APL64 Compatibility: Transition APL+Win Applications To APL64 Applications

Contents

Overview

This document outlines the differences between APL64 and APL+Win. Some differences require simple modifications to the application source code. Other differences only require a review of the associated APL64 programmer documentation available in the APL64 developer version.

To enable cross-platform capabilities, some APL+Win features are deprecated in APL64, and have been replaced by analogous APL64 features when possible. APL64 incorporates many new and improved features and greatly expands workspace memory and variable size compared to APL+Win.

APL64 Programmer Documentation

- Use the **Help** menu of the APL64 developer version to access the documentation by topic
- APL64 programmer documentation is available on-line
- Search all on-line documentation using the **Help | Search All Documentation | Search All Documentation with Google** dialog
- Select an APL64 system function, system variable, or primitive function, and click the F1 key to view context-sensitive documentation
- Use the **Help | Documentation Topic History** list to recall documentation topics previously viewed in the current APL64 developer version instance.
- Use the **Help | System Highlights | Recent System Highlights** dialog to keep up to date on APL64 news

Developer GUI (APL64 Programmer IDE)

- The APL64 developer version (APL64 programmer IDE) runs on the Windows 10/11 desktop
- Review the **Help | Developer Version GUI | Developer Version Quickstart** document to learn about the new and improved features of the APL64 programmer IDE.
- Panes (subordinate Windows) in APL64 can be docked to the main Window or floated independently on multiple monitors
- The history pane is the region of the main window where the executable APL statements, and any associated results, are illustrated.
- By default, the APL64 history pane is configured in traditional, editable format like in APL+Win
- Other history pane formats are available in APL64 selected from the **Session | History Format** menu
- The APL64 virtual keyboard is accessed using the **Session | APL Virtual Keyboard** menu or the keyboard shortcut Ctrl+B toggle.
- When the APL64 classic history pane format is selected and a history line is edited, but not executed, the left column glyph is a delta (Δ) indicating the status of that line
- In APL64, text selections may be linear (Shift + Arrow Keys) or rectangular (Shift + Alt + Arrow Keys)
- In APL+Win, when text in the history pane is selected and Enter/Return clicked, the selected text is removed and the caret moved to the next line of the history. In APL64 this is an option. Refer to the **Session | Use Selected Text on Enter in History** menu for additional information
- In APL64, the F6 and Shift+F6 keyboard shortcuts for moving the keyboard focus have been deprecated. Use Ctrl+Tab and Ctrl+Shift+Tab keyboard shortcuts instead.
- In APL64, statements in the history pane may include […]-prefixes to indicate the cause and effect of interpreter actions. The [imm] prefix precedes immediate execution errors, and the [object;event] prefix precedes callback execution statements. Refer to the documentation of the APL64)OUTPUT

system command for details. In APL+Win, the history pane output may not indicate the cause of an action.

- In APL64, set the short/long date format for loading, saving, copying, and dropping workspaces in the **File | Emit Timestamps in Long Date Format** menu. In APL+Win, see the Options/Preferences dialog box
- Entering hexadecimal values starting with the 0R notation is enabled by default in APL64. The APL+Win configuration parameter, [Experimental]EnableHex = 3, is deprecated.
- APL64 provides the Ctrl+D shortcut in place of Ctrl+K shortcut to hide and unhide the Debugger and State Indicator panes.
- The **File | Clear** menu option for clearing the active workspace in APL+Win has been moved to the **Session | Clear Current Workspace** menu in APL64.
- In APL+Win, a double right mouse click will open the object, if any, under the mouse pointer. This is the default action in APL64. In APL64, if desired, the Windows standard context menu can be presented for a right mouse click. See the **Objects | Right Mouse Double Click: Open Object Named as Caret** checkbox.
- The buttons in the APL Statements History dialog are toggle buttons. Multiple clicks will select the available options. An option is available in this dialog to 'Select Last Statement'. This option is available within the **Edit | APL Statements From The History | APL Statements History** dialog.
- In APL64 **)ed** [Enter Key], **)edit** [Enter Key] and **⎕Edit ''** [Enter Key] will open the default editor type
- Editor Autosave Options in APL64 replaces and enhances the APL+Win Resume Pending Edit Sessions feature. Use **Objects | Editor Autosave Options** to access the autosave options. For more information read the documentation: **Help | Developer Version GUI | Editing APL Objects**
- The split stile glyph is unsupported in the APL64 font, and therefore cannot be displayed in the history, the edit sessions or in ⎕av.
- In the history pane or command line, APL64 supports execution of APL statements that begin with a control structure statement

Find/Replace

- In APL64 the Find/Replace dialog is presented within the pane containing the keyboard focus, so that if the pane is floated, the find/replace dialog will be conveniently located.
- In APL+Win, when the Find/Replace dialog has the Search Type option set to APL Tokens, matching occurs regardless of the setting for the Case sensitive option. However, in APL64, the case sensitive setting is enforced.
- When the Ctrl+F3 shortcut is used to search for current token, the Find tool must be opened then closed to cancel the find highlights. This is analogous to Visual Studio. An option is provided to use the legacy APL+Win match highlighting behavior via the **Edit | Use Legacy Find Match Highlighting** checkbox.
- The Ctrl+R keyboard shortcut to present the Replace dialogue box is deprecated in APL64. Use the Windows standard Ctrl+H keyboard shortcut instead.

APL64 Object Editors

- The APL64 function and variable editing features are enhanced
- The editor panes may be docked to the main window or independently floated on multiple monitors

- Use ⎕EDIT system function open an editor in code and obtain the edted results. Refer to the ⎕EDIT, ⎕EDITEVENTS and ⎕EDITINFO sections of the **Help | APL Language | System Functions** documentation. Replacements are available for the APL+Win onEditEnd, onEditSaved and onEditStart events.
- For comprehensive documentation see the **Help | Developer Version GUI | Editing APL64 Objects** menu.

Function Editor

- An APL64 function editor provides two formats: traditional and comparative
- APL64 function editing is non-destructive by default, so that function edits can be preserved and undone/redone between editing instances and APL64 instances. Refer to the **Objects | Function Editor | Function Editing Change Records** menu item.
- An APL64 function editor increases the maximum number of characters per line to 9600
- The number of lines in an editor in APL64 is not limited to 10,000 as in APL+Win

Variable Editors

- APL+Win provides separate numeric and text editors for arrays up to rank 2
- APL64 provides a unified editor for arrays of mixed data type and any rank
- APL64 provides the)EDSS and ⎕EDSS spreadsheet-style editors for arrays with mixed datatype or rank up to two (2).

Multi-line Statements

- Using the Command Line, multi-line APL64 statements may be created and executed as a unit
- When the APL64 classic history pane format is selected the Command Line is optional
- A multi-line APL64 statement in the history pane has its subordinate lines grouped together

User Commands

- The user command file distributed with APL64 is derived from the APL+Win user command file
- The user command files for APL64 have been updated to replace ⎕CALL usages with the appropriate APL64 system functions.
- APL programmer-developed user command files may require similar modifications for compatibility with APL64
- User command]WSE is not available in APL64.
- User command]PCCOPY is not available in APL64. The reason is that the user command requires the APL+PC interpreter that is not compatible with the 64-bit Windows operating systems.
- User command]KB is deprecated APL64.

Debugging Functions and Executable Statements

- The debugger is used to display APL functions or expressions while they are executed step-by-step (tracing execution), and when a function stop is in effect (execution suspended)
- In APL64, the debugger and state indicator panes are the analogue of the APL+Win Code Walker
- The APL64 debugger and state indicator panes can be docked or floated

- The APL64 debugger and state indicator panes are configured to be displayed automatically only when needed. Configuration options are available to duplicate APL+Win non-automatic behavior, using the **Session | Debug** dialog
- Setting function stops in APL64 is the same as in APL+Win
- In APL64 function stops may be placed on empty and comment lines
- Tracing function execution in APL64 is the same as in APL+Win

APL64 Configuration

- APL64 configuration parameters are retained between APL64 instances in the ⎕UserPath folder
- The APL64 configuration parameters file has xml-format, rather than the ini-format of APL+Win
- In APL64 user and application information is stored separately at the discretion of the programmer
- APL64 command line arguments are documented in the **Help | Command Line Arguments** menu
- The **Options | APL+Win Configuration Conversion** utility is available to move applicable APL+Win configuration parameters into the APL64 configuration file
- The APL64 xml-format configuration file may be manually edited using the (free) Notepad++ utility, available here: https://notepad-plus-plus.org/downloads/.
- The APL+Win [Config]Responsiveness = n option is deprecated in APL64.
- The APL+Win Multiple Execution option is deprecated in APL64.
- The APL+Win APLW.INI [Config]ShortNames option is not currently supported in APL64.
- The APL+Win APLW.INI [Session]NavigateDirs option used to control whether to load a workspace without changing the current directory is not currently supported in APL64.

Loading APL+Win Workspaces

- APL+Win (.w3) format workspaces may be loaded or copied into APL64
- A suspended state indicator in an APL+Win workspace is not preserved when the workspace is loaded into APL64
- APL64 does not support saving workspaces in APL+Win format
- An APL64 (ws64) format workspace is saved in APL64 (ws64) format
- The c:\APLWINxx\Examples\*.w3 workspaces have been converted to APL64 *.ws64 workspaces and are available with the installation of APL64 in the c:\Users\<username>\AppData\Roaming\APLNowLLC\APL64\Examples folder.
- The c:\APLWINxx\Tools\*.w3 workspaces have been converted to APL64 *.ws64 workspaces and are available with the installation of APL64 in the c:\Users\<username>\AppData\Roaming\APLNowLLC\APL64\Tools folder.

APL64 User Tools

- APL64 programmer-developed tools may be created, and used the same as in APL+Win using the **Options | Configure User Tools** dialog.
- User tools are enhanced in APL64 so that requests for input, (⍵ or ∇), can include associated text prompts
- Refer to the **APL64 Developer Version Help | APL64 Developer Version GUI | User Tools** menu item documentation for details

Interpreter

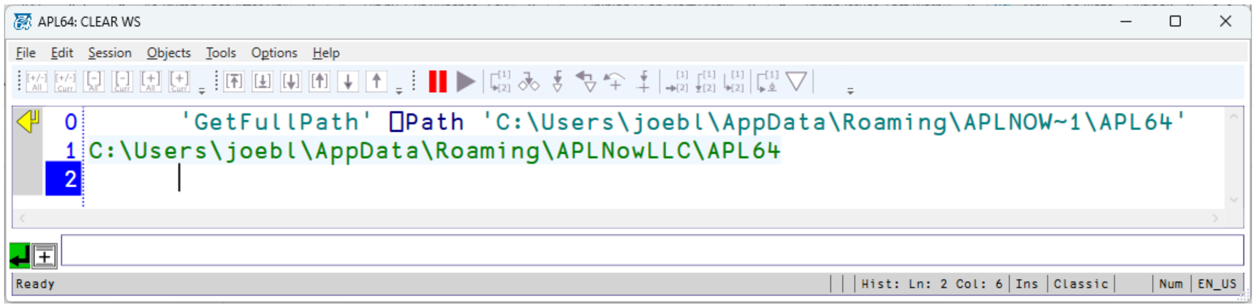
- The APL64 interpreter is a 64-bit interpreter compared to the 32-bit interpreter of APL+Win
- APL64 is built using Microsoft .Net which is an open-source, cross-platform, development environment. APL+Win is built using C/C++ and Microsoft MFC components targeting only the Windows desktop environment.

System Functions

- APL64 includes many new and improved system functions. Refer to the **Help | APL Language | System Functions** document for details.
- '#' ⎕wi 'caption' returns DEPRECATED FEATURE in APL64. The new system variable ⎕DEVCAP replaces '#' ⎕wi 'caption'.
- ⎕SCOM replaces, and significantly enhances the features of ⎕ARBIN
- The ⎕CSE system function has been enhanced, with increased performance, in APL64. No separate installer is necessary because the APL64 C# Script Engine is incorporated into APL64. Microsoft has modified many security features of the cross-platform .Net Core, so the APL64 ⎕CSE system function cannot be used to create and control a WPF or System.Windows.Forms GUI.
- The ⎕SKD, string key dictionary, system function replaces the analogous ActiveX component of APL+Win. An ⎕SKD dictionary may be used to store and access an APL64 variable.
- Most assembly language functions in the APL+Win \Tools\Asmfns.w3 workspace have been replaced in APL64 by system functions, e.g., ⎕WHERE, ⎕SSTOMAT, etc.
- The default ⎕mf mode is 4 in APL64 that adds two new columns to the ⎕mf output matrix. These columns document the execution time and count for compiling each line of execution.
- ⎕MOM and ⎕MSELF are deprecated in APL64. Consider using ⎕SKD dictionaries instead.
- The syntax of ⎕LOG has been modified. ⎕LOG is designed to be operating system cross-platform. Refer to the documentation of ⎕LOG for details. The output of ⎕LOG in a programmer-specified log file, or the operating-system event log file, is now in Unicode format.
- ⎕WCALL'GetModuleFileName' points to the APLNow32.exe component of APL64 and not the APL64 executable. Use 10⊃⎕SYSINIT (⎕IO = 1) which returns the full path of the APL64 executable.
- The behavior of ⎕SYS[24] is changed in APL64. In APL+Win, this returned the error code for the last host error (equivalent to ⎕WCALL 'GetLastError'). In APL64 this is not used and always returns a 0.
- The behavior of ⎕WGIVE is changed in APL64. In APL+Win, a positive ⎕WGIVE argument value is a count of how often the interpreter checks the Windows message queue to allow events to be processed. In APL64 the message queue doesn't exist in the same sense as in APL+Win. In APL64 a positive ⎕WGIVE argument means "enable callbacks or multiple execution injection" to happen. Any positive ⎕WGIVE argument in APL64 has the same effect. In APL64 a ⎕WGIVE argument of 0 or ¯1 has the same action as in APL+Win. In APL664 a positive ⎕WGIVE argument had no detrimental effect of performance as such an ⎕WGIVE argument has in APL+Win.
- '#' ⎕wi 'hwndmain' returns DEPRECATED FEATURE in APL64. Use **9⊃⎕sysinit** (⎕IO=1) to obtain the main APL64 window handle
- In APL64 ⎕SYS[2] (⎕IO=1) is assignable, so the APL+Win [Config] specification (fsync_dynamic=1) is deprecated
- The behavior of ⎕SINL is changed in APL64. Unlike APL+Win, all local variable names are indented uniformly and sorted in alphabetical order for each function before displaying them.

- The behavior of ⎕IDLIST,)FNS, and)VARS is changed in APL64. Unlike APL+Win, function and variable names are sorted in case-insensitive alphabetical order. IN APL64 the filter rules, including wildcard notation and regular expressions, have been enhanced in these system functions and commands. When specifying an optional argument that is a start letter or string, these system commands now only return the names that match exactly and not the subordinate names like in APL+Win.
- ⎕SYS[33] added in APL64 to control whether ⎕INBUF is immediately cleared when an error occurred. The default is 1. Set to the value to 0 to restore APL+Win behavior. Refer to the documentation **Help | Developer Version GUI | Using ⎕Inbuf** for details about ⎕INBUF in APL64.
- Syntax like '#'⎕WI 'PF'... in APL+Win is replaced by the new ⎕PF system function. Refer to **Help | APL Language } System Functions** for the details.
- The behavior of ⎕COPY, ⎕PCOPY, ⎕LOAD, and ⎕XLOAD is changed in APL64 with respect to runtime workspaces. The previous concept of a runtime workspace in APL+Win is not applicable in APL64. Therefore, the previous requirement that the workspace be a runtime workspace when using these system functions in a runtime application created with the Windows Runtime Executable (WRE) facility in APL64, does not apply.
- ⎕wi forms display with another icon on the Windows Taskbar. The reason for this is that ⎕wi forms are instantiated in APLNow32.exe that runs as a process external to APL64.
- In APL64 19⊃⎕SYS returns a seven-digit integer, where the first four digits represent the major component of the version number. The fifth digit represents the minor component of the version number in the range of 0 – 9. The sixth and seventh digits represent the build component of the version number in the range of 0 – 99.
- ⎕SYSVERN returns the version number as four components: major, minor, build, and revision.
- ⎕watchpoints: In APL64 the message displayed when a suspension is triggered by a watchpoint is "WATCHPOINT TRIGGERED", compared to "WATCH POINT TRIGGERED" in APL+Win.
- In APL64, when the APLGUI form is created in the session ('fm' ⎕wi 'Create' 'Form'), overlapping the developer version main window, and the APL64 programmer displays a tooltip in the main window, the main window will become on-top of the ⎕wi form. To avoid this behavior, turn off the tooltips in the developer version or add the 'Show' method to the ⎕wi form creation statement ('fm' ⎕wi 'Create' 'Form' 'Show').
- Learn about the improvements of ⎕ and ⍞ in APL64 from the **Help | Developer Version GUI | Quad and Quote** documentation.
- ⎕xntie converts a short (abbreviated) filename to a long filename. Unlike APL+Win, these extended file system functions in APL64 operate exclusively on full pathnames to be cross-platform compatible. So, the solution in this instance is to incorporate the ⎕path system function to convert the abbreviated pathname to the full pathname and use that value to compare to ⎕xnnames or ⎕xfnames info.

  fullPath←'GetFullPath' ⎕Path abbreviatedPath


APL64: CLEAR WS
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 2 Col: 6 Ins Classic Num EN_US

```
      'GetFullPath' ⎕Path 'C:\Users\joebl\AppData\Roaming\APLNOW~1\APL64'
C:\Users\joebl\AppData\Roaming\APLNowLLC\APL64
```

- ⎕cfinfo: The first element is not used by APL64 due to a security risk and always returns a 0 instead of the file handle of the tied file.

System Commands

- APL64 has new and improved system commands. Refer to the **Help | APL Language | System Commands** documentation for details.
- The)RSAVE system command is exclusive to APL+Win and not needed in APL64
- The evolution level for determining compatibility with APL+Win is set with the)EVLEVEL system command and is now stored in an APL64 saved workspace. The current evolution level is also reset to 2 (default) when clearing the workspace.

System Variables

- APL64 has new and improved system variables. Refer to the **Help | APL Language | System Variables** documentation for details.
- Changed behavior of ⎕TCNL,⎕TCLF (CRLF sequence): APL64 considers the ⎕TCNL,⎕TCLF (CRLF sequence) as a single line separator. In APL64, a single blank line is no longer generated for the ⎕TCLF.
- In the APL64 Developer version, ⎕TCFF generates output like ⎕TCNL. To simulate the APL+Win action of executing ⎕TCFF, use the **Session | Scroll Up History** menu item. To clear the current session history content, use the menu **Session | Clear History** menu item. These actions are also available in the context menu of the session history or session command line.
- ⎕TCBS and ⎕TCESC are Unicode characters, and not actions that remove or generate output. In APL64, Unicode characters, except for ⎕TCFF, ⎕TCNL and ⎕TCLF, are rendered to the session history using their associated glyphs.

Excel Interface

- APL64 provides the built-in ⎕XL system function to read/write Microsoft Excel format files with superior performance.
- The ⎕XL system function is cross-platform and does not require Excel to be installed
- APL+Win depends on an ActiveX interface to Excel which must also be installed. APL64 does not depend on that ActiveX interface.

Performance Evaluation

- The new APL64 ⎕PROFILE and)PROFILE tool may be used to obtain a fine-grained operations performance profile of all or a portion of an APL64-based application system. To learn more, read the **Help | APL Developer GUI | Using ⎕PROFILE** documentation.

Miscellaneous

- c: executed in APL+Win returns a SYNTAX ERROR exception, but in APL64 the exception message is **OUTER SYNTAX ERROR: LINE [1] Label not allowed in expression**
- The **X,←** concatenation syntax is supported but isn't optimized as in APL+Win.
- UTC time is universally supported in APL64. The following system functions and commands save and report in UTC time: ⎕AT, ⎕PSAVE, ⎕SAVE, ⎕TS,)COPY,)SAVE and Component file operations
- In APL64, the underscore (_) is permitted as the first character in variable and function names
- In APL+Win, when the execute primitive (⍎) is used in a function, the implicit output isn't suppressed for the)OUTPUT OFF case. This is corrected in APL64. In APL64 an exception is correctly thrown for the)OUTPUT ERROR or)OUTPUT STRICT cases.
- ι⍬ no longer returns LENGTH ERROR. In APL+Win, the same APL expression returned LENGTH ERROR exception. There is no error message in APL64 because the enhancement to monadic Iota (Index Generator) to support arrays in the right argument in APL64, and the empty case is consistent with higher rank cases.
- ι¨ now returns DOMAIN ERROR in APL64. In APL+Win, the same APL expression returned the error message LENGTH ERROR. The error message has changed due to the enhancement to monadic Iota (Index Generator) to support arrays in the right argument and a character argument of any shape or rank is still a DOMAIN ERROR.
- In APL64, the execute primitive function (⍎) can execute a character string representation of an APL expression including new lines (⎕TCNL). APL+Win stops evaluation of the character string at the first ⎕TCNL.
- For the EN-US keyboard definition, because APL64 is Unicode aware, APL64 distinguishes between ⎕av[124+⎕io] (decimal Unicode 124) and ⎕av[254+⎕io] (decimal Unicode 8739). Decimal Unicode 8739 is the Unicode glyph specified for the Magnitude and Residue primitive functions. Therefore, unlike APL+Win, the Magnitude and Residue glyph is available from the US-EN keyboard using only Alt+M. For the EN-US keyboard Shift+\ generates the decimal Unicode 124 glyph. In the APL64 font, the decimal Unicode 124 and 8739 glyphs appear graphically identical.

⎕WI and ActiveX

- ⎕WI and ⎕WCALL are Win32 technologies, which are not cross-platform compatible. These APL64 system functions can be used in the APL64 Developer version and in an APL64 Windows Runtime Executable (WRE), but not in an APL64 Cross-platform component (CPC).
- When an APL64-based application is exclusively targeting the Windows desktop environment, the ⎕WI and ⎕WCALL system function features are available for use, including GUI forms and ActiveX.
- APL64 includes the APLNow32.exe component to support these features. The APLNow32.exe component does not require a separate installation or ActiveX registration on the target workstation. The configuration of APLNow32.exe is like that in APL+Win.
- The APL+Win ActiveX-based components such as APLDraw and APLGrid may be used via ⎕WI. Installing APL64 registers its own APLGrid that supports the same ProgID name, APL.GRID, as in APL+Win, for aiding in migrating APL+Win applications to APL64.

  After installation of APL64, the APLGrid in APL+Win is temporarily inoperable. To restore the operation of the APLGrid in APL+Win, re-register the APLGrid DLL in APL+Win in a DOS Command Prompt window run as administrator with the command like: *C:\APLWIN19\REGSVR32 APLGRID.DLL*.

To use the new APLGrid in APL64, it must be created with the new ProgID name: APLNOW32.GRID, e.g., 'form.grid' ⎕wi 'Create' 'aplnow32.grid'.

- APL64 may be a client of the APL+Win ActiveX engine by using ⎕WI. The APL+Win ActiveX engine can access only APL+Win (w3) workspaces.
- Some ActiveX components may have compatibility issues with APL64 especially if the version of the ActiveX component is not up to date. Contact the vendor of the ActiveX component for update availability.
- The '#' ⎕WI 'hwndsplash' property and the [Config]SplashBitmap .ini/command line argument for displaying a custom splash screen at startup are deprecated in APL64. For a custom splash screen use the APL64 command line arguments SIP, SIT, or SIC. Use the ⎕SPLASH system function to hide the splash screen.
- THE 3⊃⎕WCALL 'W_Init' used to retrieve the command line arguments in APL+Win is deprecated in APL64. In APL64, use 10⊃⎕SYSINIT.

Runtime Version

- APL64 includes a royalty-free, perpetual license for its run-time components
- Select the APL64 runtime component type suitable for the target operating system platform.

Windows Runtime Executable

- An APL64 WRE (Windows Runtime Executable) is designed exclusively for the Windows 10/11 (x64) environment using a ⎕WI-based GUI
- Application system transition from APL+Win to APL64 is simple and worthwhile
- The APL64 runtime components are bound to an APL64 programmer-created, run-time ready workspace
- An APL64 WRE is created using the **Options | Create Runtime .Net Assembly | Create Windows Runtime Executable** utility
- An APL64 WRE is a single, ready-to-run, exe-format file targeting the Window 10/11 desktop/laptop environment
- All necessary components of the APL64 programmer's application system, including additional workspaces and files, may be included in an APL64 WRE

Cross-platform Component

- An APL64 CPC (cross-platform component) is designed to be used as a part of a .Net application running in the Windows, Linux, Android or Apple (x64) environment
- An APL64 CPC uses the .Net Nuget package format for complete compatibility with any .Net language on an operating system platform supporting .Net.An APL64 CPC is created using the **Options | Create Runtime .Net Assembly | Create Cross-platform Component** utility
- An APL64 CPC can utilize all APL64 features except those features that are based on Windows-only technology, such as ⎕WI, ⎕WCALL, ActiveX, and ⎕CALL.