFN_READONLY) Or _
01292             (-HideReadOnly * OFN_HIDEREADONLY) Or _
01293             (flags And CLng(Not (OFN_ENABLEHOOK Or _
01294                 OFN_ENABLETEMPLATE)))
01295         ' Owner can take handle of owning window
01296         If Owner <> -1 Then .hWndOwner = Owner
01297         ' InitDir can take initial directory string
01298         .lpstrInitialDir = InitDir
01299         ' DefaultExt can take default extension
01300         .lpstrDefExt = DefaultExt
01301         ' DlgTitle can take dialog box title
01302         .lpstrTitle = DlgTitle
01303
01304         If (Hook) Then
01305             'HookedDialog = Me
01306             .lpfnHook = lHookAddress(AddressOf DialogHookFunction)

```

```

01307 1 2 3 ' .flags = .flags Or OFN_ENABLEHOOK Or OFN_EXPLORER
01308 'Set m_oEventSink = EventSink
01309 End If
01310
01311 ' To make Windows-style filter, replace | and : with nulls
01312 Dim ch As String, i As Integer
01313 For i = 1 To Len(Filter)
01314     ch = Mid$(Filter, i, 1)
01315     If ch = "|" Or ch = ":" Then
01316         s = s & vbNullChar
01317     Else
01318         s = s & ch
01319     End If
01320 Next
01321 ' Put double null at end
01322 s = s & vbNullChar & vbNullChar
01323 .lpstrFilter = s
01324 .nFilterIndex = FilterIndex
01325
01326 ' Pad file and file title buffers to maximum path
01327 s = Filename & String$(MAX_PATH - Len(Filename), 0)
01328 .lpstrFile = s
01329 .nMaxFile = MAX_PATH
01330 s = FileTitle & String$(MAX_FILE - Len(FileTitle), 0)
01331 .lpstrFileTitle = s
01332 .nMaxFileTitle = MAX_FILE
01333 ' All other fields set to zero
01334
01335
01336 m_lApiReturn = GetOpenFileName(opfile)
01337 Set m_oEventSink = Nothing
01338 'ClearHookedDialog
01339 Select Case m_lApiReturn
01340 Case 1
01341     ' Success
01342     VBGetOpenFileName = True
01343     Filename = StrZToStr(.lpstrFile)
01344     FileTitle = StrZToStr(.lpstrFileTitle)
01345     flags = .flags
01346     ' Return the filter index
01347     FilterIndex = .nFilterIndex
01348     ' Look up the filter the user selected and return that
01349     Filter = FilterLookup(.lpstrFilter, FilterIndex)
01350     If (.flags And OFN_READONLY) Then ReadOnly = True
01351 Case 0
01352     ' Cancelled
01353     VBGetOpenFileName = False
01354     Filename = ""
01355     FileTitle = ""
01356     flags = 0
01357     FilterIndex = -1
01358     Filter = ""
01359 Case Else
01360     ' Extended error
01361     m_lExtendedError = CommDlgExtendedError()
01362     VBGetOpenFileName = False
01363     Filename = ""
01364     FileTitle = ""
01365     flags = 0
01366     FilterIndex = -1
01367     Filter = ""
01368 End Select
01369 Set m_oEventSink = Nothing
01370 End With
01371 End Function

```

---

```

01372 Private Function lHookAddress(lPtr As Long) As Long
01373     'Debug.Print lPtr
01374     lHookAddress = lPtr
01375 End Function

```

---

```

01376 Private Function StrZToStr(s As String) As String
01377     StrZToStr = Left$(s, lstrlen(s))

```

```

01378 End Function
01379
01380 Function VGetSaveFileName(Filename As String, _
01381     Optional FileName As String, _
01382     Optional OverWritePrompt As Boolean = True, _
01383     Optional Filter As String = "All (*.*)| *.*", _
01384     Optional FilterIndex As Long = 1, _
01385     Optional InitDir As String, _
01386     Optional DlgTitle As String, _
01387     Optional DefaultExt As String, _
01388     Optional Owner As Long = -1, _
01389     Optional flags As Long, _
01390     Optional Hook As Boolean = False, _
01391     Optional EventSink As Object _
01392     ) As Boolean
01393
01394     Dim opfile As OPENFILENAME, s As String
01395
01396     m_lApiReturn = 0
01397     m_lExtendedError = 0
01398
01399     With opfile
01400         .lStructSize = Len(opfile)
01401
01402         ' Add in specific flags and strip out non-VB flags
01403         .flags = (-OverWritePrompt * OFN_OVERWRITEPROMPT) Or _
01404             OFN_HIDEREADONLY Or _
01405             (flags And CLng(Not (OFN_ENABLEHOOK Or _
01406                 OFN_ENABLETEMPLATE)))
01407         ' Owner can take handle of owning window
01408         If Owner <> -1 Then .hWndOwner = Owner
01409         ' InitDir can take initial directory string
01410         .lpstrInitialDir = InitDir
01411         ' DefaultExt can take default extension
01412         .lpstrDefExt = DefaultExt
01413         ' DlgTitle can take dialog box title
01414         .lpstrTitle = DlgTitle
01415
01416         If (Hook) Then
01417             'HookedDialog = Me
01418             .lpfnHook = lHookAddress(AddressOf DialogHookFunction)
01419             .flags = .flags Or OFN_ENABLEHOOK Or OFN_EXPLORER
01420             'Set m_oEventSink = EventSink
01421         End If
01422
01423         ' Make new filter with bars (/) replacing nulls and double null at end
01424         Dim ch As String, i As Integer
01425         For i = 1 To Len(Filter)
01426             ch = Mid$(Filter, i, 1)
01427             If ch = "|" Or ch = ":" Then
01428                 s = s & vbNullChar
01429             Else
01430                 s = s & ch
01431             End If
01432         Next
01433         ' Put double null at end
01434         s = s & vbNullChar & vbNullChar
01435         .lpstrFilter = s
01436         .nFilterIndex = FilterIndex
01437
01438         ' Pad file and file title buffers to maximum path
01439         s = Filename & String$(MAX_PATH - Len(Filename), 0)
01440         .lpstrFile = s
01441         .nMaxFile = MAX_PATH
01442         s = FileName & String$(MAX_FILE - Len(FileName), 0)
01443         .lpstrDialogTitle = s
01444         .nMaxDialogTitle = MAX_FILE
01445         ' All other fields zero
01446
01447         m_lApiReturn = GetSaveFileName(opfile)
01448         Set m_oEventSink = Nothing
01449         'ClearHookedDialog

```

```

01450 1 2 {Select Case m_lApiReturn
01451      {Case 1
01452          VbGetSaveFileName = True
01453          Filename = StrZToStr(.lpstrFile)
01454          FileTitle = StrZToStr(.lpstrFileTitle)
01455          flags = .flags
01456          ' Return the filter index
01457          FilterIndex = .nFilterIndex
01458          ' Look up the filter the user selected and return that
01459          Filter = FilterLookup(.lpstrFilter, FilterIndex)
01460      }
01461      {Case 0
01462          ' Cancelled:
01463          VbGetSaveFileName = False
01464          Filename = ""
01465          FileTitle = ""
01466          flags = 0
01467          FilterIndex = 0
01468          Filter = ""
01469      }
01470      {Case Else
01471          ' Extended error:
01472          VbGetSaveFileName = False
01473          m_lExtendedError = CommDlgExtendedError()
01474          Filename = ""
01475          FileTitle = ""
01476          flags = 0
01477          FilterIndex = 0
01478          Filter = ""
01479      }
01480      End Select
01481  }
01482  End With
01483  End Function

01480
01481  Private Function FilterLookup(ByVal sFilters As String, ByVal iCur As Long) As String
01482      Dim iStart As Long, iEnd As Long, s As String
01483      iStart = 1
01484      If sFilters = "" Then Exit Function
01485      Do
01486          ' Cut out both parts marked by null character
01487          iEnd = InStr(iStart, sFilters, vbNullChar)
01488          If iEnd = 0 Then Exit Function
01489          iEnd = InStr(iEnd + 1, sFilters, vbNullChar)
01490          If iEnd Then
01491              s = Mid$(sFilters, iStart, iEnd - iStart)
01492          Else
01493              s = Mid$(sFilters, iStart)
01494          End If
01495          iStart = iEnd + 1
01496          If iCur = 1 Then
01497              FilterLookup = s
01498              Exit Function
01499          End If
01500          iCur = iCur - 1
01501      Loop While iCur
01502  End Function

01503
01504  Function VbGetFileTitle(sFile As String) As String
01505      Dim sFileTitle As String, cFileTitle As Integer
01506
01507      cFileTitle = MAX_PATH
01508      sFileTitle = String$(MAX_PATH, 0)
01509      cFileTitle = GetFileTitle(sFile, sFileTitle, MAX_PATH)
01510      If cFileTitle Then
01511          VbGetFileTitle = ""
01512      Else
01513          VbGetFileTitle = Left$(sFileTitle, InStr(sFileTitle, vbNullChar) - 1)
01514      End If
01515  End Function

01516
01517
01518  ' ChooseColor wrapper
01519  Function VbChooseColor(Color As Long, _
01520  1  Optional AnyColor As Boolean = True, _

```

```

01521     Optional FullOpen As Boolean = False, _
01522     Optional DisableFullOpen As Boolean = False, _
01523     Optional Owner As Long = -1, _
01524     Optional flags As Long, _
01525     Optional Hook As Boolean = False, _
01526     Optional EventSink As Object _
01527     ) As Boolean
01528
01529 Dim chclr As TCHOOSECOLOR
01530 chclr.lStructSize = Len(chclr)
01531
01532 ' Color must get reference variable to receive result
01533 ' Flags can get reference variable or constant with bit flags
01534 ' Owner can take handle of owning window
01535 If Owner <> -1 Then chclr.hWndOwner = Owner
01536
01537 ' Assign color (default uninitialized value of zero is good default)
01538 chclr.rgbResult = Color
01539
01540 ' Mask out unwanted bits
01541 Dim afMask As Long
01542 afMask = CLng(Not (CC_ENABLEHOOK Or _
01543     CC_ENABLETEMPLATE))
01544 ' Pass in flags
01545 chclr.flags = afMask And (CC_RGBInit Or _
01546     IIf(AnyColor, CC_AnyColor, CC_SolidColor) Or _
01547     (-FullOpen * CC_FullOpen) Or _
01548     (-DisableFullOpen * CC_PreventFullOpen))
01549
01550 If (Hook) Then
01551     'HookedDialog = Me
01552     'chclr.lpfHook = lHookAddress(AddressOf CCHookProc)
01553     'chclr.flags = chclr.flags Or CC_ENABLEHOOK
01554     'Set m_oEventSink = EventSink
01555 End If
01556
01557 ' If first time, initialize to white
01558 If fNotFirst = False Then InitColors
01559
01560 chclr.lpCustColors = VarPtr(alCustom(0))
01561 ' All other fields zero
01562
01563 m_lApiReturn = ChooseColor(chclr)
01564 Set m_oEventSink = Nothing
01565 'ClearHookedDialog
01566
01567 Select Case m_lApiReturn
01568 Case 1
01569     ' Success
01570     VBChooseColor = True
01571     Color = chclr.rgbResult
01572 Case 0
01573     ' Cancelled
01574     VBChooseColor = False
01575     Color = -1
01576 Case Else
01577     ' Extended error
01578     m_lExtendedError = CommDlgExtendedError()
01579     VBChooseColor = False
01580     Color = -1
01581 End Select
01582
01583 End Function

```

---

```

01584
01585 Friend Sub InitColors()
01586     Dim i As Integer
01587     ' Initialize with first 16 system interface colors
01588     For i = 0 To 15
01589         alCustom(i) = GetSysColor(i)
01590     Next
01591     fNotFirst = True
01592 End Sub

```

```

01593
01594 ' Property to read or modify custom colors (use to save colors in registry)
01595 Public Property Get CustomColor(i As Integer) As Long
01596     ' If first time, initialize to white
01597     If fNotFirst = False Then InitColors
01598     If i >= 0 And i <= 15 Then
01599         CustomColor = alCustom(i)
01600     Else
01601         CustomColor = -1
01602     End If
01603 End Property

```

---

```

01604
01605 Public Property Let CustomColor(i As Integer, iValue As Long)
01606     ' If first time, initialize to system colors
01607     If fNotFirst = False Then InitColors
01608     If i >= 0 And i <= 15 Then
01609         alCustom(i) = iValue
01610     End If
01611 End Property

```

---

```

01612
01613 ' ChooseFont wrapper
01614 Function VBChooseFont(CurFont As Font, _
01615     Optional PrinterDC As Long = -1, _
01616     Optional Owner As Long = -1, _
01617     Optional Color As Long = vbBlack, _
01618     Optional MinSize As Long = 0, _
01619     Optional MaxSize As Long = 0, _
01620     Optional flags As Long = 0, _
01621     Optional Hook As Boolean = False, _
01622     Optional EventSink As Object _
01623     ) As Boolean
01624
01625     m_lApiReturn = 0
01626     m_lExtendedError = 0
01627
01628     ' Unwanted Flags bits
01629     Const CF_FontNotSupported = CF_Apply Or CF_EnableHook Or CF_EnableTemplate
01630
01631     ' Flags can get reference variable or constant with bit flags
01632     ' PrinterDC can take printer DC
01633     If PrinterDC = -1 Then
01634         PrinterDC = 0
01635         If flags And CF_PrinterFonts Then PrinterDC = Printer.hdc
01636     Else
01637         flags = flags Or CF_PrinterFonts
01638     End If
01639     ' Must have some fonts
01640     If (flags And CF_PrinterFonts) = 0 Then flags = flags Or CF_ScreenFonts
01641     ' Color can take initial color, receive chosen color
01642     If Color <> vbBlack Then flags = flags Or CF_EFFECTS
01643     ' MinSize can be minimum size accepted
01644     If MinSize Then flags = flags Or CF_LimitSize
01645     ' MaxSize can be maximum size accepted
01646     If MaxSize Then flags = flags Or CF_LimitSize
01647
01648     ' Put in required internal flags and remove unsupported
01649     flags = (flags Or CF_InitToLogFontStruct) And Not CF_FontNotSupported
01650
01651     ' Initialize LOGFONT variable
01652     Dim fnt As LOGFONT
01653     Const PointsPerTwip = 1440 / 72
01654     fnt.lfHeight = -(CurFont.Size * (PointsPerTwip / Screen.TwipsPerPixelY))
01655     fnt.lfWeight = CurFont.Weight
01656     fnt.lfItalic = CurFont.Italic
01657     fnt.lfUnderline = CurFont.Underline
01658     fnt.lfStrikeOut = CurFont.Strikethrough
01659     ' Other fields zero
01660     StrToBytes fnt.lfFaceName, CurFont.Name
01661
01662     ' Initialize TCHOOSEFONT variable
01663     Dim cf As TCHOOSEFONT

```

```

01664     cf.lStructSize = Len(cf)
01665     If Owner <> -1 Then cf.hWndOwner = Owner
01666     cf.hdc = PrinterDC
01667     cf.lpLogFont = VarPtr(fnt)
01668     cf.iPointSize = CurFont.Size * 10
01669     cf.flags = flags
01670     cf.rgbColors = Color
01671     cf.nSizeMin = MinSize
01672     cf.nSizeMax = MaxSize
01673
01674     If (Hook) Then
01675         'HookedDialog = Me
01676         'cf.lpfnHook = lHookAddress(AddressOf CFHookProc)
01677         'cf.flags = cf.flags Or CF_EnableHook
01678         'Set m_oEventSink = EventSink
01679     End If
01680
01681     ' All other fields zero
01682     m_lApiReturn = ChooseFont(cf)
01683     Set m_oEventSink = Nothing
01684     'ClearHookedDialog
01685     Select Case m_lApiReturn
01686     Case 1
01687         ' Success
01688         VBChooseFont = True
01689         flags = cf.flags
01690         Color = cf.rgbColors
01691         CurFont.Bold = cf.nFontType And Bold_FontType
01692         'CurFont.Italic = cf.nFontType And Italic_FontType
01693         CurFont.Italic = fnt.lfItalic
01694         CurFont.Strikethrough = fnt.lfStrikeOut
01695         CurFont.Underline = fnt.lfUnderline
01696         CurFont.Weight = fnt.lfWeight
01697         CurFont.Size = cf.iPointSize / 10
01698         CurFont.Name = BytesToStr(fnt.lfFaceName)
01699     Case 0
01700         ' Cancelled
01701         VBChooseFont = False
01702     Case Else
01703         ' Extended error
01704         m_lExtendedError = CommDlgExtendedError()
01705         VBChooseFont = False
01706     End Select
01707
01708 End Function

```

---

```

01709
01710 ' PrintDlg wrapper
01711 Function VBPrintDlg(hdc As Long, _
01712     Optional PrintRange As EPrintRange = eprAll, _
01713     Optional DisablePageNumbers As Boolean, _
01714     Optional FromPage As Long = 1, _
01715     Optional ToPage As Long = &HFFFF, _
01716     Optional DisableSelection As Boolean, _
01717     Optional Copies As Integer, _
01718     Optional ShowPrintToFile As Boolean, _
01719     Optional DisablePrintToFile As Boolean = True, _
01720     Optional PrintToFile As Boolean, _
01721     Optional Collate As Boolean, _
01722     Optional PreventWarning As Boolean, _
01723     Optional Owner As Long, _
01724     Optional Printer As Object, _
01725     Optional flags As Long, _
01726     Optional Hook As Boolean = False, _
01727     Optional EventSink As Object _
01728 ) As Boolean
01729     Dim affFlags As Long
01730
01731     m_lApiReturn = 0
01732     m_lExtendedError = 0
01733
01734     ' Set PRINTDLG flags

```

```

01735     afFlags = flags
01736     afFlags = afFlags Or (Abs(DisablePageNumbers) * PD_NOPAGENUMS) Or _
01737         (Abs(DisablePrintToFile) * PD_DISABLEPRINTTOFILE) Or _
01738         (Abs(DisableSelection) * PD_NOSELECTION) Or _
01739         (Abs(PrintToFile) * PD_PRINTTOFILE) Or _
01740         (Abs(Not ShowPrintToFile) * PD_HIDEPRINTTOFILE) Or _
01741         (Abs(PreventWarning) * PD_NOWARNING) Or _
01742         (Abs(Collate) * PD_COLLATE) Or _
01743         PD_USEDEVMODECOPIESANDCOLLATE Or _
01744         PD_RETURNDC
01745     If PrintRange = eprPageNumbers Then
01746         afFlags = afFlags Or PD_PAGENUMS
01747     ElseIf PrintRange = eprSelection Then
01748         afFlags = afFlags Or PD_SELECTION
01749     End If
01750     ' Mask out unwanted bits
01751     afFlags = afFlags And Not PD_ENABLEPRINTHOOK
01752     afFlags = afFlags And Not PD_ENABLEPRINTTEMPLATE
01753     afFlags = afFlags And Not PD_ENABLESETUPHOOK
01754     afFlags = afFlags And Not PD_ENABLESETUPTEMPLATE
01755
01756     ' Fill in PRINTDLG structure
01757     Dim pd As TPRINTDLG
01758     pd.lStructSize = Len(pd)
01759     pd.hWndOwner = Owner
01760     pd.flags = afFlags
01761     pd.nFromPage = FromPage
01762     pd.nToPage = ToPage
01763     pd.nMinPage = 1
01764     pd.nMaxPage = &HFFFF
01765     If (Hook) Then
01766         ' HookedDialog = Me
01767         ' Set m_oEventSink = EventSink
01768         If (pd.flags And PD_PRINTSETUP) = PD_PRINTSETUP Then
01769             ' pd.flags = pd.flags Or PD_ENABLESETUPHOOK
01770             ' pd.lpfnSetupHook = lHookAddress(AddressOf PrintSetupHookProc)
01771         Else
01772             ' pd.flags = pd.flags Or PD_ENABLEPRINTHOOK
01773             ' pd.lpfnPrintHook = lHookAddress(AddressOf PrintHookProc)
01774         End If
01775     End If
01776
01777     ' Show Print dialog
01778     m_lApiReturn = PrintDlg(pd)
01779     ' ClearHookedDialog
01780     Set m_oEventSink = Nothing
01781     Select Case m_lApiReturn
01782     Case 1
01783         VBPrintDlg = True
01784         ' Return dialog values in parameters
01785         hdc = pd.hdc
01786         If (pd.flags And PD_PAGENUMS) Then
01787             PrintRange = eprPageNumbers
01788         ElseIf (pd.flags And PD_SELECTION) Then
01789             PrintRange = eprSelection
01790         Else
01791             PrintRange = eprAll
01792         End If
01793         FromPage = pd.nFromPage
01794         ToPage = pd.nToPage
01795         PrintToFile = (pd.flags And PD_PRINTTOFILE)
01796         ' Get DEVMODE structure from PRINTDLG
01797
01798         Dim pDevMode As Long
01799         pDevMode = GlobalLock(pd.hDevMode)
01800         CopyMemory m_dvmode, ByVal pDevMode, Len(m_dvmode)
01801         GlobalUnlock pd.hDevMode
01802         If (pd.flags And PD_COLLATE) = PD_COLLATE Then
01803             ' User selected collate option but printer driver
01804             ' does not support collation.
01805             ' Collation option must be set from the
01806             ' PRINTDLG structure:

```

1 2 3

```

01807 1 2 3 Collate = True
01808     Copies = pd.nCopies
01809     Else
01810         ' Print driver supports collation or collation
01811         ' not switched on.
01812         ' DEVMODE structure contains Collation and copy
01813         ' information
01814         ' Get Copies and Collate settings from DEVMODE structure
01815         Collate = (m_dvmode.dmCollate = DMCOLLATE_TRUE)
01816         Copies = m_dvmode.dmCopies
01817     End If
01818
01819     ' Set default printer properties
01820     On Error Resume Next
01821     If Not (Printer Is Nothing) Then
01822         Printer.Copies = Copies
01823         Printer.Orientation = m_dvmode.dmOrientation
01824         Printer.PaperSize = m_dvmode.dmPaperSize
01825         Printer.PrintQuality = m_dvmode.dmPrintQuality
01826     End If
01827     On Error GoTo 0
01828     Case 0
01829         ' Cancelled
01830         VBPrintDlg = False
01831     Case Else
01832         ' Extended error:
01833         m_lExtendedError = CommDlgExtendedError()
01834         VBPrintDlg = False
01835     End Select
01836 End Function
01837
01838 Friend Property Get DevMode() As DevMode
01839     DevMode = m_dvmode
01840 End Property
01841
01842 Public Function VBPAGESETUPDLG2( _
01843     Optional Owner As Long, _
01844     Optional DisableMargins As Boolean, _
01845     Optional DisableOrientation As Boolean, _
01846     Optional DisablePaper As Boolean, _
01847     Optional DisablePrinter As Boolean, _
01848     Optional LeftMargin As Single, _
01849     Optional MinLeftMargin As Single, _
01850     Optional RightMargin As Single, _
01851     Optional MinRightMargin As Single, _
01852     Optional TopMargin As Single, _
01853     Optional MinTopMargin As Single, _
01854     Optional BottomMargin As Single, _
01855     Optional MinBottomMargin As Single, _
01856     Optional PaperSize As EPaperSize = epsLetter, _
01857     Optional Orientation As EOrientaion = eoPortrait, _
01858     Optional PrintQuality As EPrintQuality = epqDraft, _
01859     Optional Units As EPageSetupUnits = epsuInches, _
01860     Optional Printer As Object, _
01861     Optional flags As Long, _
01862     Optional Hook As Boolean = False, _
01863     Optional EventSink As Object _
01864     ) As Boolean
01865     Dim affFlags As Long, afMask As Long
01866
01867     m_lApiReturn = 0
01868     m_lExtendedError = 0
01869     ' Mask out unwanted bits
01870     afMask = Not (PSD_EnablePagePaintHook Or _
01871         PSD_EnablePageSetupHook Or _
01872         PSD_EnablePageSetupTemplate)
01873     ' Set TPAGESETUPDLG flags
01874     affFlags = (-DisableMargins * PSD_DISABLEMARGINS) Or _
01875         (-DisableOrientation * PSD_DISABLEORIENTATION) Or _
01876         (-DisablePaper * PSD_DISABLEPAPER) Or _
01877         (-DisablePrinter * PSD_DISABLEPRINTER) _
01878     And afMask

```

```

01878 1 { If (flags And PSD_Defaultminmargins) = PSD_Defaultminmargins Then
01879     afFlags = afFlags Or PSD_Defaultminmargins
01880 } Else
01881     afFlags = afFlags Or PSD_MARGINS
01882 End If
01883 Dim lUnits As Long
01884 { If Units = epsuInches Then
01885     afFlags = afFlags Or PSD_INTHOUSANDTHSOFINCHES
01886     lUnits = 1000
01887 } Else
01888     afFlags = afFlags Or PSD_INHUNDREDTHSOFMILLIMETERS
01889     lUnits = 100
01890 End If
01891
01892 Dim psd As TPAGESETUPDLG
01893 ' Fill in PRINTDLG structure
01894 psd.lStructSize = Len(psd)
01895 psd.hWndOwner = Owner
01896 psd.rtMargin.Top = TopMargin * lUnits
01897 psd.rtMargin.Left = LeftMargin * lUnits
01898 psd.rtMargin.Bottom = BottomMargin * lUnits
01899 psd.rtMargin.Right = RightMargin * lUnits
01900 psd.rtMinMargin.Top = MinTopMargin * lUnits
01901 psd.rtMinMargin.Left = MinLeftMargin * lUnits
01902 psd.rtMinMargin.Bottom = MinBottomMargin * lUnits
01903 psd.rtMinMargin.Right = MinRightMargin * lUnits
01904 psd.flags = afFlags
01905 { If (Hook) Then
01906     'HookedDialog = Me
01907     'Set m_oEventSink = EventSink
01908     'psd.lpfPageSetupHook = lHookAddress(AddressOf PageSetupHook)
01909     'psd.flags = psd.flags Or PSD_EnablePageSetupHook
01910 End If
01911
01912 ' Show Print dialog
01913 { If PageSetupDlg(psd) Then
01914     VBPageSetupDlg2 = True
01915     ' Return dialog values in parameters
01916     TopMargin = psd.rtMargin.Top / lUnits
01917     LeftMargin = psd.rtMargin.Left / lUnits
01918     BottomMargin = psd.rtMargin.Bottom / lUnits
01919     RightMargin = psd.rtMargin.Right / lUnits
01920     MinTopMargin = psd.rtMinMargin.Top / lUnits
01921     MinLeftMargin = psd.rtMinMargin.Left / lUnits
01922     MinBottomMargin = psd.rtMinMargin.Bottom / lUnits
01923     MinRightMargin = psd.rtMinMargin.Right / lUnits
01924
01925     ' Get DEVMODE structure from PRINTDLG
01926     Dim dvmode As DevMode, pDevMode As Long
01927     pDevMode = GlobalLock(psd.hDevMode)
01928     CopyMemory dvmode, ByVal pDevMode, Len(dvmode)
01929     GlobalUnlock psd.hDevMode
01930     PaperSize = dvmode.dmPaperSize
01931     Orientation = dvmode.dmOrientation
01932     PrintQuality = dvmode.dmPrintQuality
01933     ' Set default printer properties
01934     On Error Resume Next
01935     { If Not (Printer Is Nothing) Then
01936         Printer.Copies = dvmode.dmCopies
01937         Printer.Orientation = dvmode.dmOrientation
01938         Printer.PaperSize = dvmode.dmPaperSize
01939         Printer.PrintQuality = dvmode.dmPrintQuality
01940     End If
01941     On Error GoTo 0
01942 End If
01943 Set m_oEventSink = Nothing
01944 'ClearHookedDialog
01945
01946 End Function
01947
01948 ' PageSetupDlg wrapper
01949 1 Function VBPageSetupDlg(Optional Owner As Long, _

```

```

01950     Optional DisableMargins As Boolean, _
01951     Optional DisableOrientation As Boolean, _
01952     Optional DisablePaper As Boolean, _
01953     Optional DisablePrinter As Boolean, _
01954     Optional LeftMargin As Long, _
01955     Optional MinLeftMargin As Long, _
01956     Optional RightMargin As Long, _
01957     Optional MinRightMargin As Long, _
01958     Optional TopMargin As Long, _
01959     Optional MinTopMargin As Long, _
01960     Optional BottomMargin As Long, _
01961     Optional MinBottomMargin As Long, _
01962     Optional PaperSize As EPaperSize = epsLetter, _
01963     Optional Orientation As EOrientation = eoPortrait, _
01964     Optional PrintQuality As EPrintQuality = epqDraft, _
01965     Optional Units As EPageSetupUnits = epsuInches, _
01966     Optional Printer As Object, _
01967     Optional flags As Long, _
01968     Optional Hook As Boolean = False, _
01969     Optional EventSink As Object _
01970 ) As Boolean
01971 Dim fLeftMargin As Single
01972 Dim fMinLeftMargin As Single
01973 Dim fRightMargin As Single
01974 Dim fMinRightMargin As Single
01975 Dim fTopMargin As Single
01976 Dim fMinTopMargin As Single
01977 Dim fBottomMargin As Single
01978 Dim fMinBottomMargin As Single
01979
01980 VBPageSetupDlg2 _
01981     Owner, _
01982     DisableMargins, _
01983     DisableOrientation, _
01984     DisablePaper, _
01985     DisablePrinter, _
01986     fLeftMargin, _
01987     fMinLeftMargin, _
01988     fRightMargin, _
01989     fMinRightMargin, _
01990     fTopMargin, _
01991     fMinTopMargin, _
01992     fBottomMargin, _
01993     fMinBottomMargin, _
01994     PaperSize, _
01995     Orientation, _
01996     PrintQuality, _
01997     Units, _
01998     Printer, _
01999     flags, _
02000     Hook, _
02001     EventSink
02002 LeftMargin = fLeftMargin
02003 MinLeftMargin = fMinLeftMargin
02004 RightMargin = fRightMargin
02005 MinRightMargin = fMinRightMargin
02006 TopMargin = fTopMargin
02007 MinTopMargin = fMinTopMargin
02008 BottomMargin = fBottomMargin
02009 MinBottomMargin = fMinBottomMargin
02010 End Function

02011
02012 #If fComponent = 0 Then
02013     Private Sub ErrRaise(e As Long)
02014         Dim sText As String, sSource As String
02015         If e > 1000 Then
02016             sSource = App.EXENAME & ".CommonDialog"
02017             Err.Raise COMError(e), sSource, sText
02018         Else
02019             ' Raise standard Visual Basic error
02020             sSource = App.EXENAME & ".VbError"

```

```
02021 1 2 3 Err.Raise e, sSource
02022     End If
02023     End Sub
02024 #End If

02025
02026
02027 Private Sub StrToBytes(ab() As Byte, s As String)
02028     If IsArrayEmpty(ab) Then
02029         ' Assign to empty array
02030         ab = StrConv(s, vbFromUnicode)
02031     Else
02032         Dim cab As Long
02033         ' Copy to existing array, padding or truncating if necessary
02034         cab = UBound(ab) - LBound(ab) + 1
02035         If Len(s) < cab Then s = s & String$(cab - Len(s), 0)
02036         'If UnicodeTypeLib Then
02037             Dim st As String
02038             st = StrConv(s, vbFromUnicode)
02039             CopyMemoryStr ab(LBound(ab)), st, cab
02040         'Else
02041         CopyMemoryStr ab(LBound(ab)), s, cab
02042         'End If
02043     End If
02044 End Sub

02045
02046
02047 Private Function BytesToStr(ab() As Byte) As String
02048     BytesToStr = StrConv(ab, vbUnicode)
02049 End Function

02050
02051 Private Function COMError(e As Long) As Long
02052     COMError = e Or vbObjectError
02053 End Function

02054 '
02055 Private Function IsArrayEmpty(va As Variant) As Boolean
02056     Dim v As Variant
02057     On Error Resume Next
02058     v = va(LBound(va))
02059     IsArrayEmpty = (Err <> 0)
02060 End Function
02061
```

---

**Procedure Index for Project - VBZip**

Pg #	Name	Referenced on Pages	Scope	Type
2	frmZipTester	3		Form
2	cmdRecurse_Click		Private	Sub
2	cmdSingleFile_Click		Private	Sub
2	Form_Load		Private	Sub
2	m_cZ_CommentRequest		Private	Sub
3	m_cZ_PasswordRequest		Private	Sub
3	m_cZ_Progress		Private	Sub
4	mZip	2, 3, 5-9, 12		Module
5	plAddressOf	6	Private	Function
8	ReplaceSection	7	Private	Function
5	VBZip	1-4, 6-32	Public	Function
7	ZipCommentCallback	6	Private	Function
7	ZipPasswordCallback	6, 8	Private	Function
6	ZipPrintCallback	7	Private	Function
6	ZipServiceCallback		Private	Function
9	cZip	2, 5-8, 10-12		Class
10	AddComment	2	Public	Property Let
10	AddComment	2	Public	Property Get
12	AddFileSpec	2	Public	Function
12	AllowAppend		Public	Property Let
12	AllowAppend		Public	Property Get
10	BasePath	2, 6	Public	Property Let
10	BasePath	2, 6	Public	Property Get
12	Class_Initialize	21	Private	Sub
12	ClearFileSpecs	2	Public	Sub
11	CommentRequest	2, 7, 9, 12	Friend	Sub
11	ConvertCRLFToLF		Public	Property Let
11	ConvertCRLFToLF		Public	Property Get
11	ConvertLFToCRLF		Public	Property Let
11	ConvertLFToCRLF		Public	Property Get
12	Delete	5, 9	Public	Sub
10	Encrypt	2, 4, 5, 9	Public	Property Let
10	Encrypt	2, 4, 5, 9	Public	Property Get
12	FileSpec	2, 5, 6, 9	Public	Property Get
12	FileSpecCount		Public	Property Get
11	FreshenFiles		Public	Property Let
11	FreshenFiles		Public	Property Get
10	IncludeSystemAndHiddenFiles		Public	Property Let
10	IncludeSystemAndHiddenFiles		Public	Property Get
11	MessageLevel		Public	Property Let
11	MessageLevel		Public	Property Get
11	PasswordRequest	3, 8, 9	Friend	Sub
11	ProgressReport	7	Friend	Sub
10	RecurseSubDirs	2, 12	Public	Property Let
10	RecurseSubDirs	2, 12	Public	Property Get
12	Service	4, 6	Friend	Sub
10	StoreDirectories	12	Public	Property Let
10	StoreDirectories	12	Public	Property Get
10	StoreFolderNames	2, 12	Public	Property Let
10	StoreFolderNames	2, 12	Public	Property Get
10	StoreVolumeLabel		Public	Property Let
10	StoreVolumeLabel		Public	Property Get
12	Success	2, 22, 25, 27	Public	Property Get
11	UpdateOnlyIfNewer		Public	Property Let
11	UpdateOnlyIfNewer		Public	Property Get
12	Zip	1-11, 13-32	Public	Sub
10	ZipFile	2, 5, 6	Public	Property Let
9	ZipFile	2, 5, 6, 10	Public	Property Get
13	GCommonDialog	2, 14-32		Class
21	APIReturn	20, 22-29	Public	Property Get
32	BytesToStr	27	Private	Function
21	Class_Initialize	12	Private	Sub
32	COMError	31	Private	Function
26	CustomColor		Public	Property Let
25	CustomColor	26	Public	Property Get
29	DevMode	15-17, 19, 20, 28, 30	Friend	Property Get

---

Pg #	Name	Referenced on Pages	Scope	Type
20	DialogHook	21, 23	Public	Function
31	ErrRaise		Private	Sub
21	ExtendedError	18, 20, 22-27, 29	Public	Property Get
24	FilterLookup	22	Private	Function
25	InitColors	21, 26	Friend	Sub
32	IsArrayEmpty		Private	Function
22	lHookAddress	21, 23, 25, 27, 28, 30	Private	Function
32	StrToBytes	26	Private	Sub
22	StrZToStr	24	Private	Function
24	VBChooseColor	25	Private	Function
26	VBChooseFont	27	Private	Function
24	VBGetFileTitle		Private	Function
21	VBGetOpenFileName	2, 22	Private	Function
23	VBGetSaveFileName	24	Private	Function
30	VBPageSetupDlg	29, 31	Private	Function
29	VBPageSetupDlg2	30, 31	Public	Function
27	VBPrintDlg	28, 29	Private	Function

---

## Table of Contents

frmZipTester .....	2
cmdRecurse_Click .....	2
cmdSingleFile_Click .....	2
Form_Load .....	2
m_cZ_CommentRequest .....	2
m_cZ_PasswordRequest .....	3
m_cZ_Progress .....	3
mZip .....	5
plAddressOf .....	5
VBZip .....	5
ZipServiceCallback .....	6
ZipPrintCallback .....	6
ZipCommentCallback .....	7
ZipPasswordCallback .....	7
ReplaceSection .....	8
cZip .....	9
ZipFile .....	9
ZipFile .....	10
BasePath .....	10
BasePath .....	10
Encrypt .....	10
Encrypt .....	10
AddComment .....	10
AddComment .....	10
IncludeSystemAndHiddenFiles .....	10
IncludeSystemAndHiddenFiles .....	10
StoreVolumeLabel .....	10
StoreVolumeLabel .....	10
StoreDirectories .....	10
StoreDirectories .....	10
StoreFolderNames .....	10
StoreFolderNames .....	10
RecurseSubDirs .....	10
RecurseSubDirs .....	10
UpdateOnlyIfNewer .....	11
UpdateOnlyIfNewer .....	11
FreshenFiles .....	11
FreshenFiles .....	11
MessageLevel .....	11
MessageLevel .....	11
ConvertCRLFToLF .....	11
ConvertCRLFToLF .....	11
ConvertLFTtoCRLF .....	11
ConvertLFTtoCRLF .....	11
ProgressReport .....	11
PasswordRequest .....	11
CommentRequest .....	11
Service .....	12
ClearFileSpecs .....	12
AddFileSpec .....	12
FileSpecCount .....	12
FileSpec .....	12
AllowAppend .....	12
AllowAppend .....	12
Zip .....	12
Success .....	12
Delete .....	12
Class_Initialize .....	12
GCommonDialog .....	20
DialogHook .....	20
APIReturn .....	21
ExtendedError .....	21
Class_Initialize .....	21
VBGetOpenFileName .....	21
lHookAddress .....	22
StrZToStr .....	22
VBGetSaveFileName .....	23

---

---

FilterLookup .....	24
VBGetFileTitle .....	24
VBChooseColor .....	24
InitColors .....	25
CustomColor .....	25
CustomColor .....	26
VBChooseFont .....	26
VBPrintDlg .....	27
DevMode .....	29
VBPageSetupDlg2 .....	29
VBPageSetupDlg .....	30
ErrRaise .....	31
StrToBytes .....	32
BytesToStr .....	32
COMError .....	32
IsEmpty .....	32

---