AxSuite3

ActiveX/COM support library for FreeBASIC , Based on Jose Roca Code ActiveX/COM Programming.
AxSuite3 : is an extension of AxSuite2

Disclaimer:

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Feature:

- Automation/Native Dispatch Call (Axsuite)
- vTable Call for ActiveX/COM that has dual interface (much faster than Invoke call)
- Event Sink, for ActiveX/COM event generated programming

Requirement:

- ATL.dll (or ATL71.dll) needed to host ActiveX Control and some utility functions
- FreeBASIC Compiler, IDE (FBEdit Custom Control included), tested on Win98/XP

AxPackage.zip:

- AxSuite3.pdf : this document
- AxSuite3.exe : Type Library Browser & code generator
- Ax_Lite.bi include file for AxSuite functions no static lib (all the functions are in the bi file)

- source code for all the files to buid yourself axsuite3.exe
- sample code for MSCAL.OCX calendar control using : Com 2 syntaxes / vTable Call / invoke + events

Installation:

- Copy include .bi file to ..\FreeBASIC\Inc
- Copy AtlCtl/Atl71Ctl.dll to ..\FBEdit if you use it
- Copy AxSuite3.exe to ..\FBEdit if you use it
- Copy Atl.dll or Atl71.dll to your Windows system

Nota: AxSuite3.exe can recreate all the needed files, if you have destroyed some

- In FBEdit: (optional)
  Option->Dialog Editor->Custom Control->Add->load : AtlCtl.dll and Atl71Ctl.dll
  Option->Tools Menu->Insert->Menu Item : AxSuite3 , command : AxSuite3.exe

get more samples
on AxSuite2 package in http://www.freebasic.net/ old site/arch/upload/axsuite2pkg.zip
or in that backup link https://db.tt/ypQxvyYN

Ax01: Calendar control with automation dispatch call and event (mscal.ocx)
Directx8 : tutorial on directx8 with matrix programming– taken from DirectX4VB
AxExcel : MS Excel COM programming with automation dispatch call and event.

Command List - In Functional Order

Sub AxInit(ByVal Host As Integer=False)
Initialize AxSupport library
Set Host to True for ActiveX control, set to False for non window control programming
Ex: AxInit(True)

Function AxCreate_Object overload(strProgID1 AS string, strIID1 AS string = "") as any ptr
Creating an COM Object by Program ID or Clsid (class Id) + IID (interface Id)
StrProgID: Program ID or Clsid as string
strIID1: interface id as string
Ex: Objptr = AxCreateObject("Excel.Application")
or Objptr = AxCreateObject("{8E27C92B-1264-101C-8A2F-040224009C02}" , "{8E27C92C-1264-101C-8A2F-040224009C02}")

Function AxCreate_Object overload(hwnd_control as hwnd) as any ptr
Creating an COM Object by hwnd of control (atlwin class)
Ex: Objptr = AxCreateObject(hwin)

Sub AxRelease_Object(byVal Objptr as any ptr)
Release the created object

Function AxCreateControlLic (ByVal strProgID AS lpOLEStr, byval hWndControl AS hwnd, _ byval strLicKey AS lpwstr) AS Long
Creating Licensed ActiveXControl by Program ID and License key.
Return AxScode
StrProgID: Program ID
HWnd: handle of control
StrLicKey: license key
Ex: MyScode as scode=AxCreateControlLic(MyProgID, getdlgitem(hwin,idc_stc1), MyLicKey)

---

function AxCreate_Unreg(ByVal hdll As HMODULE, byval CLSIDS As string, byval IIDS As string, _ ByVal hWndControl AS hwnd = 0) as any ptr

Creating an unregistered COM Object by library adress, Clsid (class Id) + IID (interface Id) , optionally with hWndControl
To use not registered COM components , interresting to avoid register control

---

Sub setObj(byval pxface as uinteger ptr,ByVal pThis as uinteger)
Set native dispatch AxSupport style object address
Pxface: pointer to native dispatch AxSupport style interface
PThis: Object address (returned by AtlAxGetDispatch or AxCreateObject)
Ex: SetObj @MyInterface, punk

Sub setVObj(byval pxface as uinteger ptr,ByVal vThis as variant)
Set AxSupport style interface object address
Pxface: pointer to AxSupport style interface vThis:
Variant Dispatch (returned by object)
Ex: SetVObj @MyInterface, vRet

Function ToBSTR(cnv_string As String) As BSTR
Convert string to BSTR, mostly used by ActiveX/COM
Ex: MyBSTR as BSTR=ToBSTR("Test String")

Note: Free allocated BSTR after used to prevent memory leak
with defined macro : Ax_FreeStr(bstr)

complementary function from BSTR to string
Function FromBSTR(ByVal szW As BSTR) As String

Function VariantS(ByRef v As variant)As String
To get string value of Variant string (bstr)
Ex: MyString as string=Variants(vString)

Function VariantB(ByRef v As variant)As bstr
To get bstr value of variant string (bstr)
Ex: MyBSTR as bstr=Variantb(vString)

Function VariantV(ByRef v As variant)As Double
To get numeric value of variant numeric
Ex: MyInteger as integer = Variantv(vNumeric)

Function Vptr(Byval x As Any Type*) As Variant Ptr
Assign any type Variant ptr (use callocate to create the memory allocation)*
Ex:
Dim vVar As Variant
vVar=*vptr(20) : vVar=*vptr(20.5) : vVar=*vptr("Hello there")
Note:
used when passing value as variant ptr to AxSupport sub or function

---

VLet (As variant, x As Any Type*) ! It is a macro using Vptr Functions
Assign any type to variant*
Ex: Dim vVar As Variant : Vlet(vVar, 12.75) : Vlet(vVar, "Test String")

Other useful macros
#define toVariant (x) x as any type
#define Ax_FreeStr(bs) to free bstr
#define Kill_Bstr(bs) to kill bstr

---

*any type
(variant/string/byte/short/Integer/Longint/single/Double/BSTR/Ubyte/Uinteger/Ulongint/Ushort/Ubyte)

COM automation call functions

Sub Ax_Call (pThis As lpdisptach,Byref Script As String,…)
Automation dispatch for object Call *ObjCall in AxSuite2*
Sub Ax_Put (pThis As lpdisptach,Byref Script As String,…)
Automation dispatch for object Put *ObjPut in AxSuite2*
Sub Ax_Set (pThis As lpdisptach,Byref Script As String,…)
Automation dispatch for object set *ObjSet in AxSuite2*
Function Ax_Get(pThis As lpdisptach,Byref Script As String,…)As Variant Ptr
Automation dispatch for object Get function *ObjGet in AxSuite2*

pThis: object/dispatch address

Script: representation of (dot) calling method(s), @ indicate number of passing parameter each method ,… : variant ptr parameter(s) list, the sequence should follow script.

```
Ex:     Ax_Call     xlwbks,"Item@1._OpenText@1",vptr(1),vptr("c:\xlapp\test.txt")
        Ax_Put xlapp,"Visible@1",vptr(TRUE)
        Ax_Set   msdatgrid,"DataSource@1",vptr(rsdisp)
        s as string=variants(*Ax_Get(wbk,"worksheets.Item@1.name",vptr(1)))
```

Useful extended macros

Ax_GetStr(pThis As lpdisptach,Byref Script As String,…)As string
Ax_GetVal(pThis As lpdisptach,Byref Script As String,…)As double
Ax_GetBstr(pThis As lpdisptach,Byref Script As String,…)As Bstr
Ax_GetObj(pThis As lpdisptach,Byref Script As String,…)As lpdisptach

---

Dispatch functions

Sub AxCall (ByRef pmember as tmember,...)
Native dispatch for object Call and Property put/set method
pMember: Axsupport Interface style member
,… : variant ptr parameter(s) list
Ex:

```
Vlet ( vVar , bgr(255,64,127))
AxCall MyInterface.putFontColor, @vVar
```

Function AxGet (ByRef pmember as tmember,...)as variant
Native dispatch for Function and property get method
pMember : Axsupport Interface style member
,… : variant ptr parameter(s) list , return value as variant
Ex:

```
VLet (vIndex,1)
FontColor as integer = variantv(AxGet(MyInterface.GetFontColor,@vIndex))
```

---

vTable method call

Obj->lpVtbl->Method(Obj,…) RetValue=Obj->lpVtbl->Method(Obj,…)
Start from AxSuite2, vTable call and generated code will use C vTable call syntax.
Ex:

```
Scode = dx->lpvtbl->Direct3DCreate(dx,@d3d) 'get Direct3D Interface, returning scode
d3d->lpvtbl->GetAdapterDisplayMode(d3d,D3DADAPTER_DEFAULT,@DispMode)
```

simplified macro :
Ax_Vt(Obj,Method, ...) like Ax_Vt(pVTI,getcOptions ,@pVTI2)

Control Window Functions/subs

FUNCTION AxWinFull(byVal h_parent as hwnd, name1 as string, progid as string, _
x as integer, y as integer, w as integer, h as integer, _
style as integer = WS_visible or WS_OVERLAPPEDWINDOW, _exstyle as integer = 0) as hwnd

Create normal window to hold an activeX control

FUNCTION AxWinTool(byVal h_parent as hwnd, name1 as string, progid as string, _
x as integer, y as integer, w as integer, h as integer, _
style as integer = WS_visible, exstyle as integer = WS_EX_TOOLWINDOW) as hwnd

Create tool window to hold an activeX control

FUNCTION AxWinChild(byVal h_parent as hwnd, name1 as string, progid as string, _
x as integer, y as integer, w as integer, h as integer, _
style as integer = WS_visible or WS_child or WS_border, exstyle as integer = 0) as hwnd

Create child window to hold an activeX control

h_parent : hwnd of parent window ; name1 : name of the window ; progid : progID or registered control
x ; y ; w ; h for position x,y , w for width and h for height of the window

---

Sub AxWinKill(byVal h_Control as hwnd)
Destroy control window

Sub AxWinHide(byVal h_Control as hwnd, byVal h_Parent as hwnd = 0)
Hide control window and refresh parent window

Sub AxWinShow(byVal h_Control as hwnd, byVal h_Parent as hwnd = 0)
Show control window and refresh parent window

FUNCTION AxWinUnreg(byVal h_parent as hwnd, _
x as integer, y as integer, w as integer, h as integer, _
style as integer = WS_visible or WS_child or WS_border, exstyle as integer = 0) as hwnd

Create Window container for hosting non registered window control

h_parent : hwnd of parent window
x ; y ; w ; h for position x,y , w for width and h for height of the window

---

A. Using Automation Dispatch

Must have headers:

```
#include Once "windows.bi"
#Include Once "Ax_Lite.bi"        'no  static lib



AxInit(True)     'ax  global COM initialization   True if Atl control , else False


'================================================================================
'  if needed use   AxControlChild or AxControlTool to create the control on form
and use the progid to create the window
 '================================================================================
Dim Shared As any ptr       Obj_Ptr        ' object Ptr
Dim Shared As Dword         Obj_Event    ' cookie for object events

Declare Sub Call_Sett()


' The events can be connected .....
Declare Function DCalendarEvents_Events_Connect(ByVal As IConnectionPointContainer Ptr,ByRef As dword) AS Dword
Declare Function DCalendarEvents_Events_Disconnect(ByVal As IConnectionPointContainer ptr, ByVal As dword) AS Long


Sub Call_Init(ByVal hWin As hWnd)' be called from initialization of the control form
            Obj_Ptr = AxCreate_Object (hwin )                                   'get object control address

            ' The events are now connected .....
            DCalendarEvents_Events_Connect(cast( any ptr ,Obj_Ptr),Obj_Event)        'connect object with its event

            Call_Sett()                                                         'initial settings if you want some
End Sub


 Sub Call_OnClose()                                                             ' normaly be called from close form command
            ' The events are now disconnected .....
            DCalendarEvents_Events_Disconnect(cast(any ptr ,Obj_Ptr),Obj_Event)      'disconnect  event from object
             AxRelease_Object(Obj_Ptr)     'release object
            'AxStop()                                                  'only one by project, better on the WM_close of last Form
End Sub


Sub Call_Sett()                                                 'initial settings here
            'ex put/ get /call values
            ‘change Calendar Title font
            Ax_Put    cal,"TitleFont.Name@1",vptr("Arial")
            Ax_Putcal,"TitleFont.Size@1",vptr(12)
            Ax_Put cal,"TitleFont.bold@1",vptr(TRUE)
            Ax_Put cal,"TitleFont.italic@1",vptr(true)
End Sub
'================================================================================
```

B. Using Native vTable

Must have headers:

```
#include Once "windows.bi"
#Include Once "Ax_Lite.bi"              'no  static lib


#Include Once "mscal_vTable.bi"         'vTable generated file


AxInit(True)     'ax  global COM initialization   True if Atl control , else False
'================================================================================
' remind  control form :  Classname = AtlAxWin              ProgID = MSCAL.Calendar
'  if needed use   AxControlChild or AxControlTool to create the control on form
'================================================================================
Dim Shared As any ptr       Obj_Ptr       ' object Ptr
Dim Shared As Dword         Obj_Event     ' cookie for object events

Dim Shared As ICalendar_ Ptr            pVTI          ' vTable type ptr


Declare Sub Call_Sett()


' The events can be connected .....
Declare Function DCalendarEvents_Events_Connect(ByVal As IConnectionPointContainer Ptr,ByRef As dword) AS Dword
Declare Function DCalendarEvents_Events_Disconnect(ByVal As IConnectionPointContainer ptr, ByVal As dword) AS Long
Sub Call_Init(ByVal hWin As hWnd)' be called from initialization of the control form
        Obj_Ptr = AxCreate_Object (hwin )                                   'get object control address


         pVTI = Obj_Ptr                                 'assign object to vTable  type
         ' or  pVTI= Create_ICalendar()                 ' to assign object direcly to interface vTable

        ' The events are now connected .....
        DCalendarEvents_Events_Connect(cast( any ptr ,Obj_Ptr),Obj_Event)        'connect object with its event
        Call_Sett()                                                             'initial settings if you want some
End Sub
Sub Call_OnClose()                                                              ' normaly be called from close form command
        ' The events are now disconnected .....
        DCalendarEvents_Events_Disconnect(cast(any ptr ,Obj_Ptr),Obj_Event)      'disconnect  event from object
         AxRelease_Object(Obj_Ptr)     'release object
        'AxStop()                                                 'only one by project, better on the WM_close of last Form
End Sub
Sub Call_Sett()                                                  'initial settings here
         'ex put/ get /call values
         ‘put date
         pVTI->lpvtbl->putDay( pVTI , 14 )          ' or Ax_Vt (pVTI,putDay,14)
         pVTI->lpvtbl->putMonth( pVTI , 12 )        '  or Ax_Vt (pVTI,Month(,12)
         pVTI->lpvtbl->putYear( pVTI , 2014 )       '  or Ax_Vt (pVTI,putYear(,2014)

End Sub
'================================================================================
```

C. Using Native Dispatch

Must have headers:

```
#include Once "windows.bi"
#Include Once "Ax_Lite.bi"                'no  static lib

#Include Once "mscal_Invoke.bi"          'AxSuite generated invoke call



AxInit(True)     'ax  global COM initialization   True if Atl control , else False
'================================================================================
' remind  control form :  Classname = AtlAxWin                ProgID = MSCAL.Calendar
'  if needed use   AxControlChild or AxControlTool to create the control on form
'================================================================================
Dim Shared As any ptr       Obj_Ptr        ' object Ptr
Dim Shared As Dword         Obj_Event      ' cookie for object events

Dim Shared As ICalendar                    Obj_Disp      'calendar dispatch object

Declare Sub Call_Sett()


' The events can be connected .....
Declare Function DCalendarEvents_Events_Connect(ByVal As IConnectionPointContainer Ptr,ByRef As dword) AS Dword
Declare Function DCalendarEvents_Events_Disconnect(ByVal As IConnectionPointContainer ptr, ByVal As dword) AS Long
Sub Call_Init(ByVal hWin As hWnd)' be called from initialization of the control form
        Obj_Ptr = AxCreate_Object (hwin )                                   'get object control address

        SetObj ( @Obj_Disp , Obj_Ptr )                                      'set Cal object to Calendar address


        ' The events are now connected .....
        DCalendarEvents_Events_Connect(cast( any ptr ,Obj_Ptr),Obj_Event)    'connect object with its event
        Call_Sett()                                                         'initial settings if you want some
End Sub
Sub Call_OnClose()                                                          ' normaly be called from close form command
        ' The events are now disconnected .....
        DCalendarEvents_Events_Disconnect(cast(any ptr ,Obj_Ptr),Obj_Event)  'disconnect  event from object
         AxRelease_Object(Obj_Ptr)     'release object
        'AxStop()                                          'only one by project, better on the WM_close of last Form
End Sub
Sub Call_Sett()                                            'initial settings here
        'ex put/ get /call values
        'put date
         axcall Obj_Disp.putDay , vptr(06)
         axcall Obj_Disp.putMonth , vptr(05)
         axcall Obj_Disp.putYear , vptr(1985)
End Sub
'================================================================================
```