by Richard Russell, November 2013

For a long time it was thought to be impossible to call a method in a COM object from LB (without the aid of a 'helper' DLL). I proved that to be incorrect with my Speech synthesis without a helper DLL example, which calls the **SpVoice::Speak** method from native LB code.

However the technique used in that example has limitations, specifically it works only with methods that take **exactly two** parameters (which fortunately SpVoice::Speak does) and it's not easily adapted to other applications.

I have recently realised that both those shortcomings can be overcome! I have written a little LB function **CallMethod** which allows you to call *any* method in *any* COM object, and which is simple to use.

I have tackled the need to supply a variable number of parameters (with different types) by passing them in a **structure**. For example, suppose we want to call the IDirect3DDevice9::DrawPrimitive method, which takes three parameters. This is how it's done:

```
DrawPrimitive = 81
struct parm, PrimitiveType as long, _
             StartVertex as ulong, _
             PrimitiveCount as ulong
parm.PrimitiveType.struct = type
parm.StartVertex.struct = start
parm.PrimitiveCount.struct = count
result = CallMethod(IDirect3DDevice9, DrawPrimitive, parm.struct)
```

Here **IDirect3DDevice9** is the pointer to the object's interface, **DrawPrimitive** is the zero-based index of the method to be called (81) and **parm.struct** is the structure containing the parameters for the method call.

You can't have an 'empty' structure so in a case when the method takes no parameters pass an empty string instead. For example to call the **IDirect3DDevice9::Release** method:

```
Release = 2
result = CallMethod(IDirect3DDevice9, Release, "")
```

Being able to access COM objects from LB opens up a wide range of applications, for example a media player using Direct Show or animated 3D graphics using Direct3D. Most 'modern' Windows APIs are object-based.

Here is an example of the use of the **CallMethod** function to create a desktop shortcut. It does that by calling methods in the **IShellLink** interface to create the shortcut and the **IPersistFile** interface to store it on the desktop:

```
' Create and store a shortcut, version 1.0, 03-Nov-2013
' Demonstrates calling COM methods from Liberty BASIC!
' (C) Richard Russell 2013, http://www.rtrussell.co.uk/

    call shortcut SpecialFolder$(0) + "\Test Shortcut.lnk", _
                  StartupDir$ + "\liberty.exe", _
                  DefaultDir$, _
                  "Test shortcut created by Liberty BASIC"
    end

' Use the IShellLink and IPersistFile COM interfaces to
' create and store a shortcut to the specified object.
' LnkFile$ - the path/filename of the shortcut (.LNK)
' Target$  - the object for which to create a shortcut
' StartIn$ - the working directory to start in
' Comment$ - the description of the shortcut
sub shortcut LnkFile$, Target$, StartIn$, Comment$

    open "OLE32.DLL" for DLL as #ole32
    calldll #ole32, "CoInitialize", 0 as long, r as long

    struct clsid, a as long, b as long, c as long, d as long
    struct iidsl, a as long, b as long, c as long, d as long
    struct iidpf, a as long, b as long, c as long, d as long

    clsid.a.struct = hexdec("00021401") ' CLSID_ShellLink
    clsid.b.struct = hexdec("00000000")
    clsid.c.struct = hexdec("000000C0")
    clsid.d.struct = hexdec("46000000")

    iidsl.a.struct = hexdec("000214EE") ' IID_IShellLink
    iidsl.b.struct = hexdec("00000000")
    iidsl.c.struct = hexdec("000000C0")
    iidsl.d.struct = hexdec("46000000")

    iidpf.a.struct = hexdec("0000010B") ' IID_IPersistFile
    iidpf.b.struct = hexdec("00000000")
    iidpf.c.struct = hexdec("000000C0")
    iidpf.d.struct = hexdec("46000000")

    ' Get a pointer to the IShellLink interface:
    CLSCTX.INPROC.SERVER = 1
    struct temp, v as long
    calldll #ole32, "CoCreateInstance", clsid as struct, 0 as long, _
                    CLSCTX.INPROC.SERVER as long, iidsl as struct, _
                    temp as struct, r as long
```

```
    psl = temp.v.struct
    if psl = 0 then notice "Cannot create IShellLink interface" : end

    ' Set the target object, working directory and description:
    struct parm, psz as ptr
    parm.psz.struct = Target$
    result = CallMethod(psl, 20, parm.struct) ' IShellLink::SetPath
    parm.psz.struct = StartIn$
    result = CallMethod(psl, 9, parm.struct)
' IShellLink::SetWorkingDirectory
    parm.psz.struct = Comment$
    result = CallMethod(psl, 7, parm.struct)
' IShellLink::SetDescription

    ' Query IShellLink for the IPersistFile interface:
    struct temp, v as long
    struct parm, iid as struct, ppv as struct
    parm.iid.struct = iidpf.struct
    parm.ppv.struct = temp.struct
    result = CallMethod(psl, 0, parm.struct)
' IShellLink::QueryInterface
    temp.struct = parm.ppv.struct
    ppf = temp.v.struct
    IF ppf = 0 then notice
"Cannot create IPersistFile interface" : end

    ' Convert the path/filename string to Unicode:
    wsz$ = space$(2*len(LnkFile$) + 2)
    n = len(wsz$) / 2
    calldll #kernel32, "MultiByteToWideChar", 0 as long, 0 as long, _
                       LnkFile$ as ptr, -1 as long, wsz$ as ptr, _
                       n as long, r as long

    ' Save the shortcut:
    struct parm, pszFilename as ptr, fRemember as long
    parm.pszFilename.struct = wsz$
    parm.fRemember.struct = 1
    result = CallMethod(ppf, 6, parm.struct) ' IPersistFile::Save

    ' Tidy up:
    result = CallMethod(ppf, 2, "") ' IPersistFile::Release
    result = CallMethod(psl, 2, "") ' IShellLink::Release
    calldll #ole32, "CoUninitialize", r as void
    close #ole32
end sub
```

```
' Call a COM method:
' object - a pointer to the COM object interface
' method - the zero-based index of the method to be called
' parm$  - a structure containing the parameters to be passed
function CallMethod(object, method, parm$)
    code$ = chr$(139)+"D$"+chr$(4)+chr$(139)+"T$"+chr$(8)+chr$(139)+
"L$" _
    + chr$(16)+"VW"+chr$(139)+"t$"+chr$(20)+chr$(43)+chr$(225)+
chr$(139) _
    + chr$(252)+chr$(243)+chr$(164)+chr$(80)+chr$(139)+chr$(0)+
chr$(255) _
    + chr$(20)+chr$(144)+chr$(95)+chr$(94)+chr$(194)+chr$(16)+chr$(0)

    p$ = parm$
    n = len(p$)
    calldll #user32, "CallWindowProcA", code$ as ptr, object as
long, _
            method as long, p$ as ptr, n as long, CallMethod as long
end function

' Get the location of a Special Folder:
function SpecialFolder$(csidl)
    struct idl, pp as long
    calldll #shell32, "SHGetSpecialFolderLocation", _
      0 as long, csidl as long, idl as struct, r as long
    if r = 0 then
      path$ = space$(_MAX_PATH)
      ppidl = idl.pp.struct
      calldll #shell32, "SHGetPathFromIDListA", _
        ppidl as long, path$ as ptr, r as long
      SpecialFolder$ = trim$(path$)
      open "OLE32.DLL" for DLL as #ole32
      calldll #ole32, "CoTaskMemFree", ppidl as long, _
        r as long
      close #ole32
    end if
end function
```