VBForums™

Custom Search

User Name Password Log in Help Register

□ Remember Me?

Forum What's New?

New Posts FAQ Calendar Forum Actions Quick Links Advanced Search

VBForums VBForums CodeBank CodeBank - Visual Basic 6 and earlier [VB6] Call Functions By Pointer (Universall DLL Calls)

Results 1 to 40 of 41 Page 1 of 2 **1** 2 Last

**Thread: [VB6] Call Functions By Pointer (Universall DLL Calls)**

G+ Sha Tweet Share

Thread Tools Display

Nov 22nd, 2014, 12:29 PM #1

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

| | |
|---|---|
| Join Date: | Oct 2007 |
| Location: | Beside Waldo |
| Posts: | 16,445 |

**[VB6] Call Functions By Pointer (Universall DLL Calls)**

I was recently made aware that a function I've used from time to time for calling virtual functions of COM objects was perfectly adept at calling functions from just about any standard DLL out there. So, I whipped up a 'generic' class that can call both standard DLL functions & COM VTable functions. No thunks are used, just a couple of supporting API calls in the main class, including the low-level core API function: DispCallFunc

What does this mean for you? Well, it does allow you to call DLL functions from nearly 10 different calling conventions, including the two most common: StdCall & CDecl. It also allows you to call virtual functions from COM objects. And if you wish, it means you do not have to declare a single API function declaration in your VB project. Though, personally, I'd use it for calling DLL conventions other than StdCall.

I'd consider this topic advanced for one reason only. This is very low level. If you provide incorrect parameter information to the class, your project is likely to crash. For advanced coders, we have no problem doing the research to understand what parameter information is required, be it variable type, a pointer, a pointer to a pointer, function return types, etc, etc. Not-so-advanced coders just want to plug in values & play, but when playing at such a low level, that usually results in crashes, frustration.

The attachment includes very simple examples of calling DLL functions and calling a COM virtual function. You will notice that the form has no API function declarations, though several DLL functions are called & executed correctly. A sample call to the class might look like:

Code:
```
Debug.Print myClass.CallFunction_DLL("user32.dll", "IsWindowUnicode", STR_NONE, CR_LONG, _
                                      CC_STDCALL, Me.hWnd)
```

For DLL calls, the class takes the DLL name and function name to be called. Technically, you aren't passing the function pointer to the class. However, the class does make the call to the pointer, not via declared API functions. Just thought I'd throw this comment in, should someone suggest we aren't really calling functions by pointer. The class is, the user calling the class is not, but can be if inclined to modify the code a bit.

**Limitations**: Callbacks from non-StdCall DLLs/functions
If whatever function you are calling requires a callback pointer, then stack corruption is likely from all calling conventions where you pass a VB function pointer as the callback address. The exceptions are stdCall DLLs and also CDecl calls, if the thunk/patch option in the class is used.

**Tip**: If you really like this class, you may want to instantiate one for each DLL you will be calling quite often. This could speed things up a bit when making subsequent calls. As is, the class will load the requested DLL into memory if it isn't already. Once class is called again, for a different DLL, then the previous DLL is unloaded if needed & the new DLL loaded as needed. So, if you created cUser32, cShell32, cKernel32 instances, less code is executed in the class if it doesn't have to drop & load DLLs.

vb Code:
```
' top of form
Private cUser32 As cUniversalDLLCalls
Private cKernel32 As cUniversalDLLCalls
Private cShell32 As cUniversalDLLCalls

' in form load
Set cUser32 = New cUniversalDLLCalls
Set cKernel32 = New cUniversalDLLCalls
Set cShell32 = New cUniversalDLLCalls
' now use cUser32 for all user32.dll calls, cKernel32 for kernel32, cShell32
```

**Tip**: When using the STR_ANSI flag to indicate the passed parameters include string values destined for ANSI functions, the class will convert the passed string to ANSI before calling the function. Doing so, default Locale is used for string conversion. If this is a problem, you should ensure you convert the string(s) to ANSI before passing it to the class. If you do this conversion, use STR_NONE & pass the string via StrPtr(). FYI: strings used strictly as a buffer for return values should always be passed via StrPtr() and the flag STR_NONE used; regardless if destined for ANSI or unicode functions. ANSI strings are never passed to COM interfaces. Always use StrPtr(theString) for any string parameters to those COM methods.

vb Code:
```
' how to have a VB string contain ANSI vs Unicode
myString = StrConv(myString, vbFromUnicode, [Locale ID])
' how to convert the returned ANSI string to a proper VB string
myString = StrConv(myString, vbUnicode, [Locale ID])
```

**Tip**: If you ever need to call a *private* COM interface function by its pointer/address, post #24 below shows how that can be done. A slight modification to the attached class is required.

**Change History**
- 28 Nov 2014. Added thunk/patch/workaround for passing a VB function address to a CDECL dll that expects a CDECL callback address. Sample found in post #10 below

Attached Files
prjUniDLLcalls.zip (9.3 KB, 510 views)

*Last edited by LaVolpe; Mar 12th, 2015 at 08:44 AM.* ***Reason:*** *updated comments*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

Click Here to Expand Forum to Full Width

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 22nd, 2014, 12:32 PM #2

**LaVolpe**
**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Interface Definitions, VTables, etc**

This post will be dedicated to linking to known good posts/URLs of interfaces that not only provide their GUID, but the VTable Order. Will update this post from time to time as others add information to this thread.

**So if you researched an Interface and want to share it, please post it & include at least these 3 basic pieces of information: GUID, VTable order & link describing the virtual functions.** An example would be appreciated also.

You cannot assume that the listing of functions provided on MSDN pages is the actual VTable order. It used to be, but no longer is reliable. Order is extremely important, because virtual functions are called relative to their offset from the inherited IUnknown interface. VTable entries are in multiples of four.

A starter page. Layout of the COM object

So, you have a string GUID, how do you get it to a Long value for passing to appropriate functions? Simple:

Code:
```
Private Declare Function IIDFromString Lib "ole32.dll" (ByVal lpszProgID As Long, piid As
' sample of getting the IDataObject GUID
Dim aGUID(0 To 3) As Long
Call IIDFromString(StrPtr("{0000010e-0000-0000-C000-000000000046}"), ByVal VarPtr(aGUID(0)
```

How do we know if an object supports a specific interface? We ask the object...

Code:
```
Dim IID_IPicture As Long, aGUID(0 To 3) As Long, sGUID As String
Dim c As cUniversalDLLCalls
Const IUnknownQueryInterface As Long = 0&    ' IUnknown vTable offset to Query implemented
Const IUnknownRelease As Long = 8&           ' IUnkownn vTable offset to decrement referen

    ' ask if Me.Icon picture object supports IPicture
    sGuid = "{7BF80980-BF32-101A-8BBB-00AA00300CAB}"
    Set c = New cUniversalDLLCalls
    c.CallFunction_DLL "ole32.dll", "IIDFromString", STR_NONE, CR_LONG, CC_STDCALL, StrPt
    c.CallFunction_COM ObjPtr(Me.Icon), IUnknownQueryInterface, CR_LONG, CC_STDCALL, VarPt
    If IID_IPicture <> 0& Then
        ' do stuff
        ' Release the IPicture interface at some point. QueryInterface calls AddRef intern
        c.CallFunction_COM IID_IPicture, IUnknownRelease, CR_LONG, CC_STDCALL
    End If
```

Here's a few interfaces to start this thread out...

**IUnknown**: GUID {00000000-0000-0000-C000-000000000046}
VTable Order: QueryInterface, AddRef, Release

**IPicture**: GUID {7BF80980-BF32-101A-8BBB-00AA00300CAB}
VTable Order: GetHandle, GetHPal, GetType, GetWidth, GetHeight, Render, SetHPal, GetCurDC,
SelectPicture, GetKeepOriginalFormat, SetKeepOriginalFormat, PictureChanged, SaveAsFile, GetAttributes

**IDataObject**: GUID {0000010e-0000-0000-C000-000000000046}
VTable Order: GetData, GetDataHere, QueryGetData, GetCanonicalFormatEtc, SetData,
EnumFormatEtc, DAdvise, DUnadvise, EnumDAdvise

Tip #1. Get the IDataObject from the Data parameter of VB's OLEDrag[...] events

Code:
```
Private Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destinatio

Dim IID_DataObject As Long
CopyMemory IID_DataObject, ByVal ObjPtr(Data) + 16&, 4&
' you now have an unreferenced pointer to the IDataObject
```

Tip #2. Get IDataObject of the clipboard

Code:
```
Private Declare Function OleGetClipboard Lib "ole32.dll" (ByRef ppDataObj As Long) As Long

Dim IID_DataObject As Long
OleGetClipboard IID_DataObject
' if IID_DataObject is non-null, you have a referenced pointer to the IDataObject
' Referenced pointers must call IUnknown.Release
```

**IOLEObject**: GUID {00000112-0000-0000-C000-000000000046}
VTable Order: SetClientSite, GetClientSite, SetHostNames, Close, SetMoniker, GetMoniker,
InitFromData, GetClipboardData, DoVerb, EnumVerbs, Update, IsUpToDate, GetUserClassID, GetUserType,
SetExtent, GetExtent, Advise, EnumAdvise, GetMiscStatus, SetColorScheme

**IStream**: inherits IUnknown:ISequentialStream. GUID {0000000C-0000-0000-C000-000000000046}
VTable Order: Read [from ISequentialStream], Write [from ISequentialStream], Seek, SetSize,
CopyTo, Commit, Revert, LockRegion, UnlockRegion, Stat, Clone

**ITypeLib**: GUID {00020402-0000-0000-C000-000000000046}
VTable Order: GetTypeInfoCount, GetTypeInfo, GetTypeInfoType, GetLibAttr,
GetTypeComp, GetDocumentation, IsName, FindName, ReleaseTLibAttr

**ITypeInfo**: GUID {00020401-0000-0000-C000-000000000046}
VTable Order: GetTypeAttr, GetTypeComp, GetFuncDesc, GetVarDesc, GetNames,
GetRefTypeOfImplType, GetImplTypeFlags, GetIDsOfNames, Invoke, GetDocumentation, GetDLLEntry,
GetRefTypeInfo, AddressOfMember, CreateInstance, GetMops, GetContainingTypeLib, ReleaseTypeAttr,
ReleaseFuncDesc, RelaseVarDesc

*Last edited by LaVolpe; Dec 5th, 2014 at 02:26 PM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 22nd, 2014, 06:38 PM #3

**Max187Boucher**

PowerPoster

Join Date: Aug 2011
Location: B.C., Canada
Posts: 2,887

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Very nice example LaVolpe! I had discovered something similar, just not as sophisticated

I was able to set form caption using the loadlibrary/freelibrary/getprocaddress/callwindowproc apis.

Why I wanted to do this is because a user could use their own APIs once the project is compiled, without having to recompile the whole project with the api that he needs to use. Kind of like using the script control which you can write your own function/subs on a **compiled (exe)** program, and **execute** them. This would allow users to call APIs from your compiled program with a simple textbox.

I will examine your code more closely when I get time, this is exactly what I was trying to get to, with the little experience I got

**Reply With Quote**

Nov 22nd, 2014, 07:00 PM #4

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Think you'll have fun playing with it. Gotta make sure you pass the parameters to class in proper variable type (VarType).

What will get ppl in trouble is assuming a value is Long when it's Integer. For example, if you pass the class the number 5 for a parameter that the API expects to be Long vartype, probably gonna have issues. Why? Well, Debug.Print VarType(5) = vbInteger returns true. Whole numbers have the vartype Integer, Long, Double depending on their value & the inclusive min/max ranges of the various vartypes. Likewise, if one were to declare a variable as Variant and pass that variable as a function parameter, same issue if the function expects Long vartypes. In the class' code comments, I stressed to use VB's conversion functions when in doubt, i.e., CLng(), CInt(), etc.

Edited: VB doesn't have this issue, because you declare a Function with ByRef/ByVal and the parameter type, i.e., ByVal hWnd As Long. The class has no way of knowing the function's definition because there are none. I could've forced the user to include the vartype with each parameter, but felt that was too cumbersome. What I didn't want to do was basically build a complete parsing engine, somewhat similar to what VB must be doing.

Strings sent to an ANSI function, passed as values or variable names, will be converted to ANSI if the STR_ANSI flag is passed. If these are not passed with that flag, gonna get bad results. One thing that is awkward is passing a string variable to an ANSI function, changes that variable's contents. Take a look at this example:

- variable strCaption = "Form1"
- strCaption is passed to an ANSI function with STR_ANSI flag set
- function call works flawlessly, but...
- now strCaption was changed because the class needed to convert the contents of the unicode variable strCaption to ANSI
:: was Form1 using 10 bytes (unicode), now is Form1 using 5 bytes (ANSI)

I could've reversed the process after the function was called to fix it, but felt this issue wouldn't be the norm. Also, would've had to set up some tracking system to know when to revert & when not to. Feel informing users more beneficial to all, which is what I'm using this reply for. Though I may consider readdressing this via code... ?

Above said, a workaround in these cases, pass the variable like so: strCaption & ""
-- VB sends a copy of the concatenated variable to the class, not the variable itself. No change to strCaption.

If not wanting to concatenate, pass enclosed with parentheses: (strCaption). Same result -- no change to strCaption

*Last edited by LaVolpe; Nov 23rd, 2014 at 01:53 PM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 24th, 2014, 04:23 AM #5

**Bonnie West**

Default Member

Join Date: Jun 2012
Location: InIDE
Posts: 4,057

**Re: Interface Definitions, VTables, etc**

> Originally Posted by **LaVolpe**
>
> ***So if you researched an Interface and want to share it, please post it & include at least these 3 basic pieces of information: GUID, VTable order & link describing the virtual functions.*** *An example would be appreciated also.*

> Originally Posted by **Schmidt**
>
> *To keep the VTable-order of the method-signatures correct, you cannot rely on the MSDN - they sort alphabetically, which is almost always the wrong order.*
> *I usually take a good look into e.g. the Wine-Implementations (or -Documentation), where the real VTable-Order can be seen:*
> *http://fossies.org/dox/wine-1.7.31/i...ShellItem.html*
>
> *. . .*
>
> *Olaf*

```
On Local Error Resume Next: If Not Empty Is Nothing Then Do While Null: ReDim i(True To False) As
Currency: Loop: Else Debug.Assert CCur(CLng(CInt(CBool(False Imp True Xor False Eqv True)))):
Stop: On Local Error GoTo 0
```

My CodeBank Contributions

```
Declare Sub CrashVB Lib "msvbvm60" (Optional DontPassMe As Any)
```

**Reply With Quote**

Nov 24th, 2014, 07:17 AM #6

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Bonnie, nice link for a ton of interfaces, their VTables, & function descriptions. I didn't see any GUIDs.

One thing that concerns me, for example, is the VTable description for IDataObject. The one listed at that site has 3 more public functions than what is documented on MSDN. Not only that, the 3 functions are intermixed. I can only assume that the IDataObject described is a different GUID/class than that listed on MSDN? Similar situation for

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

IStream. This is where the GUID would be helpful.

Hmmm, another ok reference for OLE & multimedia interfaces

*Last edited by LaVolpe; Nov 24th, 2014 at 07:41 AM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 24th, 2014, 05:11 PM #7

**Schmidt**
PowerPoster
Join Date: Jun 2013
Posts: 3,105

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **LaVolpe**
> *I was recently made aware ...*

I'd personally prefer smaller functions instead of the large "one size fits all" approach you've posted...

The module you got your inspiration from, basically contained the needed pieces already -
(you gave nice background-infos for Users new to this topic though)...

To describe what I mean, here's the Unicode-capable "W"-version of the stdcall:
(directly callable from a *.bas-module - a Class is not really needed for this stuff)

Code:
```
Public Function stdCallW(sDll As String, sFunc As String, ByVal RetType As VbVarType, Para
Dim i As Long, V(), HRes As Long

  V = P 'make a copy of the params, to prevent problems with VT_Byref-Members in the Param
  For i = 0 To UBound(V)
    If VarType(P(i)) = vbString Then V(i) = StrPtr(P(i))
    VType(i) = VarType(V(i))
    VPtr(i) = VarPtr(V(i))
  Next i

  HRes = DispCallFunc(0, GetFuncPtr(sDll, sFunc), CC_STDCALL, RetType, i, VType(0), VPtr(
  If HRes Then Err.Raise HRes
End Function
```

I'd think, with such dedicated functions the DispCallFunc-API looks far less "scary" to NewComers -
and is easier to call (less Parameters to type or fiddle with).

Also note, how I handled the automatic String-Pointer-passing (the Param-Array-Members can hand it out directly to you).

The other Problem you encountered, is the Back-conversion of ANSI-String-params, which could be handled automatically this way:

Code:
```
Public Function stdCallA(sDll As String, sFunc As String, ByVal RetType As VbVarType, Para
Dim i As Long, pFunc As Long, V(), HRes As Long

  V = P 'make a copy of the params, to prevent problems with VT_Byref-Members in the Param
  For i = 0 To UBound(V)
    If VarType(P(i)) = vbString Then P(i) = StrConv(P(i), vbFromUnicode): V(i) = StrPtr(P
    VType(i) = VarType(V(i))
    VPtr(i) = VarPtr(V(i))
  Next i

  HRes = DispCallFunc(0, GetFuncPtr(sDll, sFunc), CC_STDCALL, RetType, i, VType(0), VPtr(

  For i = 0 To UBound(P) 'back-conversion of the ANSI-String-Results
    If VarType(P(i)) = vbString Then P(i) = StrConv(P(i), vbUnicode)
  Next i
  If HRes Then Err.Raise HRes
End Function
```

With the complete module (code further below), one can then write
(without any StrConv-Calls in case of the ANSI-calls)...

Into a Form:

Code:
```
Option Explicit

'VTable-order according to: http://fossies.org/dox/wine-1.7.31/ocidl_8idl_source.html#100
Enum eIPicture
  IUnk_QueryInterface
  IUnk_AddRef
  IUnk_Release

  IPic_Handle
  IPic_hPal
  IPic_Type
  IPic_Width
  IPic_Height
  '... a.s.o.
End Enum

Private Sub Form_Click()
AutoRedraw = True: Cls

Dim Unk As stdole.IUnknown, lHandle As Long, lWidth As Long, lHeight As Long

  'vtbl function calls (against a COM-interface, in this case IPicture - see Enum-Def abov
  Set Unk = Me.Icon 'cast to IUnknown, so that we can call the VTable directly  without fu
  Print vbLf; "COM interface calls..."

    vtblCall ObjPtr(Unk), eIPicture.IPic_Handle, VarPtr(lHandle)
    vtblCall ObjPtr(Unk), eIPicture.IPic_Width, VarPtr(lWidth)
    vtblCall ObjPtr(Unk), eIPicture.IPic_Height, VarPtr(lHeight)

  Dim Ico As IPictureDisp: Set Ico = Me.Icon 'just for the Printouts, another cast of our
  Print , "Me.Icon.Handle = "; Ico.Handle, " IPicture.Handle = "; lHandle
  Print , "Me.Icon.Width = "; Ico.Width, " IPicture.Width = "; lWidth
```

Here the *.bas Module-Code for DispCallFunc which allows the above:

Code:
```
Option Explicit

Private Declare Function DispCallFunc Lib "oleaut32" (ByVal pvInstance As Long, ByVal offs
Private Declare Function GetProcAddress Lib "kernel32" (ByVal hModule As Long, ByVal lpPro
Private Declare Function LoadLibrary Lib "kernel32" Alias "LoadLibraryA" (ByVal lpLibFileN
Private Declare Function FreeLibrary Lib "kernel32" (ByVal hLibModule As Long) As Long
Private Declare Function lstrlenA Lib "kernel32" (ByVal lpString As Long) As Long
Private Declare Function lstrlenW Lib "kernel32" (ByVal lpString As Long) As Long
Private Declare Sub RtlMoveMemory Lib "kernel32" (Dst As Any, Src As Any, ByVal BLen As Lo
Private Enum CALLINGCONVENTION_ENUM
  CC_FASTCALL
  CC_CDECL
  CC_PASCAL
  CC_MACPASCAL
  CC_STDCALL
  CC_FPFASTCALL
  CC_SYSCALL
  CC_MPWCDECL
  CC_MPWPASCAL
End Enum

Private LibHdls As New Collection, VType(0 To 63) As Integer, VPtr(0 To 63) As Long

Public Function stdCallW(sDll As String, sFunc As String, ByVal RetType As VbVarType, Para
Dim i As Long, V(), HRes As Long

  V = P 'make a copy of the params, to prevent problems with VT_Byref-Members in the Paran
  For i = 0 To UBound(V)
    If VarType(P(i)) = vbString Then V(i) = StrPtr(P(i))
    VType(i) = VarType(V(i))
```

Olaf

*Last edited by Schmidt; Nov 24th, 2014 at 05:28 PM.*

**Reply With Quote**

Nov 24th, 2014, 07:06 PM #8

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

### Re: [VB6] Call Functions By Pointer (Universall DLL Calls)

To each their own.

Personally, for simplicity, I'll keep a one-size fits all vs a ANSI/Unicode call for each of the possible calling conventions. I feel the routines are simple enough to follow for most coders. Basically, the routine I offered for DLL calls consist of 3 sections: DLL loading/unloading, ANSI string conversion if needed, parameter referencing. The routine for COM calls consists of just the parameter referencing section. Remove all the lengthy comments and amount of code is amazingly little.

Regarding "The other Problem you encountered, is the Back-conversion of ANSI-String-params, which could be handled..." I took that out of my routines, like your solution I was looping thru the parameters on function return to convert ANSI back to 2-byte characters. However, I felt that the callers should control that should LocaleID be an issue.

Inspiration? Inspiration for this unique API came in early 2007 when I was looking for a solution for Drag & Dropping of unicode file names. The code that inspired me can be found on planet source code at this link which does include a comment from me. My first attempt of using that API can be seen in a project at the same site with this link. Since then, Ive used this API dozens of times. I've become comfortable with it over the past 7.5 years. It was your comment in another thread, that led to this posting, when you said the API could be used for standard DLLs and that thunks for calling CDecl_ DLLs were unnecessary.

Do I expect anyone to build a production application using no API function declarations? No. This project is for the curious. I can see it being used, possibly streamlined, for CDECL calls and/or COM calls primarily. I know I wouldn't use it for calling STDCALL. Though I will be posting another project this week that does use the COM call portion of this project

*Last edited by LaVolpe; Nov 24th, 2014 at 10:03 PM.* ***Reason:*** *added references*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 27th, 2014, 02:52 AM #9

**FunkyDexter**

Super Moderator

Join Date: Apr 2005
Location: An obscure body in the SK system. The inhabitants call it Earth
Posts: 6,513

### Re: [VB6] Call Functions By Pointer (Universall DLL Calls)

I've split off a thread with GeoRaker's questions to a new thread here.

You can depend upon the Americans to do the right thing. But only after they have exhausted every other possibility - Winston Churchill

Hadoop actually sounds more like the way they greet each other in Yorkshire - Inferrd

**Reply With Quote**

Nov 28th, 2014, 11:08 AM #10

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

### Re: [VB6] Call Functions By Pointer (Universall DLL Calls)

Project updated to include a patch/thunk for passing a VB function address (_stdCall) as a callback to a _CDecl function expecting a _CDecl callback address. A simple test can be performed using the C runtime dll & it's quick sort function

1) Ensure you downloaded latest version from post #1 above.
2) In a button's click event, add this:

Code:

```
    Dim c As New cUniversalDLLCalls
    Dim vData(0 To 9) As Double
    Dim lCallback As Long
    Dim Index As Long

    For Index = 0 To 9: vData(Index) = Rnd * 100: Next

    ' generate a CDECL compatible callback that wraps the VB callback function
    lCallback = c.ThunkFor_CDeclCallbackToVB(AddressOf qsort_compare_dn, 2)
    ' 4 qsort params:
    '   pointer to data to be sorted
    '   number of items to be sorted
    '   size in bytes of each item
    '   _CDecl callback function
    Call c.CallFunction_DLL("msvcrt20.dll", "qsort", STR_NONE, CR_None, CC_CDECL, _
                            VarPtr(vData(0)), 10&, 8&, lCallback)
    ' done with the wrapper, release it
    c.ThunkRelease_CDECL lCallback

    For Index = 0 To 9: Debug.Print Index + 1, vData(Index): Next
```

3) In a bas module add these 2 functions. Notice params are Double because we are passing an array of Doubles to qsort

Code:

```
Public Function qsort_compare_dn(ByRef arg1 As Double, ByRef arg2 As Double) As Long
  Select Case arg2 - arg1
  Case Is < 0: qsort_compare_dn = -1
  Case Is > 0: qsort_compare_dn = 1
  End Select
End Function

Public Function qsort_compare_up(ByRef arg1 As Double, ByRef arg2 As Double) As Long
  Select Case arg2 - arg1
  Case Is < 0: qsort_compare_up = 1
  Case Is > 0: qsort_compare_up = -1
  End Select
End Function
```

Two new methods added. Be sure to review their comments:
1) ThunkFor_CDeclCallbackToVB
2) ThunkRelease_CDECL

Reason for a thunk:
_CDecl: The caller cleans the stack. _StdCall: The callee cleans the stack
When _CDecl calls to _StdCall while expecting _CDecl, stack corruption occurs because both are trying to clean the stack causing access violations. When the reverse order occurs, then stack not cleaned.

The core API used in the attached class will do the cleaning from _StdCall to _CDecl but since that API can't be used for callbacks, we need a way to prevent one of the calling conventions from cleaning the stack. That's where the thunk comes in. It is a wrapper around a VB callback function address. _CDecl calls the wrapper, the wrapper copies the stack & calls the VB callback function. VB cleans the stack, then the wrapper replaces the stack. _CDecl now cleans the original stack. 28 bytes of commonsense genius provided by Paul Caton, whose work is linked/credited in the ThunkFor_CDeclCallbackToVB function

*Last edited by LaVolpe; Nov 28th, 2014 at 10:21 PM.*

**Insomnia** is just a byproduct of, "*It can't be done"*

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 29th, 2014, 04:32 AM #11

**Bonnie West**

Default Member

Join Date: Jun 2012
Location: InIDE
Posts: 4,057

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **LaVolpe**
>
> *One thing that concerns me, for example, is the VTable description for IDataObject. The one listed at that site has 3 more public functions than what is documented on MSDN. Not only that, the 3 functions are intermixed. I can only assume that the IDataObject described is a different GUID/class than that listed on MSDN? Similar situation for IStream. This is where the GUID would be helpful.*

Isn't the information regarding the VTable order documented in the Windows header files? For instance, I believe the following is the IDataObject's definition from the ObjIdl.h header file:

Code:

```
#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("0000010e-0000-0000-C000-000000000046")
    IDataObject : public IUnknown
    {
    public:
        virtual /* [local] */ HRESULT STDMETHODCALLTYPE GetData(
            /* [unique][in] */ FORMATETC *pformatetcIn,
            /* [out] */ STGMEDIUM *pmedium) = 0;

        virtual /* [local] */ HRESULT STDMETHODCALLTYPE GetDataHere(
            /* [unique][in] */ FORMATETC *pformatetc,
            /* [out][in] */ STGMEDIUM *pmedium) = 0;

        virtual HRESULT STDMETHODCALLTYPE QueryGetData(
            /* [unique][in] */ __RPC__in_opt FORMATETC *pformatetc) = 0;

        virtual HRESULT STDMETHODCALLTYPE GetCanonicalFormatEtc(
            /* [unique][in] */ __RPC__in_opt FORMATETC *pformatectIn,
            /* [out] */ __RPC__out FORMATETC *pformatetcOut) = 0;

        virtual /* [local] */ HRESULT STDMETHODCALLTYPE SetData(
            /* [unique][in] */ FORMATETC *pformatetc,
            /* [unique][in] */ STGMEDIUM *pmedium,
            /* [in] */ BOOL fRelease) = 0;

        virtual HRESULT STDMETHODCALLTYPE EnumFormatEtc(
            /* [in] */ DWORD dwDirection,
            /* [out] */ __RPC__deref_out_opt IEnumFORMATETC **ppenumFormatEtc) = 0;

        virtual HRESULT STDMETHODCALLTYPE DAdvise(
            /* [in] */ __RPC__in FORMATETC *pformatetc,
```

If this information is already given away by Microsoft, then why should we have to rely on 3rd party sites?

Question: Can DispCallFunc be used to invoke an arbitrary member of a class running in another thread? Can it be used to raise an event (via a public Sub) of a class running in the main GUI thread by a routine running in another thread?

BTW, for those interested in a faster way of obtaining a string from a given pointer, check out the comparisons here.

```
On Local Error Resume Next: If Not Empty Is Nothing Then Do While Null: ReDim i(True To False) As
Currency: Loop: Else Debug.Assert CCur(CLng(CInt(CBool(False Imp True Xor False Eqv True)))):
Stop: On Local Error GoTo 0
```

My CodeBank Contributions

```
Declare Sub CrashVB Lib "msvbvm60" (Optional DontPassMe As Any)
```

**Reply With Quote**

#12

Nov 29th, 2014, 08:37 AM

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

| | |
|---|---|
| Join Date: | Oct 2007 |
| Location: | Beside Waldo |
| Posts: | 16,445 |

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Bonnie, depends on the class. Obviously the header files are of great use for people that know where to search for them & want to download them. But for custom classes, TLBs may be needed or vendor documentation. Me, it's not like I'm doing low-level stuff every day, so I go out & get the info when I need it. Another link that could be helpful is this one, again 3rd party, and the only way to ensure you have a listing of the most current definitions, no listing will be newer than the originator.

Questions: Are you asking if it's thread-safe? No documentation on MSDN indicates otherwise. If you are asking about cross-processes, then don't know, never had the need to try to attach to a class outside the immediate process. I think the problem here is scope. Pointers in another process do not point to same memory address of the current process. If an object can be transferred via OLE, then that may be a possibility if the O/S does the marshalling?

*Last edited by LaVolpe; Nov 29th, 2014 at 02:03 PM.*

**Insomnia** is just a byproduct of, "*It can't be done"*

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 30th, 2014, 01:35 AM #13

**Bonnie West**

Default Member

| | |
|---|---|
| Join Date: | Jun 2012 |
| Location: | InIDE |
| Posts: | 4,057 |

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **LaVolpe**
>
> *Are you asking if it's thread-safe?*

Somewhat like that. I was wondering whether it could be useful in this specific scenario: The main GUI thread instantiates & initializes a class that in turn spawns a background worker thread (via CreateThread or SHCreateThread or some other API). When the worker thread finishes up, it uses DispCallFunc to invoke a Sub exposed by its parent class and that Sub then raises a TaskComplete() event or something similar. There would be just 1 thread per class instance so the Synchronization APIs would probably not be necessary. It would be great if that's possible.

```
On Local Error Resume Next: If Not Empty Is Nothing Then Do While Null: ReDim i(True To False) As
Currency: Loop: Else Debug.Assert CCur(CLng(CInt(CBool(False Imp True Xor False Eqv True)))):
Stop: On Local Error GoTo 0
```

My CodeBank Contributions

```
Declare Sub CrashVB Lib "msvbvm60" (Optional DontPassMe As Any)
```

**Reply With Quote**

Nov 30th, 2014, 12:43 PM #14

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

| | |
|---|---|
| Join Date: | Oct 2007 |
| Location: | Beside Waldo |
| Posts: | 16,445 |

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Bonnie, if you build a test project and verify one way or the other, post back as others may find the results beneficial. Keep in mind to execute a Sub from an interface, the interface instance must be obtainable too and the VTable offset of the sub must be known in advance.

For custom classes/interfaces that support IDispatch, should be able to get this info via IDispatch if the sub Name is known.
- QueryInterface for IDispatch implementation
- IDispatch:GetIDsOfNames (VTable offset) called to retrieve the DispID of the sub's Name and its parameter info
- IDispatch:Invoke (VTable offset) would be called to execute that Sub
I've used IDispatch:GetIDsOfNames in the past. I've never called IDispatch:Invoke manually
For a truly complicated, twisted, project that used IDispatch:GetIDsOfNames and VB VTable parsing/hacking see this one which overrides usercontrol propertysheet items for customized enumerations

**Insomnia** is just a byproduct of, "*It can't be done"*

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Dec 1st, 2014, 02:58 AM #15

**Bonnie West**

Default Member

| | |
|---|---|
| Join Date: | Jun 2012 |
| Location: | InIDE |
| Posts: | 4,057 |

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **LaVolpe**
>
> *Bonnie, if you build a test project and verify one way or the other, post back as others may find the results beneficial.*

OK, I will! Thanks for the tips!

> Originally Posted by **LaVolpe**
>
> *For a truly complicated, twisted, project that used IDispatch:GetIDsOfNames and VB VTable parsing/hacking see this one which overrides usercontrol propertysheet items for customized enumerations*

Yeah, I've seen that project of yours. I'll take a look again and see if I can fully comprehend it this time!

```
On Local Error Resume Next: If Not Empty Is Nothing Then Do While Null: ReDim i(True To False) As
Currency: Loop: Else Debug.Assert CCur(CLng(CInt(CBool(False Imp True Xor False Eqv True)))):
Stop: On Local Error GoTo 0
```

My CodeBank Contributions

```
Declare Sub CrashVB Lib "msvbvm60" (Optional DontPassMe As Any)
```

Reply With Quote

Dec 1st, 2014, 11:43 AM #16

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

| | |
|---|---|
| Join Date: | Oct 2007 |
| Location: | Beside Waldo |
| Posts: | 16,445 |

### Re: [VB6] Call Functions By Pointer (Universall DLL Calls)

Bonnie, if this is a Sub from a VB class you want to call, and you know the method's name, far easier:
:: Copy pointer to uninitialized object and call the Object.SubName, remove pointer from that object, done

Now, for a bit of fun, let's do a simple example of calling the object's IDispatch.Invoke method instead
1) Create new class, leave default name & add these 2 methods

Code:

```
Public Sub TestMe()
    MsgBox "Test"
End Sub
Public Function TestMe2() As Long
    MsgBox "Test2" & vbNewLine & "vRtn should equal " & ObjPtr(Me)
    TestMe2 = ObjPtr(Me)
End Function
```

2) Within the form add a command button & this sample code. Ensure project also includes the attached cUniversalDLLCalls class

Code:

```
Private Type DISPPARAMS
    rgVarg As Long       ' pointer to array of variant agruments
    rgDispID As Long     ' pointer to array of DISPIDs of named argument
    cArgs As Long        ' number of arguments
    cNamedArgs As Long   ' number of named arguments
End Type

' sample of getting the IDataObject GUID
Private Sub Command1_Click()
    Const IUnknownRelease As Long = 8&
    Const IDispatchIDsOfNames As Long = 20&
    Const IDispatchInvoke As Long = 24&

    Dim IID_Dispatch As Long, lDispID As Long
    Dim aGUID(0 To 3) As Long, sName As String
    Dim DP As DISPPARAMS, vRtn As Variant
    Dim c As New cUniversalDLLCalls, tc As Class1

    Set tc = New Class1

    sName = "{00020400-0000-0000-C000-000000000046}" ' GUID for IDispatch
    c.CallFunction_DLL "ole32.dll", "IIDFromString", STR_NONE, CR_LONG, CC_STDCALL, StrPt
    ' ensure object implements IDispatch...
    c.CallFunction_COM ObjPtr(tc), 0&, CR_LONG, CC_STDCALL, VarPtr(aGUID(0)), VarPtr(IID_
    If IID_Dispatch <> 0& Then
        Erase aGUID()
        sName = "TestMe2" ' << change to another known method in the class
        Call c.CallFunction_COM(IID_Dispatch, IDispatchIDsOfNames, CR_LONG, CC_STDCALL, V
        If lDispID <> 0 Then
            Call c.CallFunction_COM(IID_Dispatch, IDispatchInvoke, CR_LONG, CC_STDCALL, l
            Debug.Print "return value if any: "; vRtn
        Else
```

change sName = "TestMe2" to sName = "TestMe" to test the other sub. Note: that if parameters exist, then this gets more difficult and the DISPPARAMS structure needs to be filled out completely

**Edited**: If wanting to pursue this further, here's a bit of help with the DISPPARAMS structure

1) declare a variable vArgs() As Variant, size it to the number of parameters the method expects
2) populate it in reverse order and ensure the correct vartype of each param, example:
    method's params: (Index As **Integer**, NewValue As **Long**)
    NewValue would be: vArgs(0) = 15&, CLng(15), just ensure Long vartype
    Index would be: vArgs(1) = 1%, CInt(1), just ensure Integer vartype
3) the structure members: DP.cArgs = nrArgs: DP.rgVarg = VarPtr(vArgs(0))
4) The 3rd from last parameter in that IDispatchInvoke sample call above is 1&. That is the constant for DISPATCH_METHOD, aka, vbMethod

So how do we NOT pass an optional parameter to a VB class? Per docs, we must pass Variant of type Error with a specific error code in place of the optional parameter. How that parameter is coded determines whether VB sees it as passed or not passed. The following shows how to NOT provide the final Optional parameter of a method. Remember last parameter is 1st in the array:

Code:

```
Const DISP_E_PARAMNOTFOUND As Long = &H80020004

vArgs(0) = DISP_E_PARAMNOTFOUND
' tweak to force variant type of Error. Can use CopyMemory API if desired, but...
c.CallFunction_DLL "kernel32.dll", "RtlMoveMemory", STR_NONE, CR_None, CC_STDCALL, _
                            VarPtr(vArgs(0)), VarPtr(vbError), 2&

' Note: Proper way would be to have VB give us a Variant of Error vartype, i.e.,
' vArgs(0)= CVErr(DISP_E_PARAMNOTFOUND) but that errors
```

Ok, that was fun. If anyone is asking what's difference between a parameter and a named parameter? See this MSDN link. If you wish to use named parameters/arguments with IDispatch, you have to get the DISPID of each named argument when you made the call to DispatchIDsOfNames, passing a array of names & a return array for the DISPIDs. These are then populated separately in the DISPPARAMS members. Here is a bit more info. Bottom line, more work but doable.

Last but not least, haven't tried messing with methods that take ParamArrays, I'll let someone else play with that if they wish

*Last edited by LaVolpe; Dec 1st, 2014 at 05:43 PM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

Reply With Quote

Dec 2nd, 2014, 03:43 AM #17

### Re: [VB6] Call Functions By Pointer (Universall DLL Calls)

> Originally Posted by **LaVolpe**
>
> *Bonnie, if this is a Sub from a VB class you want to call, and you know the method's name, far easier:*
> *:: Copy pointer to uninitialized object and call the Object.SubName, remove pointer from that object, done*

But, would that work across threads (specifically, from a thread where the VB6 runtime environment hasn't been initialized)?

**Bonnie West**
Default Member

Join Date: Jun 2012
Location: InIDE
Posts: 4,057

Anyway, thanks again for your examples and links! I'm sure they'll prove invaluable in helping me figure out a solution to this multi-threading problem.

```
On Local Error Resume Next: If Not Empty Is Nothing Then Do While Null: ReDim i(True To False) As
Currency: Loop: Else Debug.Assert CCur(CLng(CInt(CBool(False Imp True Xor False Eqv True)))):
Stop: On Local Error GoTo 0
```

My CodeBank Contributions

```
Declare Sub CrashVB Lib "msvbvm60" (Optional DontPassMe As Any)
```

Reply With Quote

Dec 6th, 2014, 03:02 PM #18

**LaVolpe**
**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **Bonnie West**
>
> *But, would that work across threads (specifically, from a thread where the VB6 runtime environment hasn't been initialized)?*

You can initialize COM onto the thread via OleInitialiize or CoInitialize/Ex. Don't know if that helps or not.

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

Reply With Quote

Dec 6th, 2014, 04:01 PM #19

**Schmidt**
PowerPoster

Join Date: Jun 2013
Posts: 3,105

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **LaVolpe**
>
> *You can initialize COM onto the thread via OleInitialiize or CoInitialize/Ex. Don't know if that helps or not.*

That doesn't help - unless the COM-Class (or -Dll) supports (within its hidden ClassFactory-creation-routine) so called MTAs
(MultiThreaded Apartments) ... VB-generated COM-Dlls do not support this mode - the best you can switch on in our
VB-Compiler (when creating ActiveX-dlls) is "Apartment-Threaded" - which will support at least the creation of COM-
instances on different STAs (within the same Process) - this is due to "legacies" into the (transactional threading) of DCOM and COM+ hosting-processes, where VB-Classes (VB-generated COM-Dlls) once played a major role.

Such a started STA (which fully initializes its ThreadLocalStorage only, when you e.g. create a VB-Class-instance on the new Thread in question) runs basically with an "isolated Memory-Allocator", which e.g. creates separate allocations for all the Public-Variables which are defined in *.bas-Modules inside a given VB-Dll-Project anew - on each new STA-Thread which sees such a new (first) ClassInstance of a given VB-Dll (one can log that e.g. with an appropriate ThreadID-writing function in the Dlls Sub Main, which is also jumped into for each STA-thread anew, on first instantiation of a VBClass from a given Dll on this new given STA-thread...

So the TLS-issues is the main-culprit for the unpredictable behaviour (and crashes) we see, when we want to talk "across STAs directly" - that's also the reason why we see problems, when VB-defined callbacks are jumped into from System-APIs which run on different (System-)Threads.

MTAs (which VB doesn't support) are far more robust in this regard - but then you will have to handle Thread-synchronizing yourself again.

So, to talk across STAs in a safe way, one has to use the System-provided Marshaling-Functions for that:
CoMarshalInterThreadInterfaceInStream -> http://msdn.microsoft.com/en-us/libr...=vs.85%29.aspx
CoGetInterfaceAndReleaseStream -> http://msdn.microsoft.com/en-us/libr...=vs.85%29.aspx
which will ensure a synchronous communication - as well as serialization of any passed Params into complete Data-Copies - with each
Method-call to the other Target-Interface in question.

For a bit more flexibility I've implemented my own Variant-Array-based serialization-routines for the STA-Threading-Support
in vbRichClient5 (which communicates over Pipes, achieving a somewhat better performance than normal COM-marshaling this way).

Olaf

Reply With Quote

Dec 6th, 2014, 06:59 PM #20

**LaVolpe**
**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Take your word for it, however, I believe it can be done just don't know the details. Several years ago, was playing with VB DLL injection and got it to work for the most part. The idea was to catch VB's compilation and tweak the switches to compile the DLL as standard vs active-x. But, as you know, still can't use VB commands/objects for the most part, but a subset can be used with help of TLBs for APIs and String constants. The injection worked but was too cumbersome for me to continue with it. In the process of trying to figure out how to use VB in all of its glory in a thread that wasn't initialized for it, came across another coder that made it work using pure VB. He wouldn't release his secret, but did give me hints. Among those hints was timing & getting COM on the thread. Sorry for the vagueness, this was 8-10 years ago.

Above being said, I personally have no desire to re-visit that challenge. And hopefully, your insight shines a bit of light for Bonnie

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Dec 8th, 2014, 03:16 AM #21

**Bonnie West**
Default Member

Join Date: Jun 2012
Location: InIDE
Posts: 4,057

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Thanks a lot for the additional info, guys!

I've been doing some research about this lately and unfortunately, it appears it's impossible to safely use the DispCallFunc API for cross-thread method invocation. According to The Rules of the Component Object Model:

> Originally Posted by **MSDN**
>
> ***Apartment Threading Model***
>
> *The details of apartment-model threading are actually quite simple, but must be followed carefully, as follows:*
>
> *Every object lives on a single thread (within a single apartment).*
>
> - *All calls to an object must be made on its thread (within its apartment). It is forbidden to call an object directly from another thread. Applications that attempt to use objects in this free-threaded manner will likely experience problems that will prevent them from running properly in future versions of the operating systems. The implication of this rule is that all pointers to objects must be marshalled between apartments.*

The Inside COM+ e-book mentioned by LaVolpe on the other (VBForums) thread also states basically the same thing:

> Originally Posted by **Inside COM+**
>
> ***Single-Threaded Apartments***
>
> *The STA model owns any COM+ objects instantiated by its thread, so all method calls on the object are executed by the thread that created the object. The fact that the same thread that created an object is always used to execute its methods is important to objects that have thread affinity; an object that requires thread affinity must be executed by a particular thread. For example, some objects use thread local storage (TLS) to associate data with a specific thread. An object using TLS expects that the same thread will be used to execute all of its methods. If a different thread executes a method, any attempt to access TLS data will fail. Because objects running in the MTA or NA model can be executed on a variety of different threads, they cannot use TLS. Only objects running in an STA may have thread affinity.*

Oh well, it's too bad DispCallFunc didn't turn out to be as useful in this regard as I had hoped. Sorry for cluttering your thread...

**EDIT**

> Originally Posted by **LaVolpe**
>
> *... came across another coder that made it work using pure VB.*

Could it be this guy? Advanced VB6 DLL Construction by Mathimagics

*Last edited by Bonnie West; Dec 8th, 2014 at 03:45 AM.*

```
On Local Error Resume Next: If Not Empty Is Nothing Then Do While Null: ReDim i(True To False) As
Currency: Loop: Else Debug.Assert CCur(CLng(CInt(CBool(False Imp True Xor False Eqv True)))):
Stop: On Local Error GoTo 0
```

My CodeBank Contributions

```
Declare Sub CrashVB Lib "msvbvm60" (Optional DontPassMe As Any)
```

**Reply With Quote**

Dec 8th, 2014, 07:26 AM #22

**LaVolpe**
**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **Bonnie**
>
> *Could it be this guy? Mathimagics*

That's him. I noticed he updated his thread in 2007. Maybe he has now released all his insight? I stopped pursuing that line of logic before 2007, but dang, now I'll have to go and download his documentation in case it shows me where I went wrong way back then... Thanx a ton Bonnie, was happily content before & now I'll be tortured again. jk

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Dec 10th, 2014, 07:32 AM #23

**Jonney**
Frenzied Member

Join Date: Jan 2010
Posts: 1,083

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **LaVolpe**
>
> *That's him. I noticed he updated his thread in 2007. Maybe he has now released all his insight? I stopped pursuing that line of logic before 2007, but dang, now I'll have to go and download his documentation in case it shows me where I went wrong way back then... Thanx a ton Bonnie, was happily content before & now I'll be tortured again. jk*

I have few links but website is in Chinese about Hook ANY API Function. I don't know whether you can understand.

I sms you.

*Last edited by Jonney; Dec 10th, 2014 at 07:44 AM.*

**Reply With Quote**

Jan 10th, 2015, 09:51 AM #24

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

TIP: If by any chance, you wish to call a private method on a COM interface/class (i.e., a VB class, form, usercontrol, etc) you can, but the following modification needs to be made:

1) Within the CallFunction_COM method, you will need to remove the test for the InterfacePointer parameter being zero
2) Obviously, you will need to know or somehow locate the target private method function pointer/address

The end result, is that the call to the private method is treated just like a call to a standard DLL, with the additional requirement of passing the COM ObjPtr as the 1st parameter of that method. Example

Code:

```
Debug.Print myClass.CallFunction_COM (0&, [private function address], STR_NONE, CR_LONG, _
                        CC_STDCALL, ObjPtr(ComInterface), [any private method parameters])
```

Edited: See post #35 below for more details.
We do not need to necessarily know the function pointer of the class instance and its target method, but we definitely need to know where in the vTable the function exists. Let's say we have a simple class that has 10 methods in it and we want to call the last private method. The 10th method would be offset 36 (10*4-4) from the 1st method. We know that the 1st offset in the class will be &H1C or 28 (see post #35 on how we know this). So the adjusted vTable for the 10th method is: 28 + 36 = 64. To call that 10th method:

Code:

```
myClass.CallFunction_COM(ObjPtr(theClass), 64&, STR_NONE, CR_LONG, CC_STDCALL, [any methoc
```

*Last edited by LaVolpe; Feb 22nd, 2016 at 02:15 PM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 28th, 2015, 06:38 AM #25

**Thierry76**

Junior Member

Join Date: Apr 2014
Posts: 25

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

sorry if my question is stupid, but I am a beginner in this sort of loo level coding, and I started with Learning with you 🙂

I try in vain to call the EnumElements form a IStorage......

I there any body who could help me a little ?

Code:

```
Private Sub Command2_Click()
   Dim strFilename            As String
   Dim objStorage             As stdole.IUnknown   ' oleexp3.IStorage
   Dim reserved1 As Long, reserved2 As Long, reserved3 As Long
   Dim objESE                 As stdole.IUnknown
   Dim lngESE                 As Long
   Dim lngResult              As Long

   Dim c                      As cUniversalDLLCalls

   strFilename = "C:\OLEObject\oleObject2.bin"

   Set c = New cUniversalDLLCalls
   If StgIsStorageFile(StrPtr(strFilename)) = S_OK Then

      Debug.Assert 0

      lngResult = StgOpenStorage(StrPtr(strFilename), ByVal 0&, _
                           STGM_DIRECT Or STGM_SHARE_EXCLUSIVE Or STGM_READWRITE, _
                           ByVal 0&, ByVal 0&, objStorage)
      If lngResult = S_OK Then
         '-----------------------------------------------------------------------------
         ' METHOD EnumElements ( _                               ' VTable offset = 44
         '      BYVAL prm_reserved1 AS LONG _                       ' [in] reserved1 VT_I4
         '      , BYVAL prm_reserved2 AS DWORD _                     ' [in] *reserved2 VT_
         '      , BYVAL prm_reserved3 AS LONG _                      ' [in] reserved3 VT_I
         '      , BYREF prm_ppenum AS VBStorageIStrorage _         ' [in][out] **ppenum ISt
         '      )
         '-----------------------------------------------------------------------------

         '-----------------------------------------------------------------------------
         'Olaf Schmit way
```

Regards
Thierry

**Reply With Quote**

Nov 28th, 2015, 09:18 AM #26

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Thierry. What is the problem? Do you get an object assigned to objESE, i.e., ObjPtr(objESE) <> 0 ?

Edited: Don't know why you are testing whether the objStorage object implements IStorage? StgOpenStorage returns the objStorage which is IStorage. So, if you remove that portion of the code, replace IID_Storage with ObjPtr(objStorage) when retrieving the IEnumStatStg object.

objESE is another interface. That interface has methods that must be called to enumerate the IStorage. objESE is a IEnumSTATSTG interface. Its VTable order, after IUnknown, is:
Next, offset of 12
Skip, offset of 16

Reset, offset of 20
Clone, offset of 24

The link I provided describes each of the IEnumSTATSTG methods.

P.S. Remove the VarPtr() from the 1st 3 parameters. That is likely part of the problem. With my function, VarPtr() is used for ByRef, [in/out], [out] type of parameters only. The final parameter is correct in using VarPtr(). As is, instead of passing 0, 0, 0 you are passing 0 variable pointer, 0 variable pointer, 0 variable pointer all of which will be non-zero.

If you have questions on how to use IEnumSTATSTG, please post it in the questions portion of the forum. Myself, Olaf, Fafalone & others are likely to reply

*Last edited by LaVolpe; Nov 28th, 2015 at 10:34 AM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 28th, 2015, 10:53 AM #27

**Thierry76**
Junior Member
Join Date: Apr 2014
Posts: 25

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Thanks a lot for your quick and cleaver answer...
At this time, I was googling a lot... and found threads from you, Olaf, Fafalone, The Trick... I am far to understand all I read, but I love what I found...
First I need to go step by step to understand more deeper the out-of-the-word samples you give us....

When I will succes, promise I will share the solution I found.... but fisrt I need to learn more before post stupid question 😊

NB I know the "IEnumSTATSTG interface".... all my work is for a code share here that I try to make better (with no Tlb referece) => http://www.vbforums.com/showthread.p...-in-Office2010

*Last edited by Thierry76; Nov 28th, 2015 at 10:59 AM.*

**Reply With Quote**

Nov 28th, 2015, 10:57 AM #28

**LaVolpe**
**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **Thierry76**
>
> *Thanks a lot for your quick and cleaver answer... but fisrt learn before post stupid question* 😊

Not a stupid question & glad you posted code, otherwise, we could've gone around in circles if I didn't see you using VarPtr(), incorrectly, with my function. Assuming Olaf's & my functions work identically was a just a tad bit foolish 🙄. Regarding the QueryInterface call, no harm, no foul; just not needed.

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Nov 29th, 2015, 10:27 AM #29

**Thierry76**
Junior Member
Join Date: Apr 2014
Posts: 25

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

I have one question for the experts... All the methods in MS Interfaces I used return a vbLong ( alias VT_I4)... is it alvway the case ? ? ? it can make my code more simple ... 😕

*Last edited by Thierry76; Nov 29th, 2015 at 11:15 AM.*

**Reply With Quote**

Feb 19th, 2016, 06:40 AM #30

**jsvenu**
Member
Join Date: Apr 2015
Posts: 50

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> Originally Posted by **LaVolpe**
>
> *TIP: If by any chance, you wish to call a private method on a COM interface/class (i.e., a VB class, form, usercontrol, etc) you can, but the following modification needs to be made:*
>
> *1) Within the CallFunction_COM method, you will need to remove the test for the InterfacePointer parameter being zero*
> *2) Obviously, you will need to know or somehow locate the target private method function pointer/address*
>
> *The end result, is that the call to the private method is treated just like a call to a standard DLL, with the additional requirement of passing the COM ObjPtr as the 1st parameter of that method. Example*
>
> *Code:*
>
> ```
> Debug.Print myClass.CallFunction_COM (0&, [private function address], STR_NONE, CR_LONG, _
>                         CC_STDCALL, ObjPtr(ComInterface), [any private method parameters]
> ```

Dear Lavolpe,

I tried to hide form 2 in standard exe project.It hides if I use tc.hide directly where tc is form2 object.
**But when I try to hide form2 using CallFunction_COM giving 516 as the address of Hide method 'ie, vtableoffset as per interface _Form which derives from IDispatch located in vb6.olb in vb98 folderwhen we view the typelib using oleview.Hide is the 122th method+3(for IUnknown)+4(for IDispatch)=129*4bytes=516.**
**the Hide method of the form2 is not called.**
Can you clarify me where I am doing wrong.
I am attaching the sample standard exe project.I have used your cUniversalDLLCalls class from prjUniDLLcalls.zip sample.

regards,
jsvenu

Attached Files
Mycallcom.zip (10.6 KB, 58 views)

**Reply With Quote**

Feb 20th, 2016, 01:47 PM #31

**LaVolpe**

**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Ok, here's what I see.

When you are trying to call a public method/property from IDispatch, you do not need any offsets. You simply need the dispatch ID. You almost got the right mix with these two attempts 1) per your zip and 2) per your PM to me. The blue highlighted range of characters is the difference between the two. #2 is more correct & its fixes are described below.

Code:

```
1. Call c.CallFunction_COM(IID_Dispatch, IDispatchInvoke, CR_LONG, CC_STDCALL, ObjPtr(tc),
2. Call c.CallFunction_COM(IID_Dispatch, IDispatchInvoke, CR_LONG, CC_STDCALL, lDispID, Va
```

To get it working, two things are needed, with statement #2 above.

1. Elsewhere in your code, change 516& below to 0&. You want to call QueryInterface, offset of zero, to get the IDispatch interface pointer

Code:

```
c.CallFunction_COM ObjPtr(tc), 516&, CR_LONG, CC_STDCALL, VarPtr(aGUID(0)), VarPtr(IID_Di
```

2. The return item from IDispatch is Variant. You cannot pass VarPtr(vRtn). Just use vRtn. Passing VarPtr() will have the class interpret the parameter as Long not Variant. This, alone, will prevent the call from working

Where the offsets can come into play is if you are trying to call public or private methods of a class/form, etc, relative to the object's VTable. But that is far more difficult and generally never needed unless attempting to call a private method that is not exposed by IDispatch.

Edited:
FYI: VB's CallByName is a wrapper, of sorts, for IDispatch.Invoke. This would have worked also...
CallByName tc, "Hide", vbMethod

Also note that if IsObject(someObj)=True then ObjPtr(someObj) returns the IDispatch pointer. Your code could be reduced quite a bit to the following. Also know that lDispID can be negative or zero & testing for <> 0 is not quite correct.

Code:

```
    sName = "Hide" ' << change to another known method in the class
    Call c.CallFunction_COM(ObjPtr(tc), IDispatchIDsOfNames, CR_LONG, CC_STDCALL, VarPtr(a
    If lDispID <> 0 Then
        Call c.CallFunction_COM(ObjPtr(tc), IDispatchInvoke, CR_LONG, CC_STDCALL, lDispID,
    End If
```

*Last edited by LaVolpe; Feb 20th, 2016 at 05:11 PM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Feb 22nd, 2016, 06:10 AM #32

**jsvenu**

Member

Join Date: Apr 2015
Posts: 50

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Dear Lavolpe,

Thankyou for the reply.It worked fine.

But regarding VTables,

**Where the offsets can come into play is if you are trying to call public or private methods of a class/form, etc, relative to the object's VTable. But that is far more difficult and generally never needed unless attempting to call a private method that is not exposed by IDispatch.**

I have gone thru how to replace virtual table of a class method Method1 with another

method Method2 of the same class using SwapMethods method.I am replacing Method1 public

method with Method2 method of the same class.The sample works fine.

But when I use the same code for main form form1 as well as new form form2 public

methods instead of class methods as above the code does'nt work.Please clarify whether

there is any difference between form and class methods in terms of their vtable and where I

am doing wrong.I think both the class and forms derive from IDispatch interface.For class

&H1C(i.e,**4*8**) is the offset to first vtable public method.But when I use the same for forms it

doen'st work.I also used (146+3+4=153)*4=612& since already 145 entries are there in interface _form and then I use 146 as the next entry.But it didnt work.How can I do the same for forms.I am attaching code class_vs_form_vtables.zip which is standard exe project.

regards,
jsvenu

Attached Files
class_vs_form_vtables.zip (4.3 KB, 56 views)

Reply With Quote

Feb 22nd, 2016, 06:28 AM #33

jsvenu
Member
Join Date: Apr 2015
Posts: 50

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Dear Lavolpe ,

Sorry it is &H1C (decimal 28 or 4*7) since IUnknown+IDispatch(3+4=7*4) and I misspelled as 4*8=32 for the class
Byte offset of first index in Virtual Function Table.

regards,
jsvenu

Reply With Quote

Feb 22nd, 2016, 06:35 AM #34

jsvenu
Member
Join Date: Apr 2015
Posts: 50

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Dear Lavolpe,

You said that **is far more difficult and generally never needed unless attempting to call a private method that is not exposed by IDispatch**
Can you give me example of such private methods which are not exposed by IDispatch along with the above clarification with how to find
vtable offset of both types(IUnknown and IDispatch exposed interfaces) .

regards,
jsvenu

Reply With Quote

Feb 22nd, 2016, 08:13 AM #35

LaVolpe
Thread Starter
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

The offsets you will want are in multiples of 4 as expected. For a class, these start at &H1C, but not always. The following are relative to IDE, uncompiled, and should be verified when compiled to see if things change...

1. VB places Public methods first in the VTable, then private and friend methods. The sort order, from some quick tests, are: Public methods first, in same order listed in the class. Then private and friend, in same order listed in the class. It appears, VB makes no distinction between friend and private, as far as sorting goes, when creating the vTable for the class

2. If the class contains any public variable declarations, i.e., Public mOwner As Long, then the first item in the class is offset and no longer at &H1C. From tests, it appears each Public variable declared at top of the class offsets the first method in the class by 8 bytes or 12 bytes. VB creates a Property Get/Let for such public variables. 8 bytes for non-Object/Variant variables (Get/Let), 12 bytes for Object/Variant variables (Get/Let/Set). These appear to be in order declared and the Property Get is before Property Let/Set in the vTable.

As you've stated, the class with no public variables declared, starts at offset &H1C. Forms start at &H6F8, this includes mdi forms, mdi child forms, as well as 'normal' forms.

Using a class as an example, with no public declared variables

Code:
```
Private Sub Test1()
    Debug.Print "got private sub"
End Sub
Public Sub Test2()
    Debug.Print "got public sub"
End Sub
Friend Sub Test3()
    Debug.Print "got friend sub"
End Sub
```

A call might look like the following... where o is an instance of the test class above. Notice the order of the subs above and the order that was called after the code below is executed

Code:
```
Call c.CallFunction_COM(ObjPtr(o), &H1C, CR_LONG, CC_STDCALL)
Call c.CallFunction_COM(ObjPtr(o), &H1C + 4&, CR_LONG, CC_STDCALL)
Call c.CallFunction_COM(ObjPtr(o), &H1C + 8&, CR_LONG, CC_STDCALL)

' the print out would look like:
got public sub
got private sub
got friend sub
```

As you can see, calling VB objects by vTable is far more difficult and absolutely requires knowledge of the vTable order. Personally, I would not use my class for calling public methods/properties that are accessible from IDispatch.

Use a class instance directly or indirectly via VB's CallByName. Calling private methods of a class can be a niffty way of communicating with the class without making the method public or friend, but I'd imagine only a few scenarios may exist where you want to obfuscate in this manner.

Edited: Adding Implements to the class offsets the first method also. Once you know the structure of the class and it is finally set, then your offsets can be better determined. Adding, removing, or moving methods within the class change offsets.

Here is a routine that can help. **If it fails, it will crash**. Use for testing only...
1. Pass to it an instantiated VB-only code object: form, class, usercontrol
:: note: from outside the usercontrol, use ObjPtr(UserControl[xxx].Object)
:: from within the usercontrol, use ObjPtr(Me)
2. That object must have at least one private, public or friend method/property **else crash**
3. The debug.print statement will show the vTable offset where the first method occurs

Code:

```
Private Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destinatio

Private Sub zProbe(o As Object)

    Dim nStart As Long, nAddr As Long
    Dim nEntry As Long, bSig As Byte

    If o Is Nothing Then Exit Sub
    CopyMemory nStart, ByVal ObjPtr(o), 4&
    nAddr = nStart
    Do
        CopyMemory nEntry, ByVal nAddr, 4&
        If nEntry <> 0& Then
            CopyMemory bSig, ByVal nEntry, 1&
            If bSig = &H33 Or bSig = &HE9 Then ' native or pcode signature
                Debug.Print "first method from vTable "; nStart; " is "; nAddr - nStart
                Exit Do
            End If
        End If
        nAddr = nAddr + 4&
    Loop

End Sub
```

*Last edited by LaVolpe; Feb 22nd, 2016 at 04:46 PM.*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Feb 23rd, 2016, 06:32 AM #36

**jsvenu**
Member
Join Date: Apr 2015
Posts: 50

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Dear Lavolpe,
Thankyou very much for the awesome reply.Now swapping works fine.
In the following test code given I want to know

**1.How to differentiate between public,private members/methods.**
**2.If I comment Exit Do and run the code how to make it give all total members/methods without crashing.Here &H33 and &HE9 are used.How to know about this constants.**
**3.You said that is far more difficult and generally never needed unless attempting to call a private method that is not exposed by IDispatch.What are those private methods not exposed by IDispatch.**

Private Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (ByRef Destination As Any, ByRef Source As Any, ByVal Length As Long)

Private Sub zProbe(o As Object)

Dim nStart As Long, nAddr As Long
Dim nEntry As Long, bSig As Byte

If o Is Nothing Then Exit Sub
CopyMemory nStart, ByVal ObjPtr(o), 4&
nAddr = nStart
Do
CopyMemory nEntry, ByVal nAddr, 4&
If nEntry <> 0& Then
CopyMemory bSig, ByVal nEntry, 1&
If bSig = &H33 Or bSig = &HE9 Then ' native or pcode signature
Debug.Print "first method from vTable "; nStart; " is "; nAddr - Start
Exit Do
End If
End If
nAddr = nAddr + 4&
Loop

End Sub

regards,
jsvenu

**Reply With Quote**

Feb 23rd, 2016, 07:29 AM #37

**LaVolpe**
**Thread Starter**
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

> *If I comment Exit Do and run the code how to make it give all total members/methods without crashing.Here &H33 and &HE9 are used.How to know about this constants.*

Here is a project on planet-source-code site that was developed by Paul Caton. The snippet I provided for getting the code offset comes from that project. In that project, you can see how Paul used IsBadCodePtr API to locate the final method of the code page. That project may help understanding the logic. As far as where the bSig constants come from, I assume that was the result of Paul decompiling and/or debugging compiled code. Those constants were only used to locate the initial offset of the code page. Paul used another routine to take that starting point and scan memory, from that starting point, to locate the final method. Note that those constants are likely applicable to VB-code pages only. I doubt that logic will work on all COM objects.

> *How to differentiate between public,private members/methods*

Not sure I understand the question. If you know how many methods exist in the code page and how many methods are exposed by IDispatch, then the difference is the private/friend method count. To get the number of public methods from IDispatch might be doable by first getting the ITypeInfo interface. The ITypeInfo interface can be retrieved via IDispatch::GetTypeInfo. From that you can call ITypeInfo.GetTypeAttr which fills in a TYPEATTR structure. In that structure are the counts you are curious about. I would be curious if TYPEATTR.cbSizeVft returns the size of the vTable with or without private methods

Edited: Note that Friend methods ,within VB only, can be exposed when an object is early bound, not late bound (i.e., something declared as generic Object). Friend functions are not exposed via IDispatch.

> *What are those private methods not exposed by IDispatch.*

All private/friend methods of the code page are not exposed by IDispatch. Only public methods are exposed. There is no way that I know of that will allow you to determine the parameter information or return types of any private/friend methods.

In any case, it is usually a requirement to know the vTable in advance before modifying it or calling methods from it. Trying to discover the vTable count (including private methods) and method details of each method associated with the vTable, on the fly, as far as I know, is not possible. If it is possible, you'll likely find that on sites that discuss hacking and VB decompiling.

Edited: Follow-up based solely on curiosity...
Get the ITypeInfo interface, then the TYPEATTR structure

Code:

```
Private Type TYPEATTR
    guid(0 To 3)            As Long
    lcid                    As Long
    dwReserved              As Long
    memidConstructor        As Long
    memidDestructor         As Long
    pstrSchema              As Long
    cbSizeInstance          As Long
    TYPEKIND                As Long
    cFuncs                  As Integer
    cVars                   As Integer
    cImplTypes              As Integer
    cbSizeVft               As Integer
    cbAlignment             As Integer
    wTypeFlags              As Integer
    wMajorVerNum            As Integer
    wMinorVerNum            As Integer
    tdescAlias              As Long
    idldescType             As Long
End Type

.... test code, o is a test class
Dim ITInfo As IUnknown
Dim pAttrs As Long, uTA as TYPEATTR

    ' offset 16 = IDispatch.GetTypeInfo
    c.CallFunction_COM ObjPtr(o), 16&, CR_LONG, CC_STDCALL, 0&, 0&, VarPtr(ITInfo)
    If Not ITInfo Is Nothing Then
         ' offset 12 = ITypeInfo.GetTypeAttr
         c.CallFunction_COM ObjPtr(ITInfo), 12&, CR_LONG, CC_STDCALL, VarPtr(pAttrs)
         If pAttrs <> 0 Then
             CopyMemory uTA, ByVal pAttrs, LenB(uTA)
```

Using the above code in a couple different scenarios, the uTA members show this:
1) Standard class with just 1 public method
:: uTA.cFuncs = 1: uTA.cbSizeVft = 32 (IUnknown+IDispatch+1 method -- [3+4+1]*4 = 32 )
2) Standard class with 1 public & 1 private method
:: uTA.cFuncs = 1: uTA.cbSizeVft = 36 (IUnknown+IDispatch+2 methods -- [3+4+2]*4 = 36 )
3) Standard class with 1 public & 1 private method & 1 publicly declared variable
:: uTA.cFuncs = 3: uTA.cbSizeVft = 44 (IUnknown+IDispatch+2 methods+1 Get/Let-- [3+4+2+1+1]*4 = 44
:: note that uTA.cFuncs changed because of the VB-generated public Get/Let for the variable

Observations:
1. The uTA.cbSizeVft appears to be correct size to include all methods
2. The uTA.cFuncs includes only public methods, including VB-generated Get/Let/Set
3. If adding an Implements statement to the class, the uTA.cbSizeVft increases, but other key uTA members do not change. uTA.cbSizeInstance does change.
4. With a form, not a class, just 1 private method. uTA.cbSizeVft returns 1788 = &H678+4

Getting the offset for the final method in the class, assuming all Public methods are listed first in the class, is fairly straightforward: uTA.cbSizeVft - 4

Getting the 1st method's offset using ITypeInfo is not so straightforward. If no public methods exist at all, then don't see how ITypeInfo will help. Paul Caton's logic may be the only way.

1. If no public variables are declared, then it is straightforward.
- Call ITypeInfo.GetFuncDescr for function index 0
- Read returnPointer+28 into an Integer and that is the vTable offset
- Release the pointer

2. If public variables exist then I don't see a way to locate the first public method unless that first method is NOT a property, i.e., a public sub or function. Otherwise, the first function offset will be the Property Get of the 1st declared public variable. If first method is a public function or sub, then:
- Call ITypeInfo.GetFuncDesc in a loop from 0 to uTA.cFuncs-1 (see notes above)
- Read returnPointer+16 into a Long and that determines if method is property or not
- If method type is a property (not = 1), release pointer & continue looping
- Else read returnPointer+28 into an Integer and that is the vTable offset, release pointer, exit loop

*Last edited by LaVolpe; Feb 24th, 2016 at 09:19 AM. **Reason:** had offset incorrect, fixed*

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

**Reply With Quote**

Feb 25th, 2016, 07:37 AM #38

**jsvenu**
Member

| | |
|---|---|
| Join Date: | Apr 2015 |
| Posts: | 50 |

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Dear Lavolpe,
Thank you very much for the offset correction by editing and thunks reply.
I have gone thru your reply about thunks.
I wrote one thunk for calling form object member function using redirection from a function called thru addressof operator since we cannot call member functions directly using **addressof**.
I am unable to get my form object Friend Function WndProc to run even though I use thunking .
I am sending the project zip as attachement.
I took the byte codes in pushparamthunk UDT object thunk .
To execute the asm byte codes in pushparamthunk UDT I use **CallwindowProc**.Is this the only way to execute asm.Are there any other methods.
Am I missing something because when I run the code the app **crashes**.

Please clarify.
regards,
jsvenu

Attached Files
vbproj.zip (3.1 KB, 53 views)

Reply With Quote

Feb 25th, 2016, 08:16 AM #39

LaVolpe
Thread Starter
VB-aholic & Lovin' It

Join Date: Oct 2007
Location: Beside Waldo
Posts: 16,445

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

I do not know if your ASM is correct, don't know if stack corruption is occurring or not. The flow seems simple enough, create ASM that will push the objptr on the stack along with the subclass parameters, then forward to the static module function address. Whether your ASM is doing this correctly or not, I do not know.

Since your questions no longer apply to this thread (the class posted at top of thread), you may want to post your questions in the VB forum, not the codebank. Here are some possible solutions. But I don't want to use this thread to discuss subclassing techniques.

Google for these key terms: self-subclass paul caton. Any hits that are on planetsourcecode are what you want. Paul has produced reliable classes that are templates for subclassing to methods within any class, form, usercontrol, etc. Those classes already include the ASM and the ASM source, if interested in reviewing/modifying it.

Subclassing to a private method in form or class can be done a bit easier if you are already using a module. May want to look at this thread

**Insomnia** is just a byproduct of, "*It can't be done*"

Classics Enthusiast? Here's my 1969 Mustang Mach I Fastback. Her sister '67 Coupe has been adopted

**Newbie? Novice? Bored? Spend a few minutes browsing the FAQ section of the forum.**
**Read the HitchHiker's Guide to Getting Help on the Forums.**
**Here is the list of TAGs you can use to format your posts**
**Here are VB6 Help Files online**

{Alpha Image Control} {Memory Leak FAQ} {GDI+ Classes/Samples} {Unicode Open/Save Dialog} {Icon Organizer/Extractor}
{VB and DPI Tutorial} {XP/Vista Manifest Creator} {UserControl Button Template} {stdPicture Render Usage}

Reply With Quote

Feb 25th, 2016, 10:16 AM #40

jsvenu
Member

Join Date: Apr 2015
Posts: 50

**Re: [VB6] Call Functions By Pointer (Universall DLL Calls)**

Dear Lavolpe,
Thankyou for the response.
Here is the asm code for pushparamthunk

```
'asm code
push [esp]
mov eax, 1234h // Dummy value, pushed parameter
mov [esp + 4], eax
nop // nop Adjustment so the next long is aligned
nop
nop
mov eax, 5678h // Dummy value, destination function
jmp eax
nop
nop
```

I generated byte codes thru the use of vc++ win32 app in which I dissembled the asm and took code bytes thru the vc++ IDE for the
code

```
WinMain(...)
{
_asm
{
'asm code
push [esp]
mov eax, 1234h // Dummy value, pushed parameter
mov [esp + 4], eax
nop // nop Adjustment so the next long is aligned
nop
nop
mov eax, 5678h // Dummy value, destination function
jmp eax
nop
nop

}
}
```

regards,
jsvenu

Reply With Quote

Page 1 of 2 **1** 2 Last

Quick Navigation **CodeBank - Visual Basic 6 and earlier** **Top**

« Previous Thread | Next Thread »

VBForums VBForums CodeBank CodeBank - Visual Basic 6 and earlier [VB6] Call Functions By Pointer (Universall DLL Calls)

**Posting Permissions**

You may not post new threads
You may not post replies
You may not post attachments
You may not edit your posts

**BB code** is On
**Smilies** are On
**[IMG]** code is On
**[VIDEO]** code is On
HTML code is Off

**Forum Rules**

Contact Us VB Forums Top

Learn to code. $105K avg salary. No cost until you're hired.
APPLY NOW