APL+Win v10.2 Update

Dec. 2, 2010

This update is recommended for all current APL+Win subscribers that have version 10.1 or 10.0 installed. Current APL+Win subscribers can download the update for free at http://www.apl2000.com/software.php.

Update description:

This update release includes the following important new enhancements and fixes in version 10.2:

Enhancement: New onComAction event on the system object

onComAction is a new event on the system object that is useful in debugging and/or error tracing purposes in the APL+Win ActiveX Server. The onComAction event ONLY fires in the APL+Win ActiveX Server session.

The ⎕WARG value is a nested vector of the right argument of the ⎕WI command that was executed in the client session.

Example:

The following was executed in the CLIENT session:

```
      'a' ⎕wi 'Create' 'APLW.WSEngine'
a
      'a' ⎕wi 'visible' 1
```

Then the following was executed in the SERVER session:

```
     '#' ⎕wi 'onComAction' '⎕wself ⎕wevent ⎕warg'
```

Then the following were executed in the CLIENT session (note that errors occured in the client session when invoking the XCall method because the Foo function was not defined in the server session; however the event still fired in the server session):

```
      'a' ⎕wi 'XExec' '⍳10'
1 2 3 4 5 6 7 8 9 10
      'a' ⎕wi 'XCall' 'Foo' 'LEFT' 'RIGHT'
⎕WI ERROR: exception 80004005
      'a' ⎕wi 'XCall' 'Foo' 'LEFT' 'RIGHT'
            ^
      'a' ⎕wi 'XVariable' 'x' 999
      'a' ⎕wi 'XVariable' 'x'
999
```

```
      'a' ⎕wi 'XSysVariable' 'io' 0
      'a' ⎕wi 'XSysVariable' 'io'
0
```

This resulted in the following events in the SERVER session:

```
      # ComAction  XExec #iota 10
      # ComAction  XCall Foo LEFT RIGHT
      # ComAction  XVariable x 999
      # ComAction  XVariable x
      # ComAction  XSysVariable io 0
      # ComAction  XSysVariable io
```

Enhancement: Setting of Stop/Trace in the function edit session takes effect immediately

In this version, toggling the Stop/Trace settings in a function edit session takes effect in the workspace immediately without needing to save the function so long as the function has not been changed (other than setting stops/traces) then the Stop/Trace settings themselves. The benefit of this is that a timestamp for a function is preserved when no meaningful edits have been made to the function.

Enhancement: Open character variables in the editor the same way as a function with a line number in square brackets

This enhancement opens character editor sessions (vector and matrix) the same way as functions so that if there is a line number in square brackets following the character variable, the editor sessions opens with the cursor on that line number, rather than to the topmost line.

Enhancement: A new notation has been introduced that allows passing ActiveX objects as arguments to other ActiveX methods in a more convenient and efficient manner. In the past, passing an ActiveX object as an argument to another ActiveX method required using a second, separate invocation of (⎕WI 'obj') on each argument object. For example, the last line of code below passes the 'xsl' object as an argument to the 'transformNode' method:

```
      'xml' ⎕wi 'Create' 'Msxml2.DOMDocument'
      'xml' ⎕wi 'Xload' 'C:\mydoc.xml'
      'xsl' ⎕wi 'Create' 'Msxml2.DOMDocument'
      'xsl' ⎕wi 'Xload' 'C:\stylesheet.xslt'
      'xml' ⎕WI 'XtransformNode' ('xsl' ⎕WI 'obj')
```

It is now possible to pass the 'xsl' object directly as an argument using the following alternative notation:

```
      'xml' ⎕WI 'XtransformNode(<xsl)'
```

This notation can also be used with named parameters such as this:

```
      'xml' ⎕WI 'XtransformNode(stylesheet:<xsl)'
```

The < notation tells ⎕WI that the name following it is another ActiveX object that should be directly passed as an argument.

The < notation also has the side effect of telling ⎕WI that the argument should be "passed by value" rather than "passed by reference" (⎕WI normallypasses object arguments by reference including the VT_BYREF internal flag for the argument). This is important for working around implementation limitations of some ActiveX objects that only accept object argument "by value" and fail when passed their arguments "by reference". Most ActiveX objects understand "by reference" and "by value" arguments equally well, but a few objects that use non-standard implementations do not. This enhancement provides a work around for the limitations of such objects.

The < notation can be used without an object name immediately following it to denote only the "pass by value" attribute while using the traditional notation for accessing the object's pointer. In other words, you can also code the previous two lines like this using traditional ⎕WI 'obj' notation:

```
      'xml' ⎕WI 'XtransformNode(<)' ('xsl' ⎕WI 'obj')
```

or:

```
      'xml' ⎕WI 'XtransformNode(stylesheet:<)' ('xsl' ⎕WI 'obj')
```

In this case, the only affect of < notation is to denote "pass by value" argument semantics. However, in cases where the object being passed as an argument is already bound to a named ActiveX object, it is more convenient and efficient to use the object name following the < prefix. The traditional capability of passing a object argument is retained for cases where "pass by value" semantics are required but an unnamed dynamic result of some other method is being passed as the argument.

Another side effect of using < notation is that it suppresses returning a copy of the argument's pointer in the result of ⎕WI. The examples above that use < notation only return the transformed XML as their result whereas the first example that does not use < notation returns both the transformed XML and a reference to the object's pointer in its result.

New Feature: ⎕DTR System Function

The new ⎕DTR system function performs a deep transpose of an array. It works like monadic transpose but recurses to each child element and transposes it as well.

Syntax: result← ⎕DTR array

Argument: array is any APL array

Result: an array that is transposed

Example:

```
      2 3⍴⍳6
1 2 3
4 5 6
      ⎕DTR 2 3⍴⍳6
1 4
2 5
3 6
```

Bug Fix: New [Session]VistaStyleFileDialogs APLW.INI configuration setting

The new [Session]VistaStyleFileDialogs=1 APLW.INI setting enables Windows Vista style file open and save dialogs when running on Vista and newer operating systems. These dialogs are visible when loading, copying and saving workspaces from the options in the File menu. This setting is enabled by default.

If you wish to use the older Windows XP open and save common dialogs, you must set [Session]VistaStyleFileDialogs=0 in the APLW.INI file.

Bug Fix: A Button that was a child of a Frame to be disabled eventhough the enabled property for the Button was 1 (enabled).

Bug Fix: Contrary to the manual, adding "[Config]RestartCmd=none" to the APLW.INI didn't suppress the crash log report generated when APL+Win crashed.

Bug Fix: APL+Win may crash when resuming in Code Walker after suspension on a :CASE statement. The message "DESTINATION ERROR: Inconsistent internal state" is now displayed.

Bug Fix: When resizing version 10 of the APLGrid object, the "ATL 9.00" string would flash momentarily in the center of the grid object.

Bug Fix: The ZIP32.dll and UNZIP32STATIC.dll were loading when APL+Win was started and in the wrong location in Windows memory. This could result in the maximum workspace size to be limited in some instances. The DLLs are now loaded when the first time a Zip object is created.

Bug Fix: Restored ability to set a Stop and Trace on line 1 of a function that was empty or started with a comment.

Bug Fix: Fixed some conditions that caused APL+Win to crash when executing

```
'#' ⎕wi 'Reset'
```

Bug Fix: Fixed obscure bug that caused APL+Win to crash due to buffer overrun in the system.

Please send any questions or comments to support@apl2000.com.

APL2000 Technical Support