Legacy []WI GUI in VisualAPL

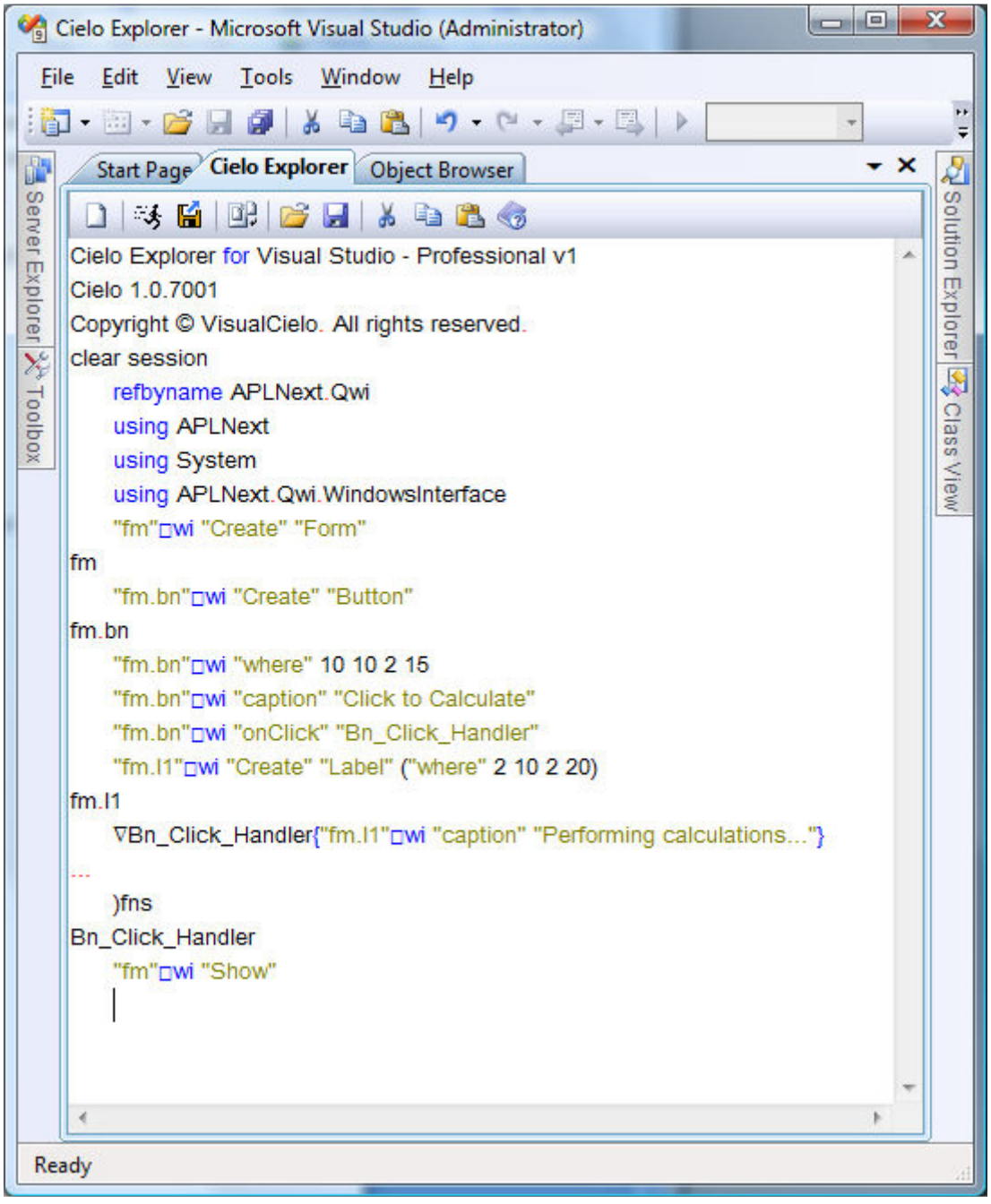
The legacy []wi feature was used to support APL+Win access to Win32 Forms and Controls for GUI (graphical user interface) development.

For compatibility with legacy APL application systems, VisualAPL has implemented the []wi feature as a .Net assembly, "APLNext.Qwi.dll" that is installed with VisualAPL and is based upon the System.Windows.Forms namespace.

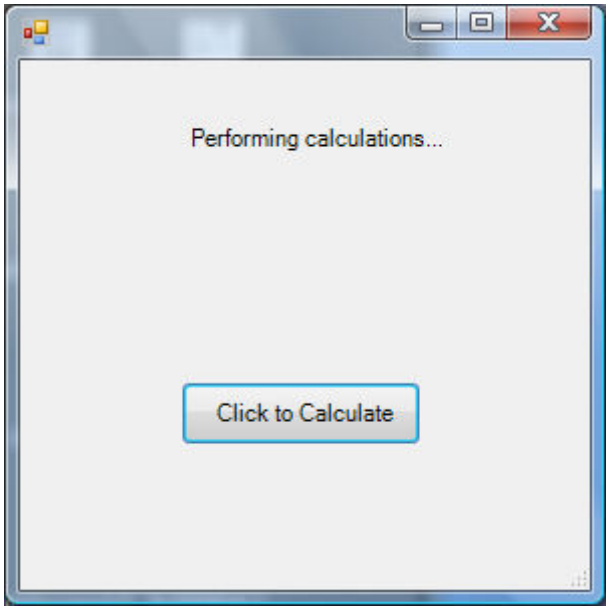
To use the []wi feature in a VisualAPL project or a Cielo Explorer session or script, a reference to this .Net assembly must be made. For example:

```
refbyname APLNext.Qwi
using APLNext
using System
using APLNext.Qwi.WindowsInterface
```

Once this reference is made, the []wi feature may be used to implement a GUI for the application system, for example:

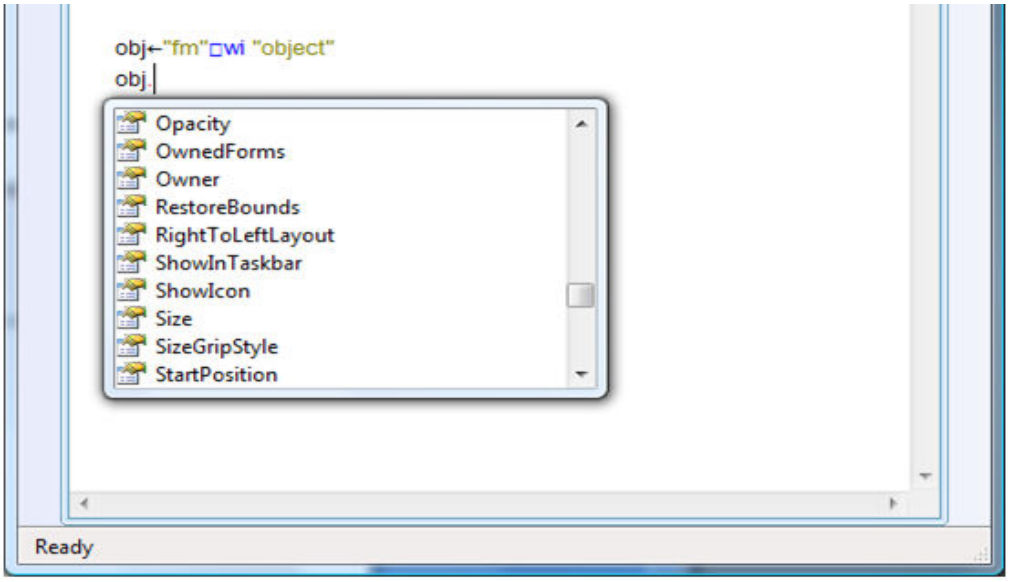


The form is presented and when the button is clicked the form appears as:



Summary of []wi Feature Support in VisualAPL

In VisualAPL, the underlying objects covered by the []wi-feature are fully accessible, so that the legacy GUI controls now include many new properties, methods and events in the VisualAPL environment. Use the ‘object’ property of the GUI control to access all its members.



In addition, many new GUI controls are available because all GUI controls supported by the System.Windows.Forms .Net namespace can be accessed by the []wi feature.

Legacy []wi Features Summary

For proper understanding the []wi feature, refer to the APL+Win documentation.

Bracketed items, "[…]", indicate those legacy features which are not available in the VisualAPL .Net environment, however in most cases there is a .Net method, property or event which provide functionality similar to the legacy feature which is not available.

In case a control has a property or method with the same name as the legacy []wi member, the legacy []wi member is accessed through []wi and the same-named, .Net property or method is accessed directly using the 'object' property of the GUI control.

[]-operators associated with the []wi feature:

Same as in APL+Win:

[]wself
[]warg
[]wres
[]wevent

Enhancements to []wi in VisualAPL"

[]wsender	The object control that can be used to access the members on the control which raised the event
[]wievent	The event object that can be used to access all of the EventArgs members.

General []wi features:

Same as in APL+Win:

[]wi supports arbitrary APL scripts for events, e.g. "fm.b" []wi "onClick" "a=1+1"

[]wi supports the events: onNew, onAction, onEvent, onDefer

Controls and Members:

Button
caption
style 0 1 2 4 8 [16 32 64] 128 256 512 768 4096 8192 12288
[imagespace]
imagelist
imageindex
value

Check - CheckBox
caption
order
style 0 1 2
value

Combo
autocomplete [32] 16 8 4 2 1
style [4096 n/a in .Net] , 2048, 1024, 512, [256], [128 16 8 automatic in .Net], [64, 32], 4, 2, 1
list
imagelist
limit
name
value
[text]

CommandBar
bitmap
caption
captiondock
captionfloat
dock
dockable
[dockbreak]
dockheader
dockideal
docklength
dockmargin (read only property)

dockmax
dockmin
dockmindepth
dockside
dockvert (read only property)
floatlayout
floatsize
floatwhere
imagelist
imagelistdiabled
imagelisthot
normsize // read only
order
showtext
siblings (read only property)
style
0 1 [2] 4 8 [16] [32] [64] [128] [256] 512

CommandButton
caption
image
size (read only property)
style
0 1 2 [4] 8 16 [32] 64
value
width
[wrap]

DateTime
style
0 1 2 3 4 8 16 32 64 128 256
value

limit
today
range
firstday
monthdelta
color
dropfont
text
format
tooltip
[minsize]
[text]
[today]

Edit
text
range
selection
style
    0 1 2 4 8 16 [32] 64 128 256 512 1024 2048 4096 8192 16384
border
seltext
LineToChar

Form
border
caption
visible
value

Frame
style
    0 [1] 2 [3] [4] 5 [6] [7]
caption

Imagelist
style
    can not be set, is always 0
imagesize
imagealloc [obsolete]
imagenames
maskcolor
overlays
colordepth
AddImages
imagecount
himage

Label
caption
edge
style
    0 1 2 4 8 32 64

List
list
style
    0 1 2 16 32 64 128 [256]
value

Listview
viewmode
    largeimage smallimage list report
viewalign
    top left none
imagelistlarge
imagelist
imagelistuser
highlightfocus
list
style
    1 [2] 4 8 [16] 32 [64] 128 [256] 512 1024 2048 4096 8192 [16384] [32768] 65536
columndisplay
sortorder
value
AddRows
InsertRows
DeleteRows
SetRows
SetImages
[SetChecks]
SetCells
GetRows
GetCells
EnsureVisible
[Arrange]
[AutoFit]
count
roworigin
searchstring
[sourceformats]
[targetformats]
[dragimage]
[SetLinks]

MDIForm
Arrange

Menu
caption
value
style
    0 1 2 3 8
separator
shortcut
enabled
visible
order

imagelist
imageindex
opened

Page
style
order
extent
visible
[imageindex]
Close
border

Option
caption
value
style
    0 1
order
To group the Options, use the Panel, for example:
"fm.panel" []wi "Create" "Panel"
"fm.panel.op1" []wi "Create" "Option"
"fm.panel.op2" []wi "Create" "Option"
etc.

Picture
style
    0 2 4 16 64
bitmap
[origin]
[imagesize]
[image]
[bitmapsize]
[hdc]

ProgressBar
style
    0 1 2 3
value
[Stepit]

RichEdit
style
    0 4 8 16 32 64 1024 [2048] 4096 8192 16384 32768 65536 131072
range
rtf
selcolor
text
selection
seltext
selalign
selbullet
selfont
selindents
selrtf

selstyle
zoom
[msversion]
[canpaste]
[canredo]
[canunder]
[selnumstart]
[selnumstyle]
[selnumtab]
[selpargall]
[seltabs]
[undolimit]
[undoname]
[redoname]
border
font
CharToLine
LineToChar
[Undo]
[Redo]
[ScrollCaret]

Selector
style
    0 1 2 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768 65536
fixedtabsize
padding
imagelist
color
border
value
tabrows
pages

Spinner
range
value
border
style
    0 1 4 8 16 32 64
[wrap]
[buddy] Use NumericUpDown .Net class for equivalent,
in general, the NumericUpDown has so much more functionality,
it should replace any Spinners in code

"fm.nd" []wi "Create" "NumericUpDown"
Many, many properties, methods, options.

Status
imagelist
status
    Column 4:
        0 1 2 8 16 32 64
color
HitTest

SetStatus
PaneWhere
where
extent
size
[status]

Toolbox
list
style
    0 1 16 [32]
value

TrackBar
style
    0 1 2 4 8 [16] [32]
range
value
tickinterval
increment
color
[ticks]
[selection]
[sliderlen]
[tickpos]
value
[channelwhere]
[sliderwhere]

Tree
imagelist
imagelistuser
[dragimage]
[labeledithwnd]
list
indent
style
    0 1 2 [4] 8 16 32
InsertNodes
DeleteNodes
FindNode
ShowNode
Expand
SortChildren
EnsureVisible
[GetInfo]
[SetInfo]
count
border
[searchstring] // always returns ""

User Defined Classes
onAction
Event
Defer

"#"
newclasses
onNew
Defer
Event

General
size
where
caption
text
limitwhere
scale
tooltip
name
visible
enabled
font
pointer
[-1] 0 1 2 3 [4] 5 6 7 8 9 10 11 12 13 14 15
color
children
properties
methods
events
order
opened
self
<delta>udp
data
value
[keys]
[instance]
[links]
[mode]
[modified]
[modifystop]
[noredraw]
[scrollaccel]
[scrollmargin]
[state]
[suppress]
[tabgroup]
[tabparent]
[tabstop]
[targetformats]
translate
1 5
[Hide]

Differences from the legacy []warg for events:

Selector:
onChange
    []warg[1] = []warg[0]

Treeview:
onClick:
    []warg[1] = "label"
    []warg[2] = 0

onExpanding:
onExpanded:
    []warg[1] = 1

onCollapsing: [new event]
onCollapsed: [new event]
    []warg[1] = 0

General events:
onMouseXXX:
    []warg[3] = []warg[2]

onKeyDown:
    []warg[1] = 1
    []warg[4] = 0
    []warg[5] = 0 // to be supported in .Net 3.0

    Unsupported virtual keys:
     3

onExit event:
    []warg = "#"

onClose event:
The right argument to the 'Close' method is not assigned into []wres in the 'onClose' event.

Status onClick:
    []warg[1] = "pane"

Event Notes:

[]wres for onKeyPress, onKeyDown, and onKeyUp accepts:
[-2] -1 0 [>0 is not supported]

Setting []wres in onExit cancels the focus change, but does not allow the redirection of focus to another control.

[DDE] DDE is essentially obsolete since ActiveX was implemented by Microsoft.

Using System.Windows.Forms Directly in VisualAPL

The legacy []wi feature has limited support for newer Windows GUI controls, for example “ToolStrips”. A better solution would be to use these newer controls directly.

Since each GUI control is an independent object, it can be used and re-used in various contexts, such as on different forms, throughout the application.

In the following example do not click on the form’s [X] button while experimenting with this example, otherwise it will be necessary to re-run the code each time.

It is necessary t o reference the applicable .Net assemblies so that they are accessible in the Cielo Explorer session, script or VisualAPL project:

```
refbyname System.Drawing

using System.Drawing
```

Type the following in the Cielo Explorer session:

```
a = Form()

ms = MenuStrip()

file = ToolStripMenuItem()

open = ToolStripMenuItem()

exit = ToolStripMenuItem()

help = ToolStripMenuItem()

about = ToolStripMenuItem()

ms.Items.AddRange(file help)

refbyname System.Drawing

using System.Drawing

ms.Location = Point(0,0)

ms.Name= "menuStrip1"

ms.Size= Size(292,24)

ms.TabIndex = 0
```

```
file.DropDownItems.AddRange(open exit)

a.Show()

 a.Controls.Add(ms)

file.Name= "File"

file.Size = Size(35,20)

file.Text = "File"

open.Name= "open"

open.Size= Size(152,22)

open.Text= "Open"

exit.Name= "exit"

exit.Size= Size(152,22)

exit.Text = "Exit"

file.DropDownItems.Remove(open)

file.DropDownItems.Insert(0, open)
```

//VisualAPL arrays may be used to create an modify these controls:

```
a = Form()

ms = MenuStrip()

tsm = ToolStripMenuItem() ToolStripMenuItem()
ToolStripMenuItem() ToolStripMenuItem() ToolStripMenuItem()

file = 0

open = 1

exit = 2

help = 3

about = 4
```

```
ms.Items.AddRange(tsm[file] tsm[open])

refbyname System.Drawing

using System.Drawing

ms.Location = Point(0,0)

ms.Name= "menuStrip1"

ms.Size= Size(292,24)

ms.TabIndex = 0

tsm[file].DropDownItems.AddRange(tsm[open] tsm[exit])

a.Show()


a.Controls.Add(ms)


tsm[file].Name= "File"

tsm[file].Size = Size(35,20)

tsm[file].Text = "File"

tsm[open].Name= "open"

tsm[open].Size= Size(152,22)

tsm[open].Text= "Open"
```

Windows Presentation Foundation (WPF) Recommended:

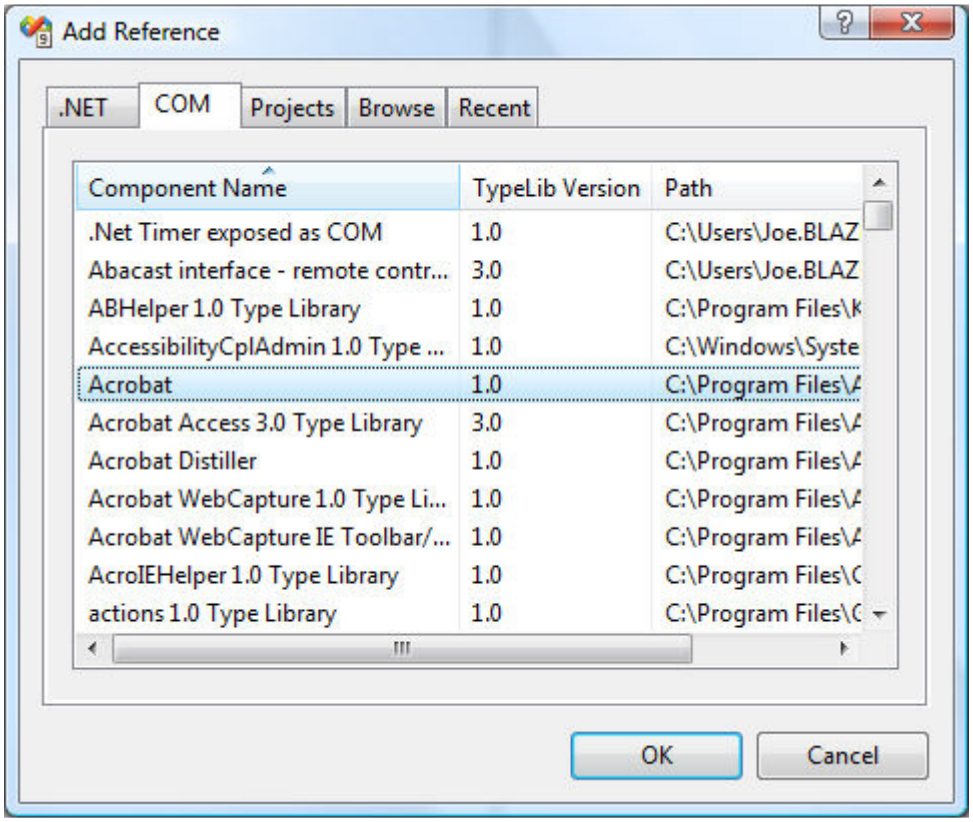
Microsoft expends tremendous resources to upgrade the features of .Net. As a result, better options for GUI construction have become available to the application system programmer.

Win32 forms and controls, as well as the System.Windows.Forms .Net namespace, have been deprecated by Microsoft because of the availability of **Windows Presentation Foundation** (WPF and XAML-format GUI specification) in .Net 3.5. When developing an application system's GUI, WPF should be seriously considered. It provides superior GUI presentation and graphics options for end users and provides the option to separately develop the GUI from the application system business rules. Microsoft has developed new GUI development tools which use WPF.

ActiveX Support in Visual Studio:

The legacy []wi feasture in APL+Win was also used to access ActiveX (COM) components. In VisualAPL this can be done directly, without the need for the overhead of []wi.

To support the transition from Win32 to .Net, Microsoft implemented robust support for ActiveX so that a reference to an ActiveX (COM) .dll can be made in any .Net language project. ActiveX GUI controls can be added to the Windows Forms toolbar too.



It is still possible to use the VisualAPL implementation of []wi to access ActiveX controls and ActiveX objects. Since VisualAPL is object oriented, the legacy re-directions syntax, using “>” is no longer necessary. Instead the dot syntax, using “name.member” is preferred. In addition the object itself can be returned and used rather than the legacy integer pointer to the object.