Modifications Included in APL64 Pre-Production Version (2020.0.6)

Table of Contents

Overview

This document describes the enhancements and bug fixes incorporated into the fourth pre-production version of APL64 released in 2020.0.6. This document also describes the evaluator comments and APLNow responses based on the prior pre-production versions of APL64.

Enhancements

Minimize/Maximize Editors Pane and History Pane

APL64 Developer Version GUI: Editors Pane of the Main Window
The Editors pane section of the main window and the History pane section of the main window are shared depending on the position of the GridSplitter Bar between these sections. Double-clicking the left mouse button on this GridSplitter Bar will alternately minimize/maximize the Editors pane section of the main Window. The user may manually drag this GridSplitter Bar to any intermediate position to share the space proportionally between the Editors pane and History pane sections of the main window.

This enhancement will facilitate convenient viewing of editors and tool windows (such as Find/Replace) when the APL64 developer version main window is maximized.

Minimize/Maximize Debugger/State Indicator (SI) Pane

APL64 Developer Version GUI: Debugger/State Indicator Pane of the Main Window
The Debugger/State Indicator (SI) pane section of the main window and the remaining area of the main window are shared depending on the position of the GridSplitter Bar between these sections. Double-clicking the left mouse button on this GridSplitter Bar will alternately minimize/maximize the Debugger/SI pane section of the main Window. The user may manually drag this GridSplitter Bar to any intermediate position to share the space proportionally between the Debugger/SI panes and remaining area of the main window.

The history area and the non-debugger/si area of the main window have minimum heights, so that they are both always at least marginally visible.

New Menu: Objects | Editors Pane Format | Open Non-modal Object Editor In Prior Location

When this option is checked, a non-modal function or variable editor will be opened in the same floating location previously used for this named object.

This action is possible only if:
(a) Objects | Editors Pane Format | Layout: Editors Floated is user-selected and
(b) The object's editor is not already open and
(c) The named object exists in the Objects | Recent Object Names list.

If this action is not possible, the Objects | Editors Pane Format | Layout selection will be used to present the editor.

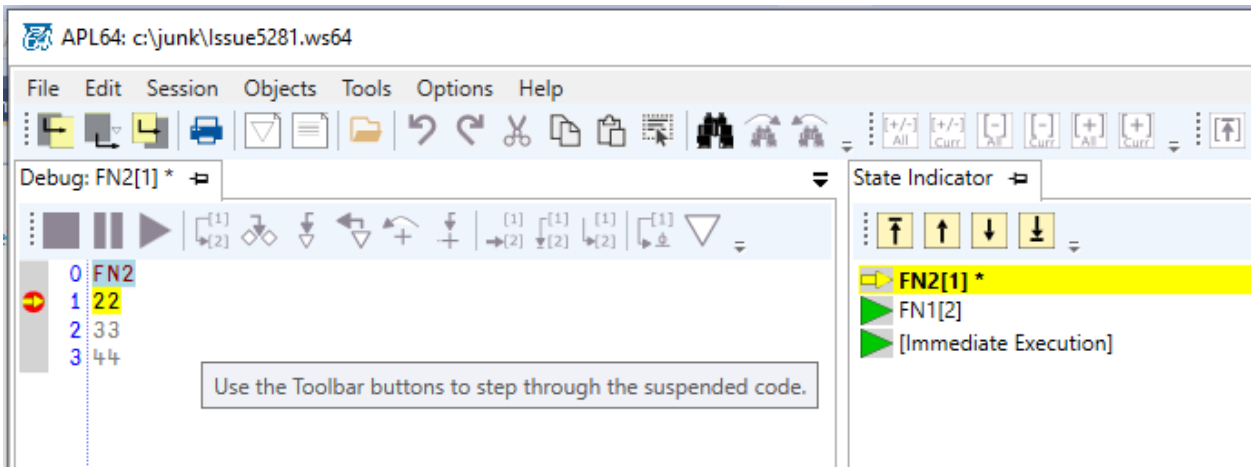
Floating editors is applicable the location of the floating editor on the workstation's display(s) does not intersect with the
location of the APL64 developer version main window on the workstation's display(s). If there is only one display available on the workstation, the APL64 developer version main window should not be maximized.

Toolbar Arrangements Saved for Main Window, Function Editor and Variable Editor

These elements of the APL64 developer version GUI have multiple toolbars which can be user-arranged. The user-arrangements will now be saved when these objects are closed for subsequent use in the APL64 xml-format configuration file.

State Indicator Toolbar Button Backgrounds Are Now Light Yellow

This background coloration will distinguish these buttons from the Control Structure buttons on the main window toolbar.



Autosave Function and Variable Editor Content

The Objects | Editor Autosave Options | Autosave Active is checked by default.

The autosave of Variable editor content is now supported.

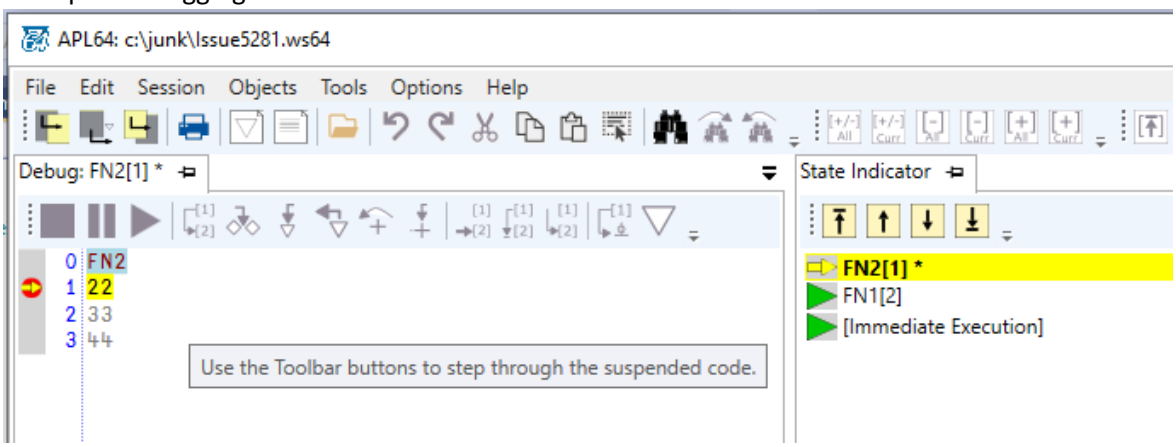
Menu: Edit | Bookmarks options are disabled when Classic history format is editable

Bookmarks are not supported when the Classic history format is editable, therefore, the Bookmarks' submenus are disabled.

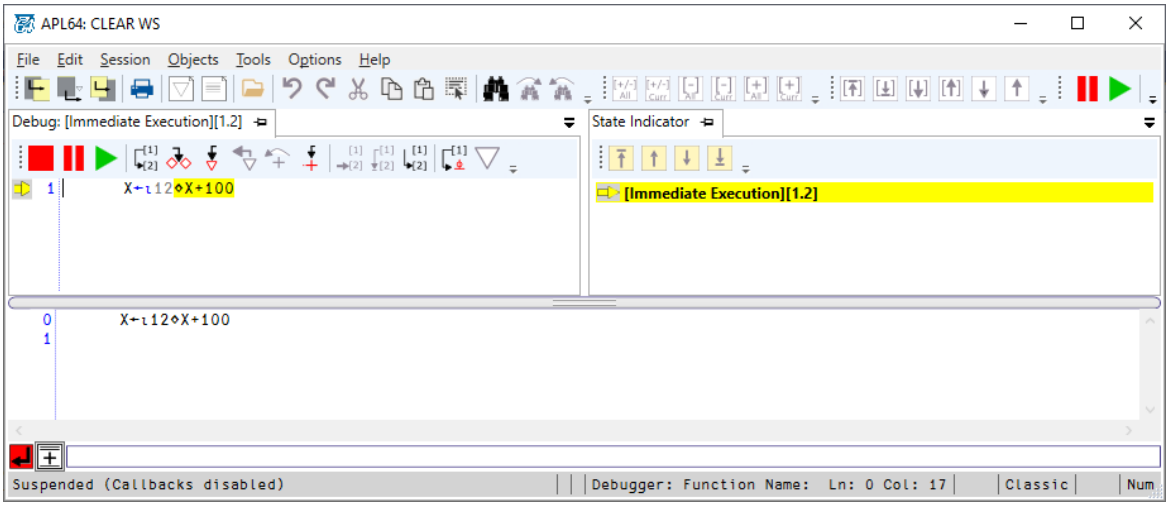
State Indicator (SI) for debugged function added to Debugger pane's caption

Note: The Debugger Pane caption also shortened Debugger.

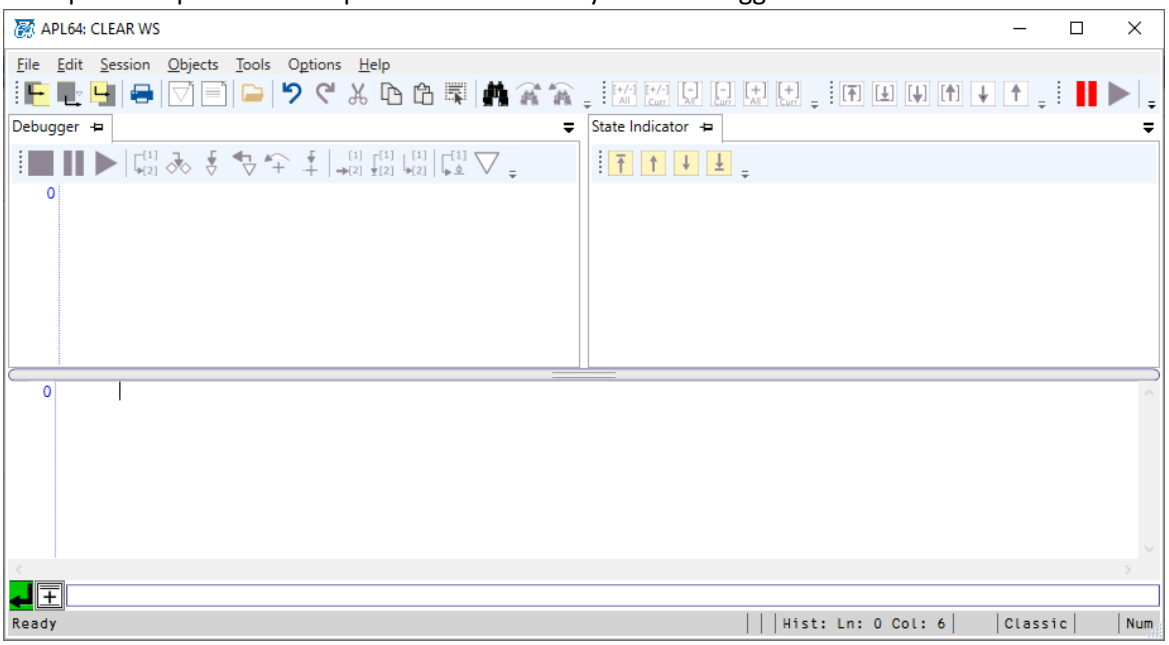
Example: Debugging APL64 user-defined functions:



Example: Debugging an APL executable statement:



Example: Interpreter not suspended and manually show debugger:



UTC time supported universally in APL64

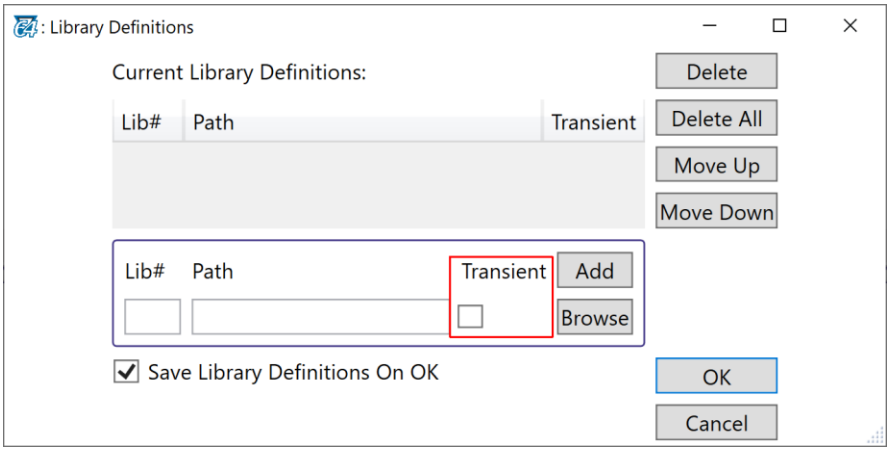
The following system functions and commands were updated to now save and report in UTC time: ⎕AT, ⎕PSAVE, ⎕SAVE, ⎕TS,)COPY,)SAVE and Component file operations.

The State Indicator debugger pane and mouse double clicks

The State Indicator debugger pane now supports the mouse double click actions for opening the object named at the caret in the editor. Similarly, a right mouse click also displays a context menu with a keyboard shortcut (Ctrl+Shift+O) to open the object named at the caret position.

In the Library Definitions dialog, the "Transient" checkbox is now unchecked by default

This will allow any changes in Library Definitions to be saved by default.



Step by primitive OVER function and **Step by primitive INTO function** debugger toolbar buttons implemented in Debugger Pane

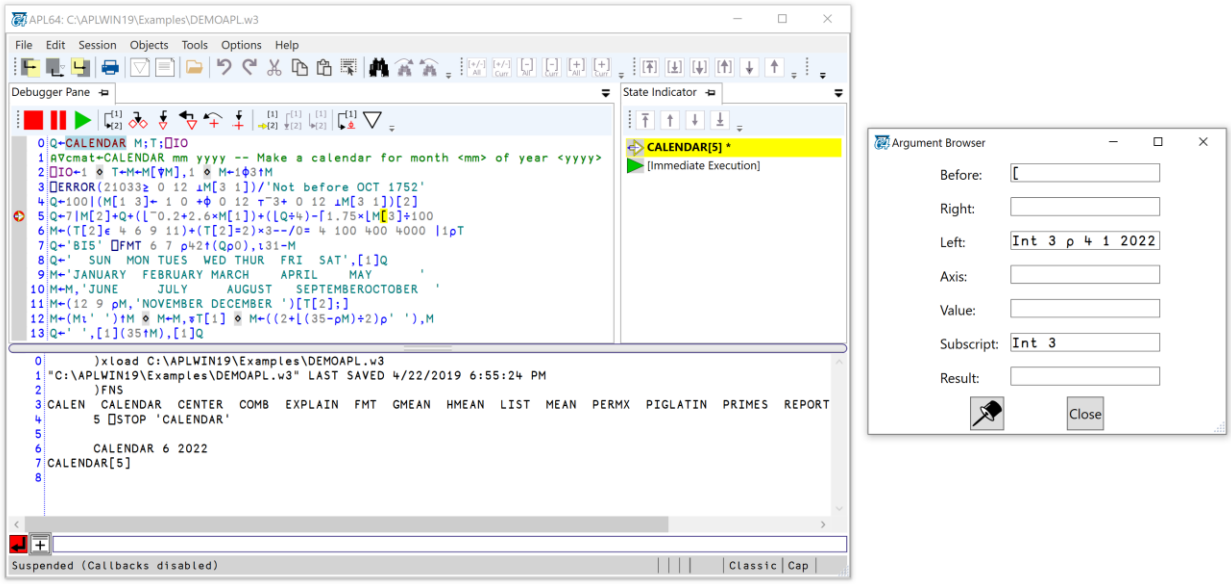
- Step by primitive over function (Ctrl+F10)
- Step by primitive into function (Ctrl+F11)

⎕cwxx Debugger Stepping-by-Primitive Variables [Work In Progress]

These six system variables provide a method to use the current values associated with a suspended function for debugging purposes. Refer to the System Variables document for the details.

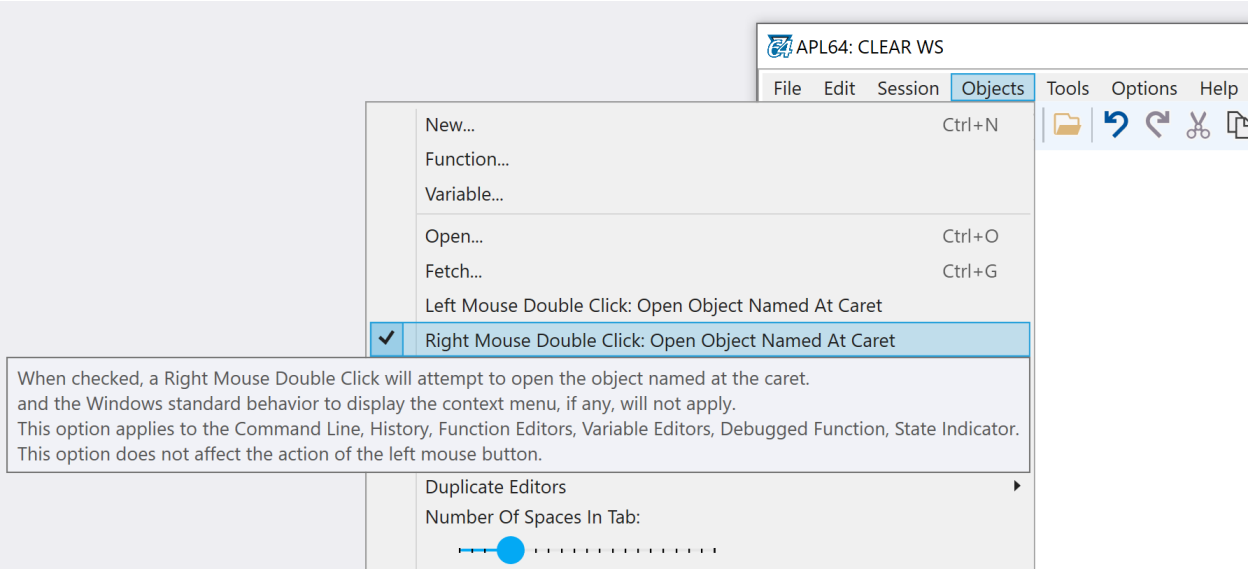
Argument Browser in Debugger [Work In Progress]

When you invoke stepping by primitive, Debugger displays an Argument Browser that has slots to display any of seven values that are defined at the point of suspension; the seven values are before argument, right argument, left argument, axis, value, subscript, and result. When you step over by primitive, the system has already evaluated the function on which it is stopped, so there will usually be content for the right argument and result. The left argument, axis, and subscript slots will have a display only when the function is dyadic, you have specified an axis along which the function operates, or you are indexing into an array, respectively. The value slot shows the result of an assignment (in which case there is no right argument).



Menu Objects | Right Mouse Double Click: Open Object Named At Caret set by default

The menu **Objects | Right Mouse Double Click: Open Object Name At Caret** is now set, by default. This replicates the right mouse double functionality in APL+Win.



Bug Fixes

The proper APL command s←0 ⋄ '3',⍪s+←1 returned SYNTAX ERROR

History didn't appear in list of active panes (accessed with Ctrl+Tab) until there was output to the history

Beware that the History active pane will only appear for an empty Classic history when it is editable.

The proper APL command ⍎⊃'12' '34' '5678' returned LENGTH ERROR error message

This happened because the execute primitive function didn't properly handle matrix arguments with more than one row.

The proper APL command a←'' ⋄ 1×a returned DOMAIN ERROR

The proper APL command ⎕wres←3.5 (or any floating value) returned DOMAIN ERROR

The proper APL commands ⍎0 0⍴0 and ⍎0 0⍴⍬ returned DOMAIN ERROR

The proper APL command aaa←⎕cm⍬ returned VALUE ERROR

The proper APL command ⎕cn ⍬ returned SYSTEM ERROR:Exception: Object reference

New menu: Session | Classic History Format | When At End Of Editable History ↓ To Command Line

If the Classic History format is selected and it is editable and the cursor is on the last line of the history (above the APL command Line), pressing the down arrow key (↓) will move the focus to the APL Command Line when checked or add a new line in the history when unchecked.

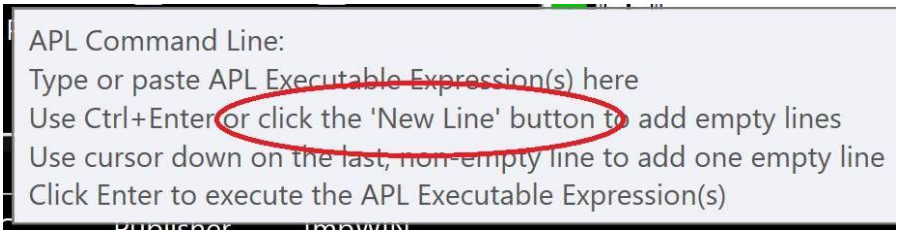
The OEM6 keyboard key, which for FR-FR keyboard definition produces no glyph, was mistakenly included in the list of keys which could be used for a user-defined tool.

This resulted in the Options | Configure User Tools... opening as a blank form then after a few seconds terminating APL64.

Incomplete error message: "XFHOST ERROR _sopen 1005 13 32 The process cannot access the file because it is"

The error message was truncated. The complete error message is: "XFHOST ERROR _sopen 1005 13 32 The process cannot access the file because it is being used by another process."

Out of date tooltip in the APL Command Line (see below)



Setting ⎕STOP on line 1 of a function with a comment didn't work

This only occurred with ⎕stop and not when the stop was set in the function.

Executing the)si system command incorrectly displayed a blank line in the history

Loading an APL+Win workspace incorrectly updated the functions with the current timestamp

□AT returned TYPE CHECK ERROR: Expected… for unknown object names

Pressing the Enter key in an undocked function editor window scrolled the function to the bottom

Evaluators' Comments & APLNow's Responses

This section contains the summary of the evaluators' comments (feedback) and APLNow's responses to the Pre-Production Version of APL64 (2020.0.5.10649). We would like to thank the evaluators for their contribution.

focus does NOT jump to the Command window

I can now type APL expressions in the Session Window and the results appear on pressing Enter: that is, focus does NOT jump to the Command window. That, combined with the ability to clear the Session window partially – noted that in the April l2022 document—is very very welcome as it returns to the familiar APL+Win behavior. (might raise questions about the need for the Command window … I've got used to and quite like the Command (or Input) window).

Developer's Response: If your preference is to enter APL expressions in the APL command window, ensure the following settings are in effect in APL64:

1. Select **Session | History Format | Classic**
2. Check **Session | Classic History Format | Is Editable**
3. Uncheck **Session | Classic History Format | Set Focus When Editable**

□cse 'methods' and 'properties' with APL+Win and APL64 are not identical

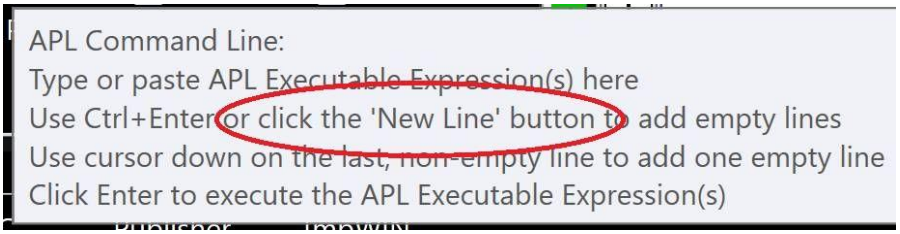
Correct. In addition, there are differences in the implementation and operation of the CSE in APL64 and APL+Win. Refer to the documentation in the Help menus below:

(a) **APL+Win Compatibility | Compatibility Modifications for Existing APL+Win Applications**
(b) **APL Language | C# Script Engine Manual**

Should □cse in APL64 enumerate in lowercase?

In APL64, we have eliminated the check for case, so all □CSE methods, properties and event names can have any case: Upper, Lower, Mixed

Out of date tooltip in the APL command line (see below)



This typo will be corrected in a future release of APL64.

Suggest a system function, may be ⎕sts (for session state or status), that will enumerate every user setting in the active workspace

We will take your suggestion into consideration for a future release of APL64.

Suggest a system function, may be ⎕stw (workspace state or status), that will provide summary statistics on workspace objects

We will take your suggestion into consideration for a future release of APL64.

APL64 reports SYNTAX ERROR when trying execute: s←0 ⋄ '3', ⍎ s+←1

This has been corrected in the 2020.0.6 pre-production release.

Make '#' the default left hand argument of ⎕CSE such that ⎕cse 'methods' works in the same way as '#'⎕cse 'methods'

Implementing your request would preclude a potential future use for an empty left argument to the ⎕CSE system function. Therefore, the suggestion is not currently being considered.

Add an Execute method such that fully qualified c# statements can be executed directly and without creating an instance of ⎕CSE

The behavior you request cannot be implement in the general case because in a single APL64 CSE statement, the required .Net assemblies cannot be identified and loaded. This is not a deficiency of APL64 ⎕CSE, but the innate behavior of .Net programming, i.e. the programmer must explicitly specify the .Net assemblies to be loaded.

An undocked function editor should remain visible at all times

The behavior you request is based on Microsoft-deprecated MDI behavior which cannot be used in APL64. The Pane document mechanism is not based on MDI technology. The Ctrl+Tab shortcut should be used to move the focus among the panes in the APL64 developer version GUI.

The prompt: "This will end your APL Session" should be made topmost!

A modification to address this issue originally appeared in the 2020.0.5 pre-production release.

It is not correct that the APL64 Find/Replace dialog remains topmost even if the user Alt+Tab to another application

We agree and will research to see what can be done in the case of non-modal windows like Find/Replace, Ctrl+O, Ctrl+G.

Pressing Enter in an undocked function editor window scrolls the function to the bottom (or top)

This has been corrected in the 2020.0.6 pre-production release.

Suggest allowing double clicking a function displayed in the State Indicator to open in the function editor

This option will be available in the next APL64 pre-production version.

Suggest allowing a double mouse click on debugger pane caption to open function maximized in editor

This capability is not currently supported in the AvalonDock GUI component used by both Visual Studio 2019 and APL64. The currently defined action is:

(a) Right Mouse click on function editor pane tab then select Float
(b) Click full screen button in upper right of floating function editor pane

The APL command ⍎⊃'12' '34' '5678' returned LENGTH ERROR error message

This has been corrected in the 2020.0.6 pre-production release.

What precise minimum set of APL64 files must I include in my app folder to allow the app to run (in developer mode)?

When the 'app' is not a run-time application, but is a command line usage of APL in developer mode, the APL64 installer should be used to install the APL64 components to the desired location and then the application-specific components should be copied to the same location. **Note**: The list of APL64 files can change and the list is not published by APL2000.

However, when the 'app' is a runtime application, use the **Options | Create Runtime .Net Assembly | Create Windows Runtime Executable** utility to define and create the single-file application. Documentation on this utility is accessed in the menu **Help | Developer Version GUI | Create Windows Runtime Executable**.

Results from my ICS testing

I loaded an APL+Win workspace—this creates Excel like pivot tables & uses ICS quite extensively; the workspace had □lx set. The function ran seamlessly. I tried this with the first version of APL64. Finally (very happy to report), success!

Function editor disappears when I press Ctrl+F or Ctlr+H, requiring Alt+Tab to show it again

Here is what may be happening in your scenario. The function editor pane and any other panes in APL64 which can be floated, behave the same way as floated panes in Visual Studio 2019. This is because they both use the same GUI control 'Avalon Dock'. When panes are floated, the user must provide sufficient display space so that at least a portion of the floated pane is not within the display area of any other pane or window. Or else, refrain from using floating panes.

Ctrl+Enter [keyboard shortcut] does not open a blank line in the APL Session like in APL+Win

This enhancement will be included in the 2020.0.6 pre-production release of APL64 for the editable Classic history format.

A function disappears in the function Editor which displays entirely blank

We are awaiting a fix for this from the third-party developers responsible for the toolkit used to build the editors in APL64. When the fix is available, it will be implemented and tested before it is released in a future release of APL64.

Tooltips (sometimes) not showing for the debugger pane toolbar

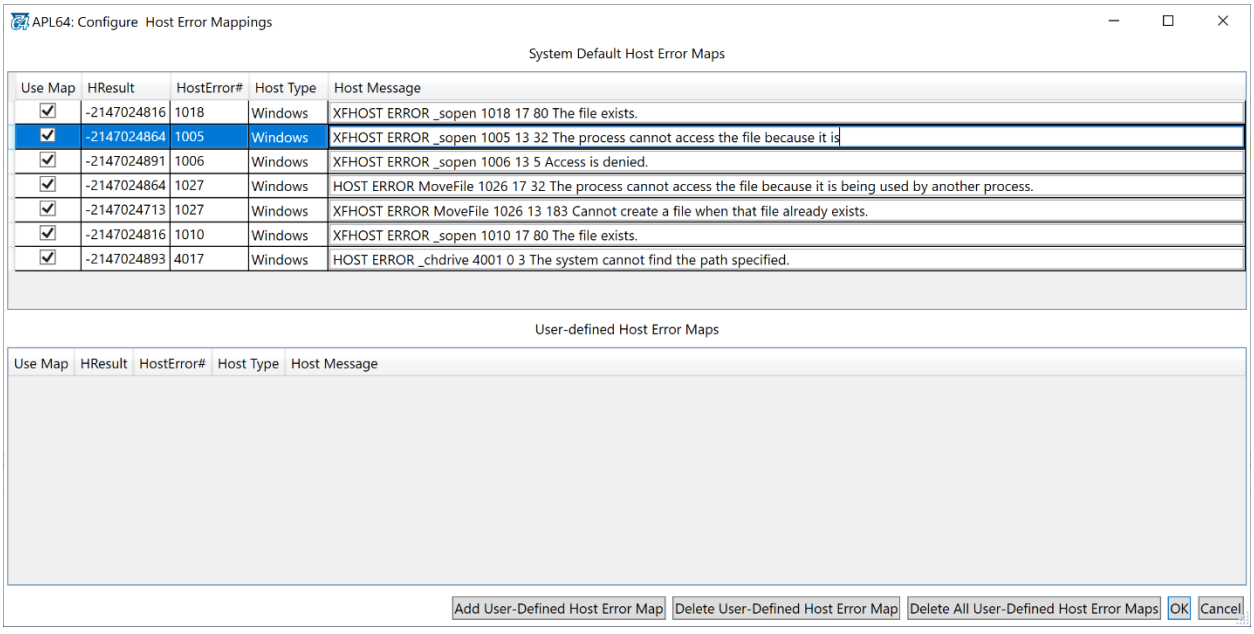
We have replicated your issue and currently investigating it. A fix for this will be available in a future pre-production release of APL64.

When I try to install this [2020.0.5] release, I get the message "A more recent version of APL64 is already installed on this computer."

Per step 2 in the install instructions in my message, you must uninstall the prior version before installing the latest version.

Incomplete error message: "XFHOST ERROR _sopen 1005 13 32 The process cannot access the file because it is"

For your information, the problem was that the host error mapping was incomplete for host error 1005 as demonstrated in the highlighted row in the dialog below:



This has been corrected in the 2020.0.6 pre-production release.

MessageBoxes displayed by a C# ActiveX called by APL64 do not display on the screen

This behavior was attributed to an error in the user's application.

Commenting/Decommenting a block of lines in the function editor can comment/uncomment one too many lines when the cursor is positioned outside of the selection block

We agree that the current behavior should be changed. Be on the lookout for this change in a future pre-production release.

Installer at startup gives the message: APL64 cannot be installed on systems with .NET Core version lower than 5.0

We are investigating the issue. In the interim, you can go to *https://dotnet.microsoft.com/en-us/download/dotnet/5.0* to download and install the installer **.NET Desktop Runtime 5.0.16 x64**.

Setting ⎕STOP on line 1 of a function with a comment doesn't work

We verified that this occurs with ⎕stop and not when the stop is set in the function. This will be investigated and corrected in a future APL64 release.

The font size doesn't change in the Debugger Pane

This is a bug and will be corrected in a future release of APL64.

a←'' ⋄ 1×a fails with DOMAIN ERROR in APL64, but works in APL+Win

This has been corrected in the 2020.0.6 pre-production release.

)si system command displayed a blank line in the APL Session [history]

We agree that **)SI** when there isn't any state should not produce a blank line. This has been corrected in the 2020.0.6 pre-production release.

⎕wres←3.5 (or any floating value) returned DOMAIN ERROR

This has been corrected in the 2020.0.6 pre-production release.

⌽0 0⍴0 and ⌽0 0⍴⍬ returned DOMAIN ERROR

This has been corrected in the 2020.0.6 pre-production release.

aaa←⎕cm⍬ returned VALUE ERROR

This has been corrected in the 2020.0.6 pre-production release.

⎕cn ⍬ returned SYSTEM ERROR:Exception: Object reference not set to an instance of an object.

This has been corrected in the 2020.0.6 pre-production release.

Floating a docked pane leaves the docking area visible taking up space in the session and does not autohide

Currently an AvalonDock pane does not provide a PaneFloated event which is the essential trigger to resize an empty editor docking pane. We have asked the vendor for this modification.

Suggest allowing floating editor panes to be listed when mousing over the APL64 icon in the Windows Taskbar like in Visual Studio

We like the Visual Studio behavior too. We will ask the vendor if this behavior can be available in AvalonDock.

When undocking a function editor could you make it automatically use the right half of the screen on the entire screen height (minus the taskbar)?

Not with the currently available AvalonDock toolkit in APL64. This is because there is no event for an AvalonDock pane such as Float or Dock or preview versions of these events. If there were a Float or PreviewFloat event, then this request would be trivial. We have asked the vendor of this toolkit for events like these, but they have not agreed to implement them.

Menu option File/Clear... does not exist

Per the **Help | APL+Win Compatibility | Compatibility Modifications for Existing APL+Win Applications**:

*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.*

We need those debugger panes to away automatically when I'm no longer debugging and manually when desired (even if the code is still stopped)

By design if there remain any potential lines of code in the debugged function pane the debugger/si panes will remain open when the display of the debugger/si is set to 'automatic' mode.

The APL+Win-style non-automatic display of the debugger is available by making these configuration settings:

- Session | Debugger | Automatically Show/Hide Debugger
- Use Session | Debugger | Show/Hide Debugger (Ctrl+D) as necessary. The menu item heading with change appropriately

Is there a way to Import the last History Log automatically at session start (like it did in APL+Win)?

We will investigate if this is possible in APL64, considering that the workspace may be loaded with an operation □LX.

I can't get □PW to exceed 40 while "Set □PW based on window width" is set

This occurred when APL64 was previously closed with a maximum sized window. Upon restarting APL64, □PW returned an incorrect value. We'll investigate this.

On French keyboards, the Options | Configure User Tools... opened a blank form then after a few seconds terminated APL64

We've determined what was causing this issue. The OEM6 keyboard key, which for FR-FR keyboard definition produces no glyph, was mistakenly included in the list of keys which could be used for a user-defined tool. This has been corrected in the 2020.0.6 pre-production release.

Provide a TOOLS folder with mostly the same utility workspaces as APL+Win

This folder has already been planned for the production release of APL64. Similar with the Examples folder.

Why do you install APL64 by default in C:\Program Files\APLNowLLC\APL64 and not simply in C:\APL64 by default?

This folder is Microsoft's recommended installation location for Windows applications.

Why are you hiding the APL64.xml config file and the User Command Processor UCMD*.sf files in the hidden C:\Users\eric\AppData\Roaming\APLNowLLC\APL64 folder?

First, this user's path isn't hidden. And second, the user's path guarantees the user has R/W privileges to perform actions like updates to the XML configuration file.

Provide default settings which correspond to APL+Win behavior (wherever possible)

We have been attempting to do this all along, whenever possible. If you notice something that wasn't, please don't hesitate to point it out to us.

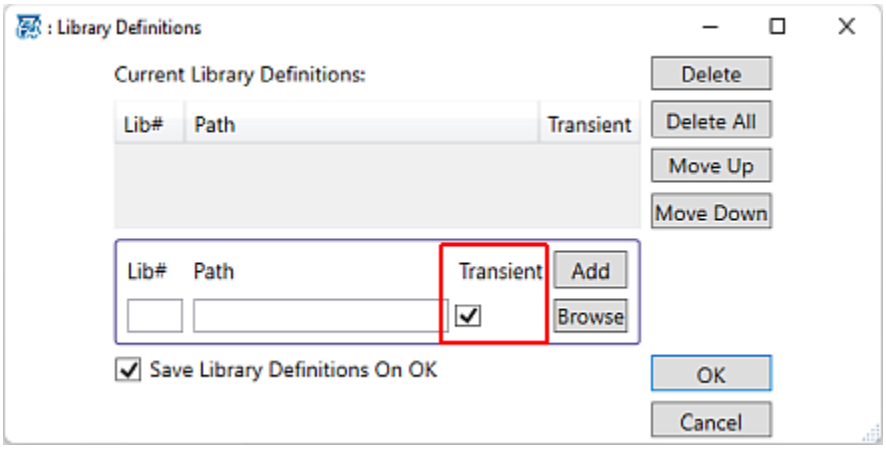
If I'm stepping through code and drop down to the session to try some code, I cannot resume stepping through the debugger unless I first click on the code listing in the debugger pane. Can this process be improved?

Currently this is by design considering the possibility that there is debuggable code elsewhere in the GUI (History/Command Line). Also one the keyboard focus is moved from the debugger to the history/command line, there is no way to the application to know the prior location of the keyboard focus.

If it becomes possible to add an additional behavior (switch) we will consider it.

In the Library Definitions dialog, the "transient" checkbox should be unchecked by default

We agree the Transient option should not have been checked by default. This has been corrected in the 2020.0.6 pre-production release.



I could well do without having resizable dialogs (e.g. Library Definitions) in APL64

Resizable dialogs in APL64 are by design and they don't require resizing when not necessary. The primary benefit of the resizable dialogs is to allow them to be visible on higher resolution displays above 2K.

Will the performance (not only APL primitives, but also Windows GUI) with APL64 in its final version 1.0 be at least as good as with APL+Win?

There can be no guarantee that every customer's existing APL code will run faster or at least as fast in APL64 than in APL+Win. Performance improvement is an ongoing process which will continue throughout the life of the APL64 product. Feedback from customers is appreciated which identifies areas which need performance improvement.

The Windows GUI tools in APL64 are identical with those in APL+Win, e.g. ⎕WI, so performance will similar. The ⎕WI GUI tools cannot be 'updated' to 64 bit code because ⎕WI is:

- Fundamentally based on Win32-based technology, which was deprecated by Microsoft in 1999
- Can operate only in the Windows operating system environment, so is available in APL64 only if the Windows OS is targeted

Could you keep the region gutter when commenting those lines in the function editor, even though the gutter will be entirely empty?

We've will investigate the issue to see whether it will be possible to retain the region gutter when commenting lines with control structure statements.

Function editor does not remember expanded/collapsed regions state when re-editing the function

We will investigate the issue and make a correction in a future pre-production release.

Ctrl+Shift+S (Save Workspace) does not work when the function editor is floated

We have decided that floating editors should support these keyboard shortcuts: Ctrl+Shift+S for Save Workspace, Ctrl+Shift+L for Load Workspace, Ctrl+Shift+C for Copy Workspace and Ctrl+Shift+R for Clear Current Workspace. Look for this in a future pre-production release.

Labels should not be listed in Open Object dialog

You are correct. Unlike APL+Win, when a function containing a label was suspended, the Open Object dialog showed the label as a variable. The same thing occurred with an inner-function but listed as a function. This has been corrected in the 2020.0.6 pre-production release.