Module8: ActiveX Controls

In 1996, Microsoft renamed the OLE 2.0 technology to ActiveX. An ActiveX Control is like an OLE Server but it is stored inside an OCX file rather than a DLL file and it may be instantiated as a GUI object inside another GUI application.

§ 8.1 Creating an ActiveX Control in an OCX

§§ 8.1.1 The *ActiveXControl* Object

Since an ActiveX control is intended to be the child of another GUI object, the Dyalog APL *ActiveXControl* object, unlike the *OLEServer* object, cannot be created as a child of the Root. It must be created as the child of a *Form*. This *Form* object is the notional container object for the ActiveX. When the *ActiveXControl* is created within another application, the *Form* will be replaced by the new GUI parent from within the application. (Some details about the container object that is involved in any particular environment may be found from the *ActiveXContainer* object which is returned by the *Container* property of the *ActiveXControl*.)

8.1.1.1 Create a *Form* with an *ActiveXControl* object child.

```
'F'⎕WC'Form'('BCol' 120 20 230)
⎕CS'AXC'F.⎕WC'ActiveXControl'
```

The *ClassID* property contains a unique identifier for this control. *Container.BCol* should return the *Form BCol*.

§§ 8.1.2 The *Create* Callback

An *ActiveXControl* is created whenever it is used. A function is usually required which will initialise a control. This function may, for example, create the children of the ActiveX as soon as it exists. Two events are supplied for initialisation code, *Create* and *PreCreate*.

8.1.2.1 Assign the *onCreate* property of *F.AXC* to function ∇*cre8*∇, where

```
      ∇ cre8
[1]     'C'⎕WC'Calendar'
      ∇
```

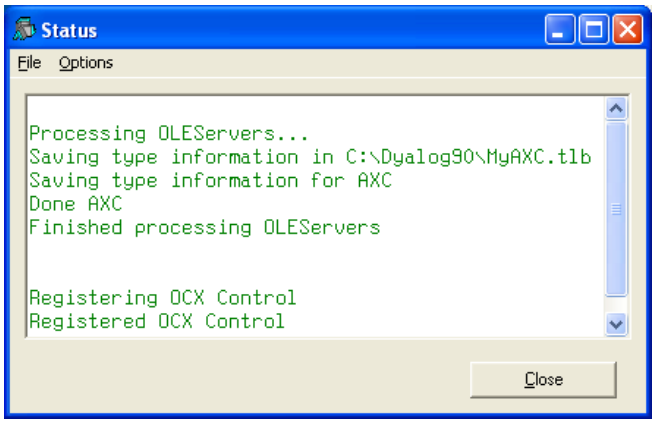
This will create a *Calendar* child of the *ActiveXControl*. Or you could experiment with a more complicated callback such as:

```
      ∇ cre8;BB
[1]     :With 'BB'⎕WC'BrowseBox'
[2]         StartIn←'C:\'
[3]         onFileBoxOK onFileBoxCancel←1
[4]         Caption←'Resource Brower'
[5]         HasEdit←1
[6]         Msg←⎕DQ''
[7]         :If 'FileBoxOK'≡2⊃Msg ⋄ Dir←Target
[8]         :Else ⋄ Dir←'' ⋄ :EndIf
[9]     :EndWith
      ∇
```

§§ 8.1.3 Creating the OCX on `)SAVE`

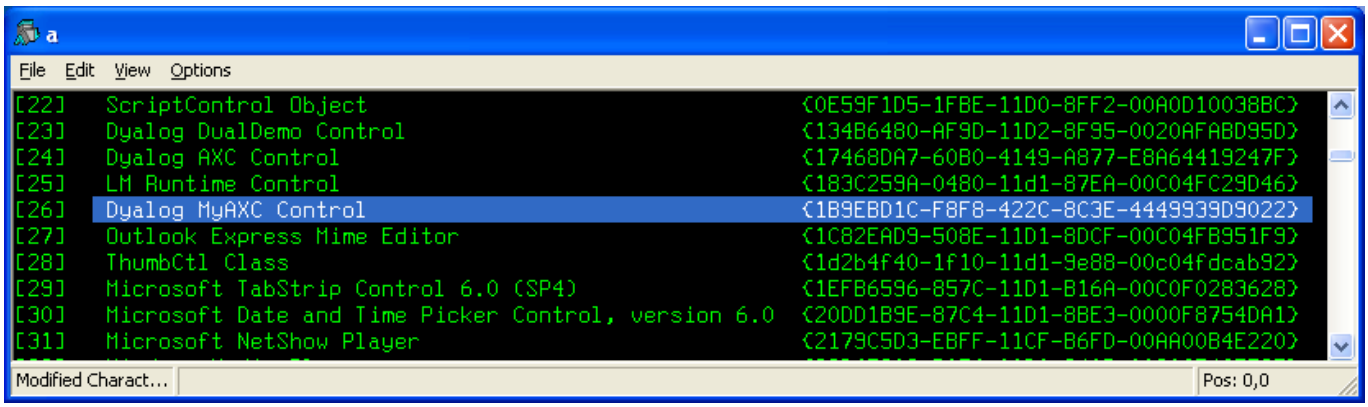
With the above ActiveX we have not introduced any functions to be exported as methods (or properties), nor any variables to be exported as properties. If we had functions or variable to be exported for use by the external application then we could set their calling structure using `ActiveXControl` methods `SetFnInfo` or `SetVarInfo` under program control. An external event may also be declared using `SetEventInfo`. Alternatively, the entire control might be constructed manually and the calling information set in the property sheets obtained by right-clicking on the name and selecting [Properties][COM Properties].

8.1.3.1 Check that [File][Make OCX/DLL on)SAVE] is checked and `)SAVE` the workspace as MyAXC.DWS. This generates type information in a TLB file, just as is done for an `OLEServer`, and then the created OCX Control is registered in the Windows registry in a similar way to OLE Servers.



There now exists a MyAXC.OCX file as well as the MyAXC.DWS file, and there is a new entry in the list of OLEControls:

```
      a←⍒↑#.OLEControls
```



This list of controls contains the unique identifier found in the `ClassID` property of the `ActiveXControl`. This ClassID is also to be found in the Registry.

8.1.3.2 Run **REGEDIT.EXE** and find details of Dyalog.MyAXC.MyAXCCtrl.1 under key
\HKEY_CLASSES_ROOT\Dyalog.My AXC.MyAXCCtrl.1\CLSID
In particular, note the file MyAXC.OCX mentioned under key InProcServer32.

8.1.3.3 Write an ActiveX control which reads and writes a character matrix in component 2 of an APL component file (component 1 should always be a description). Create the file if it doesn't exist. Display the component for editing in a multi-line `Edit` object.

An OCX file represents a set of ActiveX controls which are ready to run when supplied with the particular Dyalog APL Dynamic Link Library DYALOG.DLL in use when the OCX file was saved.

§ 8.2 Using your ActiveX Control

§§ 8.2 1 Creating an Instance from an *OCXClass*

The way that you use an ActiveX control in Dyalog APL is by way of an *OCXClass* object. An *OCXClass* object is created from a registered control by setting the *ClassName* of the *OCXClass* at create time to the name of the control as given in *#.OLEControls*. The name given to the *OCXClass* object is then the name of a new *Type* of object, instances of which may be created in the usual way as children of appropriate parents.

8.2.1.1 In a CLEAR WS create a new class (*Type*) of object and an instance of this new class on a *Form*.

```
'MyAXC'⎕WC'OCXClass' 'Dyalog AXC Control'
'Frm'⎕WC'Form'
'Frm.Inst'⎕WC'MyAXC'
```

Change some of the properties of this instance.

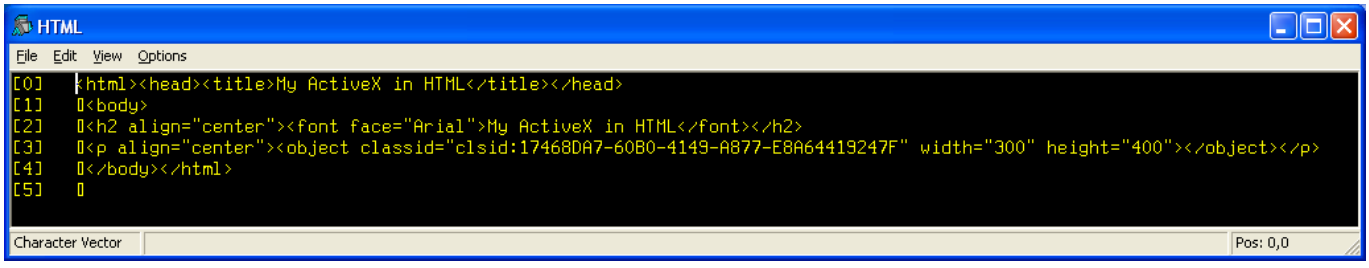
This control may be called from any OLE-aware application via a suitable language such as VBA or JavaScript.

8.2.1.2 Load the supplied DUALBASE workspace and follow the instructions in chapter 14 of the marvellous Dyalog APL *Interface Guide* (which may be downloaded from www.dyalog.com [Download zone]). Further instruction on the Dyalog Dual Control can be found in the APL98 course notes in APL98OCX.ZIP which may also be downloaded from [Product][OLE Controls (OCX's)][ActiveXControl].

§§ 8.2 2 Using Controls in IE 6.0

ActiveX controls may be incorporated in HTML pages by reference to the classid of the control.

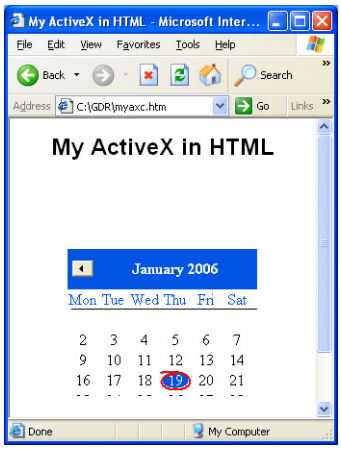
8.2.2.1 Create a native file called MyAXC.HTM containing the HTML string in the window below, using the *ClassID* of your object. The squish-quad represents carriage return/linefeed, *ie* *⎕AV[3 2+⎕IO]*.



The source and display look something like:

```
<html>
        <head>
                <title>My Active X Control</title>
        </head>
        <body>
                <h2 align="center">
                <font face="Arial">A X C</font>
                </h2>
                <p align="center">
                <object
            classid="17468DA7-60B0-4149-A877-E8A64419247F"
```

```
            width="300" height="400">
               </object>
               </p>
        </body>
</html>
```



More information on how to do this is to be found in http://support.microsoft.com/kb/q159923/ where licensing of controls is also discussed.

§§ 8.2 3 Using Controls in Visual Basic and VBA

In APL98OCX.RTF and in chapter 14 of the *Interface Guide* you will find explanations of how to call the Dual Control example from VB, VBScript and IE.

§ 8.3 Browsing registered OLE Controls

§§ 8.3.1 Having a quick Look

A snippet from a function by Dick Bowman shows how you might go about investigating the controls in your, possibly very large, list of `#.OLEControls`.

```
:For Item :In  ⊃⍕'.'⎕WG'OLEControls'
     Item
     'Inst'⎕WC'OCXClass'Item
     'F'⎕WC'Form'
     'F.X'⎕WC'Inst'
     :Trap 11
         2 ⎕NQ'F.X' 'ShowHelp' 'Type'
     :End
 :End
```

§§ 8.3.2 Having a deeper Look

For a deeper look into the registered controls on your list, the supplied workspace OCXBROWS.DWS may be used to investigate available controls.

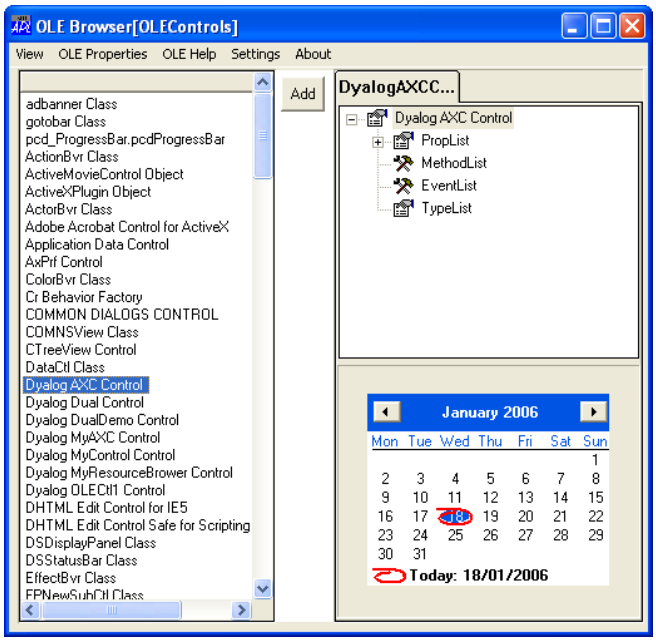
8.3.2.1 Now XLOAD the workspace OCXBROWS.DWS and trace the `⎕LX`, ▿CLASSES.LISTCLASSES▿. Skip over line [17] and trace into line [18]. Select 'Dyalog MyAXC Control' and hit [Details]. Trace into and examine line [41].

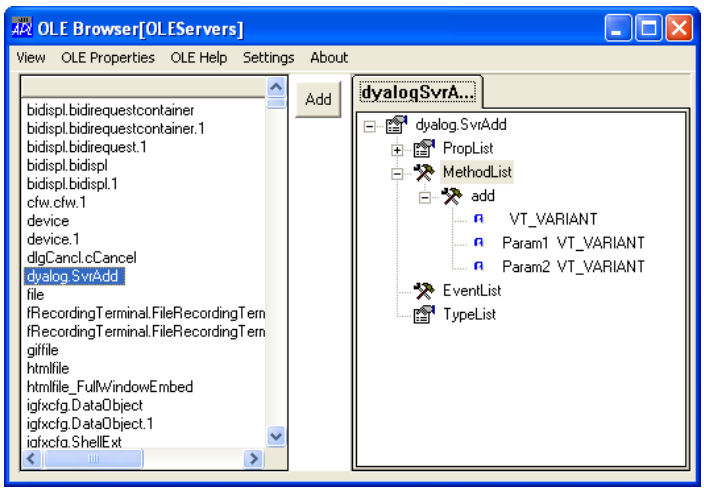
Hint: Normalise `Form` to continue beyond line SHOWCLASS[15].

Alexander Balako has kindly given his workspace OLEBROWS.DWS as a free download from www.dyalog.com [Download][OLEBROWS]. This workspace enables exploration of both the ActiveX controls in you list `#.OLEControls`, and the OLE servers in your list `#.OLEServers`.

8.3.2.2 In OLEBROWS.DWS, attempt to trace the `⎕LX`, `∇OLEBrowser.MAIN∇`, and note the use of the `Create Event` on line [1].

Hint: Put a `⎕STOP` on line [4] and run the function past line [3].





§§ 8.3.3 Trying some Examples

The Dyalog APL `ActiveXControl` object allows you to package an application as an ActiveX control. The application then comprises at least two files; your OCX file and the Dyalog APL dynamic link library file, DYALOG.DLL, **with which your OCX was created**. Both files have to be visible when the OCX is **registered**. Also the full path name of the OCX is recorded in the registry so it cannot be moved around easily. It is conventional to place OCX and DLL files in the ..\Windows\System32\ directory.

Let us call some controls from an APL workspace. If you have the VideoSoft FlexArray control on your list then you can start it with:

```
'FAC'⎕WC'OCXClass' ':-) VideoSoft FlexArray Control'
'Frm'⎕WC'Form'
'Frm.Flex'⎕WC'FAC'
```

8.3.3.1 You probably have Windows Media Player on your list. Create an instance of this control on a `Form` and examine the instantiated object from the inside. Research the `FileName` and `QueueEvents` properties of the class. In Dyalog APL version 10, the `Grid` object can have controls in cells. Run the following lines of code and investigate the possibility of playing 25 tunes at once.

```
'WMP'⎕WC'OCXClass' 'Windows Media Player'
'F'⎕WC'Form'('Coord' 'Pixel')
H W←(F.Size-4)÷5
'F.G'⎕WC'Grid'(5 5⍴'')(0 0)
F.G.CellHeights←H ⋄ F.G.CellWidths←W
F.G.Size←F.Size
F.G.ShowInput←1
F.G.TitleHeight←0 ⋄ F.G.TitleWidth←0
'F.G.WMP'⎕WC'WMP'
```

```
F.G.Input←'F.G.WMP'
```

The Google search bar, supplied free by Norbert Jurkiewicz, as mentioned in Module4, can be summarised, in essence, in the following function which takes any Google search string as its Rarg. The function creates the Microsoft Web Browser *MSWB* class object and an instance of the class on a suitably sized *Form*. The *DocumentComplete* event is set to 1 in order to terminate the ensuing ⎕*DQ* when the entire document has been obtained. Then come the crucial lines. We invoke the objects *Navigate* method with a carefully crafted argument:

```
http://www.google.com/search?ie=UTF-8&oe=UTF-8&sourceid=deskbar&q=',XXX
```

The *XXX* is a character string containing a request as one would type it into Google. The rest is the URL-specific command line suitable for a general Google search.

```
     ∇ searchFor XXX;To
[1]    'MSWB'⎕WC'OCXClass' 'Microsoft Web Browser'('QueueEvents' 0)
[2]    'F'⎕WC'Form'('Coord' 'Pixel')('OnTop' 1)('Border' 0)
[3]    F.Size←400 600
[4]    F.EdgeStyle←'Recess'
[5]    'F.IE'⎕WC'MSWB'
[6]    F.IE.Posn←0 0
[7]    F.IE.Coord←'Prop'          ⍝ MUST USE PROP to look ok
[8]    F.IE.Size←100 100
[9]    F.IE.onDocumentComplete←1  ⍝ exit ⎕DQ when finished
[10]   To←'http://www.google.com/search?'
[11]   To,←'ie=UTF-8&oe=UTF-8&sourceid=deskbar&q=',XXX
[12]   F.IE.Navigate To
[13]   ⎕DQ'F'
[14]   F.IE.Refresh               ⍝ MUST REFRESH to look ok
     ∇
```

8.3.3.2 Explore the possibility of viewing 25 Internet sites simultaneously in a 5 by 5 *Grid*.

8.3.3.3 Please ask for the next module on ⎕*NA* ☺.