Using ActiveX DLLs in Liberty BASIC

Part 2
Originally published in NL 131, 2005
-
DennisMcK
Using ActiveX DLLs in Liberty BASIC | More XZip Routines | Listview and File Dialog
You did read part 1, didn't you? In part 2 we'll add some more XZip methods and tie everything together in a simple zip archiver application.
Download files here.

ActiveX Demo.zip

- Details
- Download
- 91 KB

The full code for the zip application is in the ActiveX Demo.zip file included with this newsletter and it's about 700 lines long so we'll just cover the major points here. For starters let's outline what the program needs to do, and decide what controls we will use.
At a minimum we need to:

1. Create new zip files.
2. Open a zip file.
3. Display the contents in detail.
4. Add one or more files to the archive at one time, with the option to preserve the relative path.
5. Delete one or more files from the archive at one time.
6. Extract the files from the archive to a folder of choice.

Items 1 and 2 can be handled with a standard filedialog. A new zip file is created by adding a file to a zip archive and specifying the path for the new zip file. If the zip file does not exist then XZip creates it. Items 3 and 5 suggest that the display be a multi-column and multi-row control with the ability to select more than one row at a time. The best choice for this would be a listview control. Item 4 can be accomplished with a multi-select file dialog, and a small dialog window can handle the 'preserve paths' option. Item 6 requires a folder to be chosen so we will use a Shell BrowseForFolder for that task. Listviews and multi-select file dialogs are not native to LB, so that means do it yourself. The code for these, as well as other items, is based entirely or in part on code from the Liberty BASIC 4 Companion and LB Workshop, reprinted with permission. Both are available at Alyces Restaurant.

More XZip Routines

Part 1 covered the fundamentals of initializing COM and supplied the first methods for this zip application, but a few more methods will be needed to accomplish our minimum application goals. To begin with, we will need a few more routines for XZip. Part 1 gave the routine for "Pack", and this next sub expands that functionality to include the path. If you studied part 1 you should have no trouble with these new methods.

```
Sub Zip.Pack.PreservePath XZipObject, src$, zip$
 'Preserves the path of the file being zipped.
 Calldll #com, "dhCallMethod", XZipObject As Ulong,
".Pack(%s, %s, %d)" As Ptr,_
   src$ As Ptr, zip$ As Ptr, 1 As Long, r As Long
End Sub
```

To delete a file from the archive...
Note: if all files are deleted from the archive, the archive itself is also deleted.

```
Sub Zip.Delete XZipObject, file$, zip$
 'Deletes file$ from zip$.
 Calldll #com, "dhCallMethod", XZipObject As Ulong,
".Delete(%s, %s)" As Ptr,_
   file$ As Ptr, zip$ As Ptr, r As Long
End Sub
```

This next sub uses slightly modified code from the example in part 1 to display the contents of a zip file. The results are now output to our listview control.

```
Sub ShowZipFiles XZipObject, hList, zipFile$
  'Display a list of the files contained in the zip file
  'along with their uncompressed size and any stored path.
  'Display the files in a listview control.
  tFolder = 1
  tFile = 2

  Call ListView.DeleteAllItems hList

  Calldll #com, "dhGetValue", "%o" As Ptr, comObj As Struct, _
    XZipObject As Ulong, ".Contents(%s)" As Ptr, zipFile$ As Ptr,
 r As Long
  objItems = comObj.obj.struct: comObj.obj.struct = 0

  count = GetValueLong(objItems, ".Count")

  For Idx = 1 To count
    Calldll #com, "dhGetValue", "%o" As Ptr, comObj As Struct, _
    objItems As Ulong, ".Item(%d)" As Ptr, Idx As Long, r As Long
    objItem = comObj.obj.struct: comObj.obj.struct = 0

    If GetValueLong(objItem, ".Type") = tFile Then
      listIdx = listIdx + 1
      fileName$ = GetValueStr$(objItem, ".Name")
      fileTime$ = GetValueDateTime$(objItem, ".Date")
      fileSize = GetValueLong(objItem, ".Size")
      filePath$ = GetValueStr$(objItem, ".Path")

      r = ListView.InsertItem(hList, listIdx-1, 0, fileName$)
      r = ListView.SetItem(hList, listIdx-1, 1, fileTime$)
      r = ListView.SetItem(hList, listIdx-1, 2, Str$(fileSize))
      r = ListView.SetItem(hList, listIdx-1, 3, filePath$)
    End If
    Call SetNothing objItem
  Next Idx

  Call SetNothing objItems
End Sub
```

Occasionally we will need to know how many files are in the archive.

```
Function GetArchiveFileCount(XZipObject, zipFile$)
  Calldll #com, "dhGetValue", "%o" As Ptr, comObj As Struct, _
    XZipObject As Ulong, ".Contents(%s)" As Ptr, zipFile$ As Ptr,
 r As Long
  objItems = comObj.obj.struct: comObj.obj.struct = 0
  GetArchiveFileCount = GetValueLong(objItems, ".Count")
  Call SetNothing objItems
End Function
```

Listview and File Dialog

This next part presents the routines for the listview control and the multi-select file dialog. For in-depth explanations of these items please consult the Liberty BASIC 4 Companion. The listview and file dialog code is portable except for the function ListView.GetSelectedFiles$ which is tailored for this zip application.

```
'----------- Listview control routines ---------------
[InitListView]
  LVS.NOSORTHEADER = 32768
  LVS.REPORT = 1
  LVS.SHOWSELALWAYS = 8

  Struct LVCOLUMN, _
    mask As Ulong, _
    fmt As Long, _
    cx As Long, _
    pszText$ As Ptr, _
    cchTextMax As Long, _
    iSubItem As Long, _
    iImage As Long, _
    iOrder As Long

  Struct LVITEM, _
    mask As Ulong, _
    iItem As Long, _
    iSubItem As Long, _
    state As Ulong, _
    stateMask As Ulong, _
    pszText$ As Ptr, _
    cchTextMax As Long, _
    iImage As Long, _
    lParam As Long, _
    iIndent As Long
Return

Function CreateListView(hParent, hInst, style, l, t, w, h)
  Calldll #user32, "CreateWindowExA", _WS_EX_CLIENTEDGE As Long,_
    "SysListView32" As Ptr, "" As Ptr, style As Long,_
    l As Long, t As Long, w As Long, h As Long,_
    hParent As Long, 0 As Long, hInst As Long,_
    "" As Ptr, CreateListView As Ulong
End Function

Function ListView.InsertColumn(hLV, col, width, txt$)
  LVM.INSERTCOLUMN = 4123
  LVCF.WIDTH = 2
  LVCF.TEXT = 4

  LVCOLUMN.mask.struct = LVCF.WIDTH Or LVCF.TEXT
  LVCOLUMN.cx.struct = width
  LVCOLUMN.pszText$.struct = txt$
  Calldll #user32, "SendMessageA", hLV As Long,
 LVM.INSERTCOLUMN As Long, _
    col As Long, LVCOLUMN As Struct, ListView.InsertColumn As Long
End Function

Function ListView.InsertItem(hLV, row, col, txt$)
  LVM.INSERTITEM = 4103
  LVIF.TEXT = 1

  LVITEM.mask.struct = LVIF.TEXT
  LVITEM.iItem.struct = row
  LVITEM.iSubItem.struct = col
  LVITEM.pszText$.struct = txt$

  Calldll #user32, "SendMessageA", hLV As Long, _
    LVM.INSERTITEM As Long, 0 As Long, LVITEM As Struct, _
    ListView.InsertItem As Long
End Function

Function ListView.SetItem(hLV, row, col, txt$)
  LVM.SETITEM = 4102
  LVIF.TEXT = 1

  LVITEM.mask.struct = LVIF.TEXT
  LVITEM.iItem.struct = row
  LVITEM.iSubItem.struct = col
  LVITEM.pszText$.struct = txt$

  Calldll #user32, "SendMessageA", hLV As Long, _
    LVM.SETITEM As Long, 0 As Long, LVITEM As Struct, _
    ListView.SetItem As Long
End Function

Sub Listview.SelectAll hLV
  LVIS.SELECTED = 2
  LVM.SETITEMSTATE = 4139

  LVITEM.stateMask.struct = LVIS.SELECTED
  LVITEM.state.struct = LVIS.SELECTED
  Calldll #user32, "SendMessageA", hLV As Long, _
    LVM.SETITEMSTATE As Long, -1 As Long, _
    LVITEM As Struct, r As Long
End Sub
```

```
Sub ListView.ClearAll hLV
  LVIS.UNSELECTED = 0
  LVIS.SELECTED = 2
  LVM.SETITEMSTATE = 4139

  LVITEM.stateMask.struct = LVIS.SELECTED
  LVITEM.state.struct = LVIS.UNSELECTED
  Calldll #user32, "SendMessageA", hLV As Long, _
    LVM.SETITEMSTATE As Long, -1 As Long, _
    LVITEM As Struct, r As Long
End Sub

Sub ListView.DeleteAllItems hLV
  LVM.DELETEALLITEMS = 4105
  Calldll #user32, "SendMessageA", hLV As Long, _
    LVM.DELETEALLITEMS As Long, 0 As Long, _
    0 As Long, r As Long
End Sub

Function ListView.GetSelectedCount(hLV)
  LVM.GETSELECTEDCOUNT = 4146
  Calldll #user32, "SendMessageA", hLV As Long,
 LVM.GETSELECTEDCOUNT As Long, _
    0 As Long, 0 As Long, _
    ListView.GetSelectedCount As Long
End Function

Function ListView.GetSelectedFiles$(hLV)
  'Gets the selected files in the listview and
  'returns them as a chr$(13) delimited list of filepaths.
  LVM.GETSELECTEDCOUNT = 4146
  LVM.GETNEXTITEM = 4108
  LVNI.SELECTED = 2
  LVM.GETITEMTEXTA = 4141
  Calldll #user32, "SendMessageA", hLV As Long,
 LVM.GETSELECTEDCOUNT As Long, _
    0 As Long, 0 As Long, _
    ItemsSelected As Long

  LVITEM.mask.struct = LVIF.TEXT
  LVITEM.cchTextMax.struct = _MAX_PATH
  LVITEM.pszText$.struct = Space$(_MAX_PATH)

  'Start search at -1 so LVM.GETNEXTITEM will start at item 0.
  SelectedItemIndex = -1
  For index = 1 To ItemsSelected
```

```
    Calldll #user32, "SendMessageA", hLV As Long,
 LVM.GETNEXTITEM As Long, _
      SelectedItemIndex As Long, LVNI.SELECTED As Long, _
      SelectedItemIndex As Long

    'get the path
    LVITEM.iSubItem.struct = 3
    Calldll #user32, "SendMessageA", hLV As Long,
 LVM.GETITEMTEXTA As Long, _
      SelectedItemIndex As Long, LVITEM As Struct, _
      selItem As Long
    path$=Winstring(LVITEM.pszText$.struct)
    path$ = Trim$(path$)
    If path$ <> "" Then items$ = items$ + path$

    'get the file name
    LVITEM.iSubItem.struct = 0  'first column
    Calldll #user32, "SendMessageA", hLV As Long,
 LVM.GETITEMTEXTA As Long, _
      SelectedItemIndex As Long, LVITEM As Struct, _
      selItem As Long

    file$=Winstring(LVITEM.pszText$.struct)
    file$ = Trim$(file$)
    If file$ <> "" Then items$ = items$ + file$ + Chr$(13)
  Next index
  ListView.GetSelectedFiles$ = items$
End Function

'---------- Multi-select or single-
select file dialog -----------------
Function GetOpenFileName$(title$, path$, filter$, filterIdx,
 multiselect)

'Returns a single file path or a chr$(13) delimited string of filepath
s
  OFN.EXPLORER = 524288

  Struct ofn, _
    lStructSize As Long, _
    hwndOwner As Long, _
    hInstance As Long, _
    lpstrFilter$ As Ptr, _
    lpstrCustomFilter$ As Ptr, _
    nMaxCustFilter As Long, _
    nFilterIndex As Long, _
    lpstrFile$ As Ptr, _
    nMaxFile As Long, _
    lpstrFileTitle$ As Ptr, _
    nMaxFileTitle As Long, _
    lpstrInitialDir$ As Ptr, _
    lpstrTitle$ As Ptr, _
    Flags As Long, _
    nFileOffset As Word, _
    nFileExtension As Word, _
    lpstrDefExt As Long, _
    lCustData As Long, _
    lpfnHook As Long, _
    lpTemplateName As Long

  ofn.lStructSize.struct = Len(ofn.struct)

  ofn.lpstrFilter$.struct = filter$
  ofn.nFilterIndex.struct = filterIdx
  'Allow a lot of files to be chosen, 32000 characters
  ofn.lpstrFile$.struct = Chr$(0) + Space$(32000) + Chr$(0)
  ofn.nMaxFile.struct = 32000
  ofn.lpstrInitialDir$.struct = path$ + Chr$(0)
  ofn.lpstrTitle$.struct = title$ + Chr$(0)
  ofn.lpstrDefExt.struct = 0

  If multiselect = 1 Then
    ofn.Flags.struct = _OFN_ALLOWMULTISELECT Or
 _OFN_PATHMUSTEXIST Or OFN.EXPLORER
  Else
    ofn.Flags.struct = _OFN_PATHMUSTEXIST Or OFN.EXPLORER
  End If

  Calldll #comdlg32, "GetOpenFileNameA", ofn As Struct, r As Long

  If r Then
    q$ = Chr$(34)
    path$ = Winstring(ofn.lpstrFile$.struct)
    ofnPath$ = Left$(path$,ofn.nFileOffset.struct)
    If Right$(ofnPath$,1) <> "\" Then
      ofnPath$ = ofnPath$ + "\"
    End If

    offset = ofn.lpstrFile$.struct + ofn.nFileOffset.struct
    file$ = Winstring(offset)
    GetOpenFileName$ = ofnPath$ + file$
    If multiselect = 1 Then
      If GetOpenFileName$ <> "" Then
        GetOpenFileName$ = GetOpenFileName$ + Chr$(13)
      End If
      While file$<>""
        offset = offset + Len(file$) + 1
        file$ = Winstring(offset)
        If file$<>"" Then
          GetOpenFileName$ = GetOpenFileName$ + ofnPath$ +
 file$ + Chr$(13)
        End If
      Wend
    End If
    GetOpenFileName$ = Trim$(GetOpenFileName$)
  Else
    'User cancelled or error,
    'see CommDlgExtendedError API.
  End If
End Function
```

The last listview function ListView.GetSelectedFiles$ and the GetOpenFileName$ function return a chr$(13) delimited string of file paths. This next function parses these strings and puts the file paths into an array.

```
Function ParseOFNlist(strList$, maxElements)
  'Fills array OFNlist$() with paths from a chr$(13) delimited string
  'or a single non-delimited path.
  maxElements = max(maxElements, 1)
  Redim OFNlist$(maxElements)
  idx = 0
  If Instr(strList$, Chr$(13)) Then
    While 1
      OFNlist$(idx) = Word$(strList$, idx+1, Chr$(13))
      If OFNlist$(idx) = "" Then Exit While
      idx = idx + 1
      If idx > maxElements Then
        Redim OFNlist$(1) 'erase the list
        Exit Function 'return 0, failed
      End If
    Wend
  Else
    OFNlist$(0) = strList$
  End If
  If OFNlist$(0) <> "" Then ParseOFNlist = 1 'return success
End Function
```

That covers the XZip methods and the custom widgets, excluding the BrowseForFolder. The rest of the code is simple enough. Now you have a free zip application that you can personalize to suit yourself, and the tools to use COM and ActiveX dlls in your Liberty BASIC programs. Don't forget that XZip.dll must be registered on your system for this to work.

Enjoy.