Modifications Included in APL64 Pre-Production Version (2020.0.7)

Table of Contents

Overview

This document describes the enhancements and bug fixes incorporated into the seventh pre-production version of APL64 released in 2020.0.7. This document also describes the evaluator comments and APLNow responses based on the prior pre-production versions of APL64.

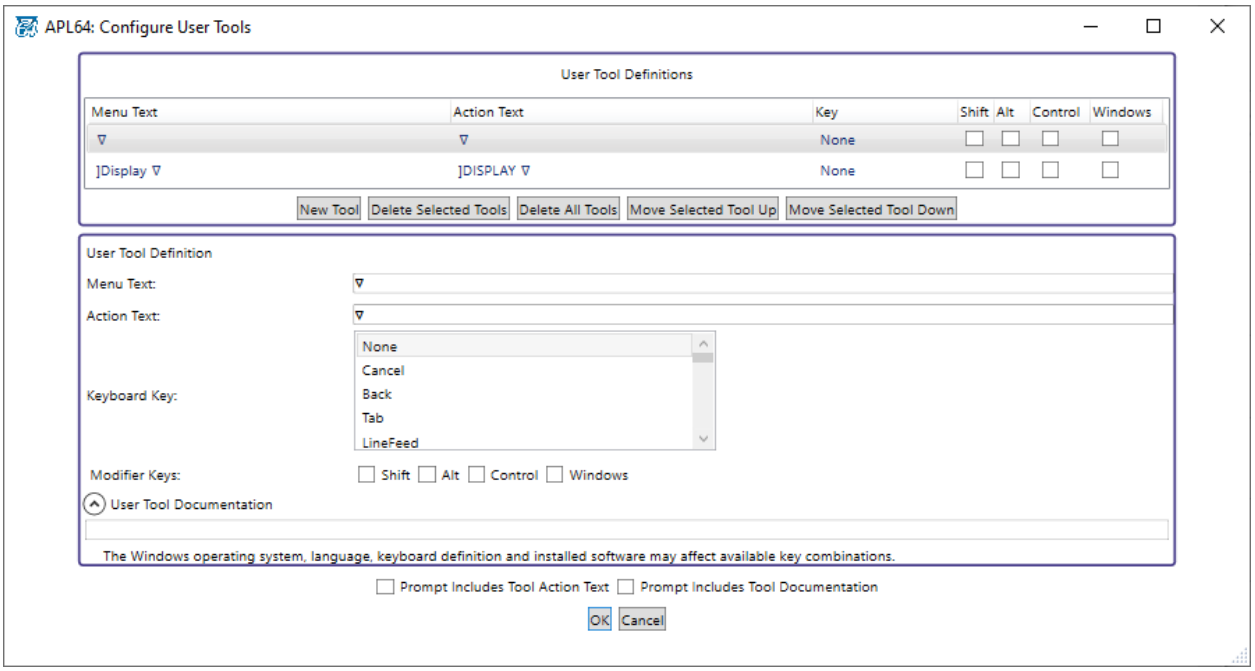
Enhancements

New ⎕XL Actions:

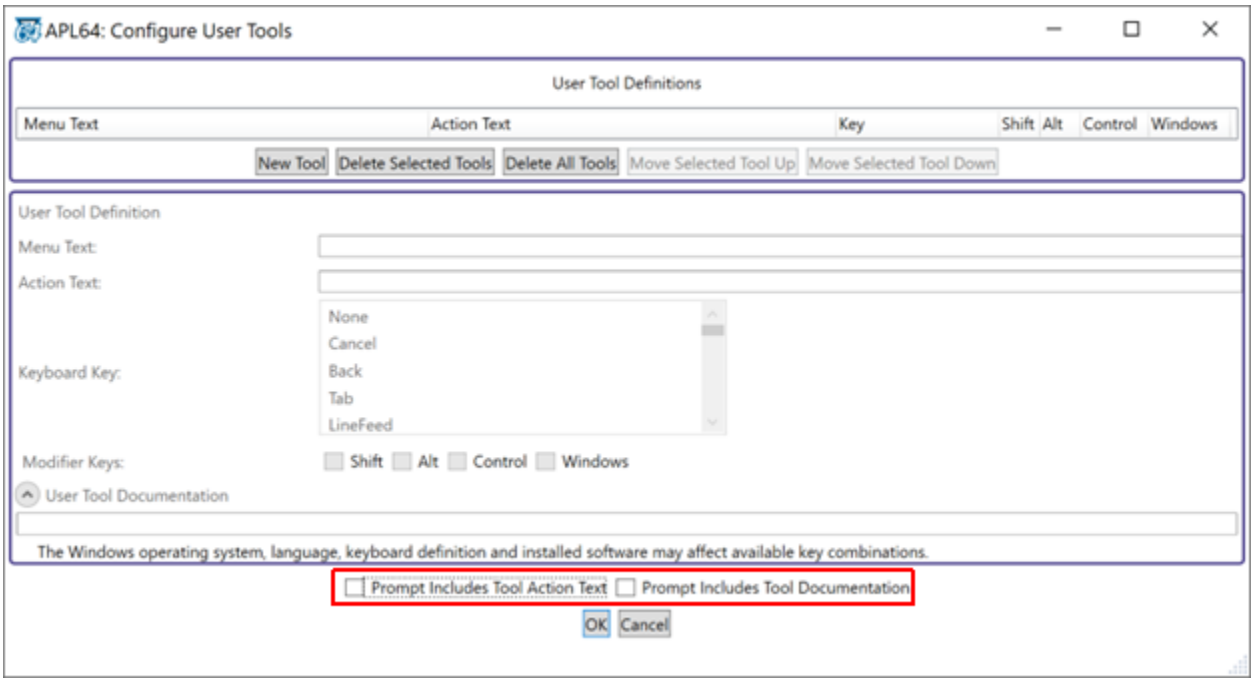
GetA1AddrFromR1C1Addr, GetA1ColNameFromColNum, GetR1C1AddrFromAbsA1Addr, GetUsedRange and GetWorksheetNames. Descriptions and examples for these actions are in the menu **Help | APL Language | Using ⎕XL**.

Options | Configure User Tools: Compact Prompt Option

The Configure User Tools dialog now provides to options to make the user tool argument input dialog (prompt) more compact: Prompt Includes Tool Action Text and Prompt Includes Tool Documentation.



New User Tools Definitions options: Prompt Includes Tool Action Text and Prompt Includes Tool Documentation



- When Prompt Includes Tool Action Text is checked, the user tool prompt will include the Action Text in the User Tool Definition.
- When Prompt Includes Tool Documentation is checked, the user tool prompt will include the optional user tool documentation defined in the User Tool Documentation.

The result for executed User Tools no longer includes the prefix string "**>[UserTool]**"

New menu **Session | Configuration Settings | Apply Default Settings**

This option allows APL64 to restart with all the default configuration settings.

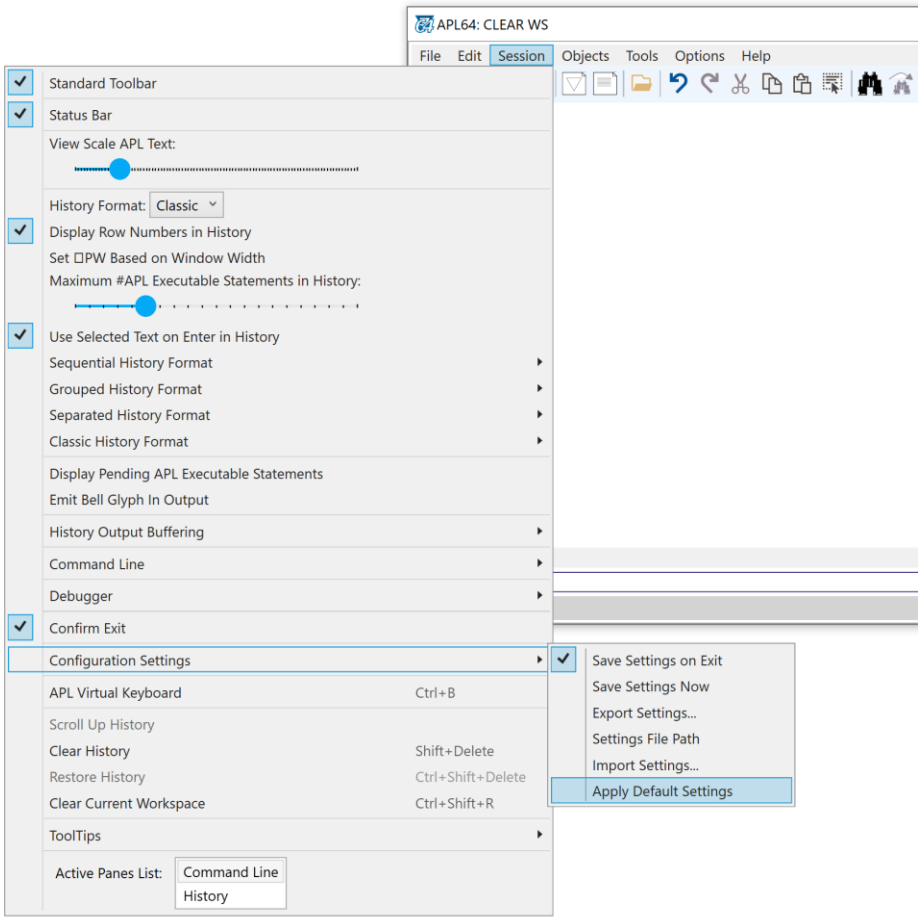


Fig: The menu location

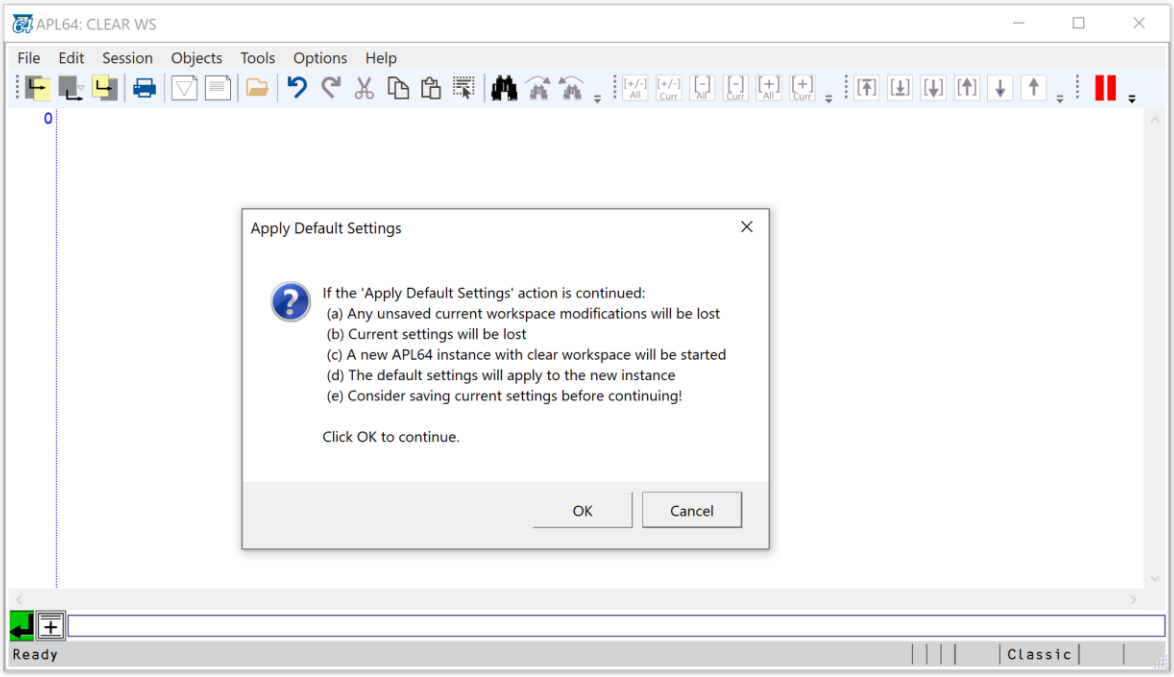
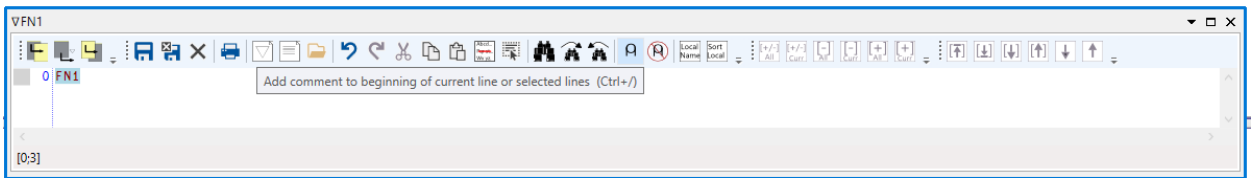


Fig: Warning message

Function Editor: Comment & Uncomment Toolbar Buttons



Menu: **Objects | Editors Pane Format | Open Non-modal Object Editor In Prior Location** is enabled by default

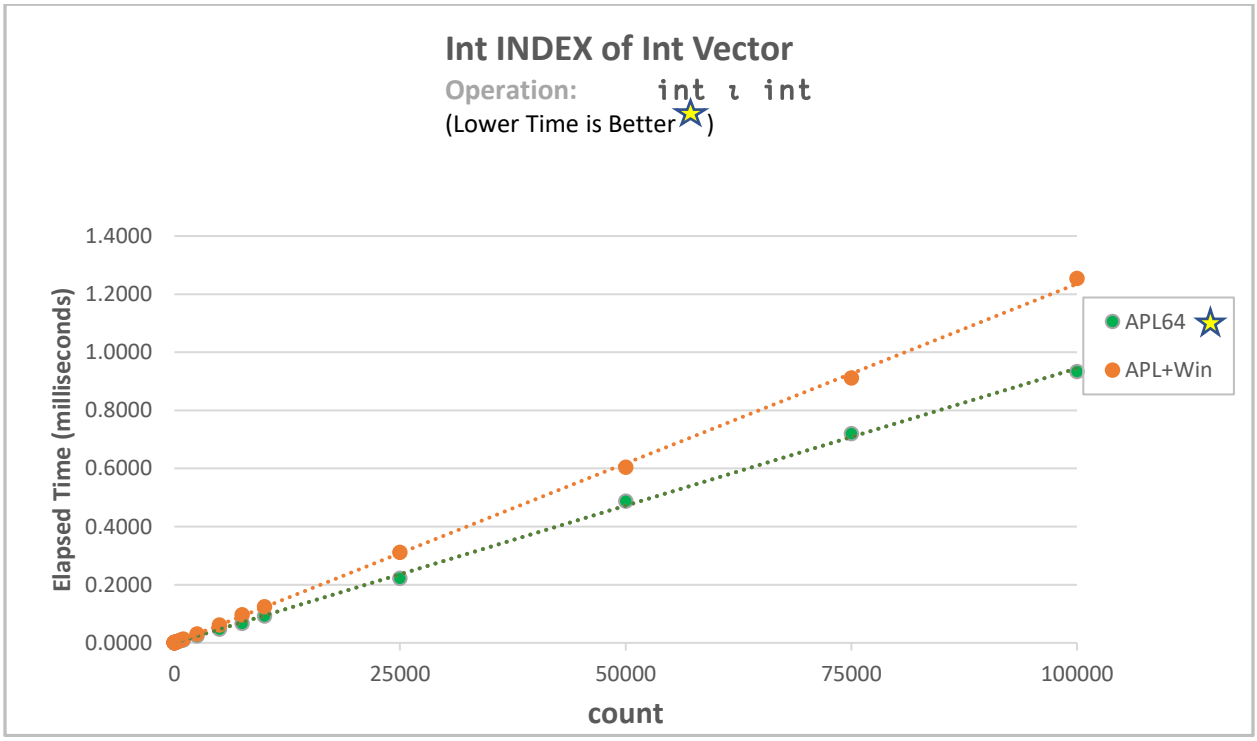
When this option is enabled, a non-modal function or variable editor will be opened in the same floating location previously used for this named object.

A marginal (50%) performance improvement in ⎕LJUST for Rank 2 character arrays

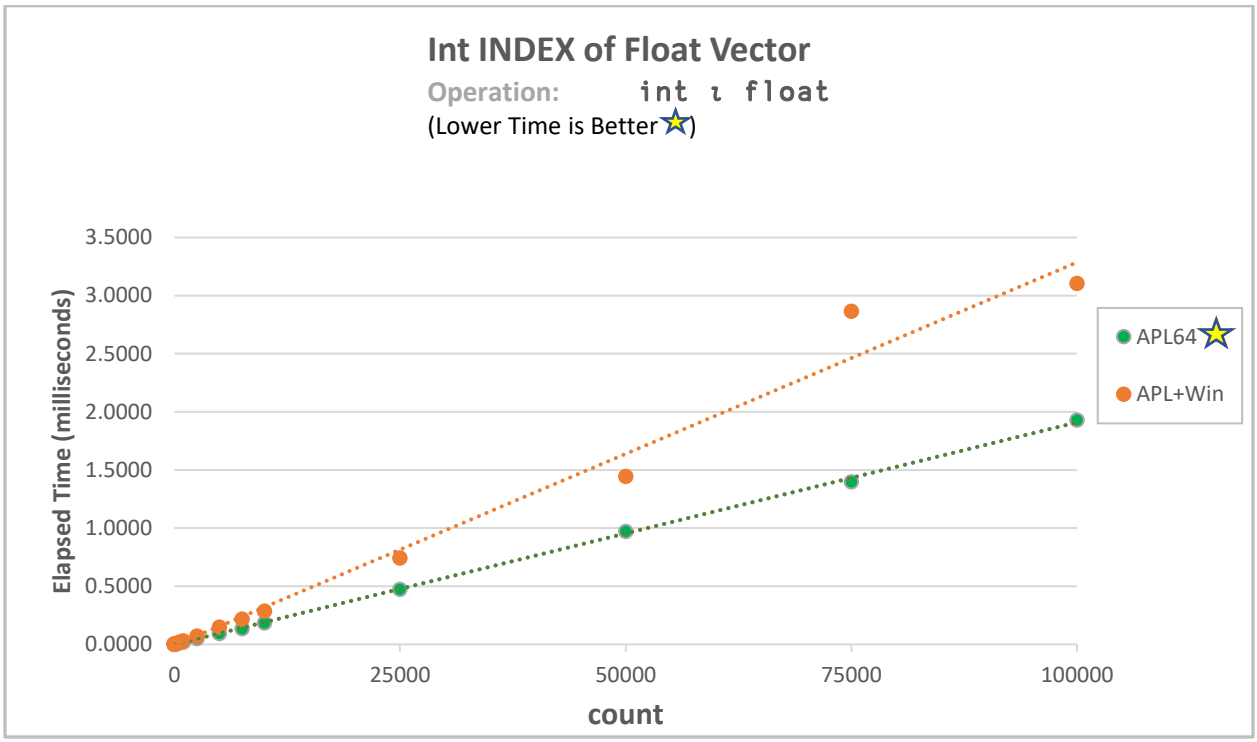
Performance Improvements

The graphs below illustrate performance improvements for ‘Index of’ and ‘Member of’ operations in APL64 compared to APL+Win 19.

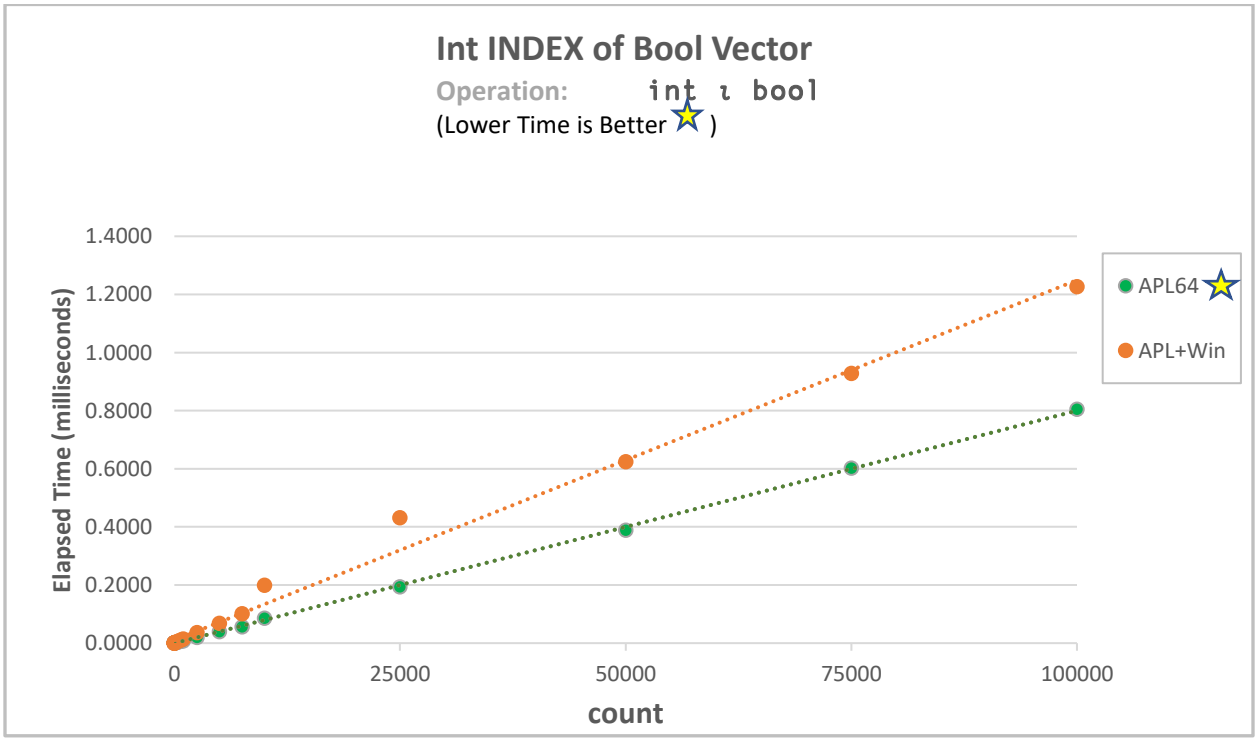
Integer INDEX of Integer Vector



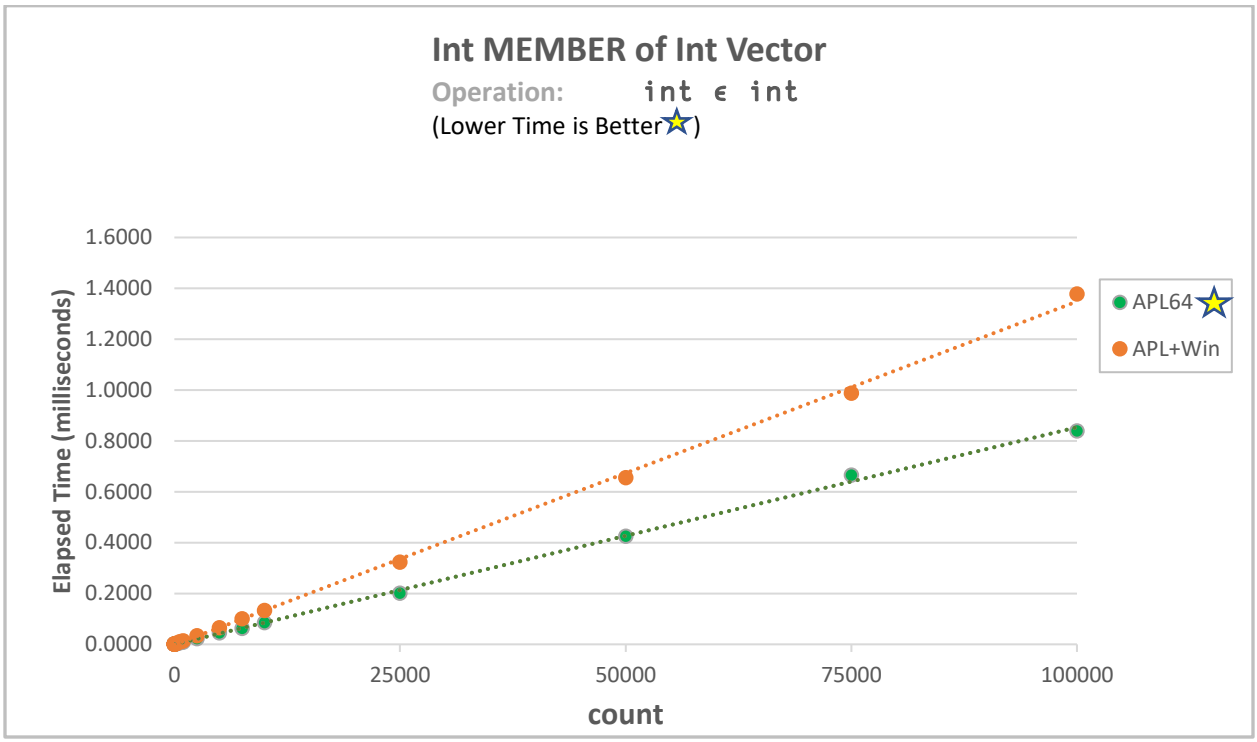
Integer INDEX of Float Vector



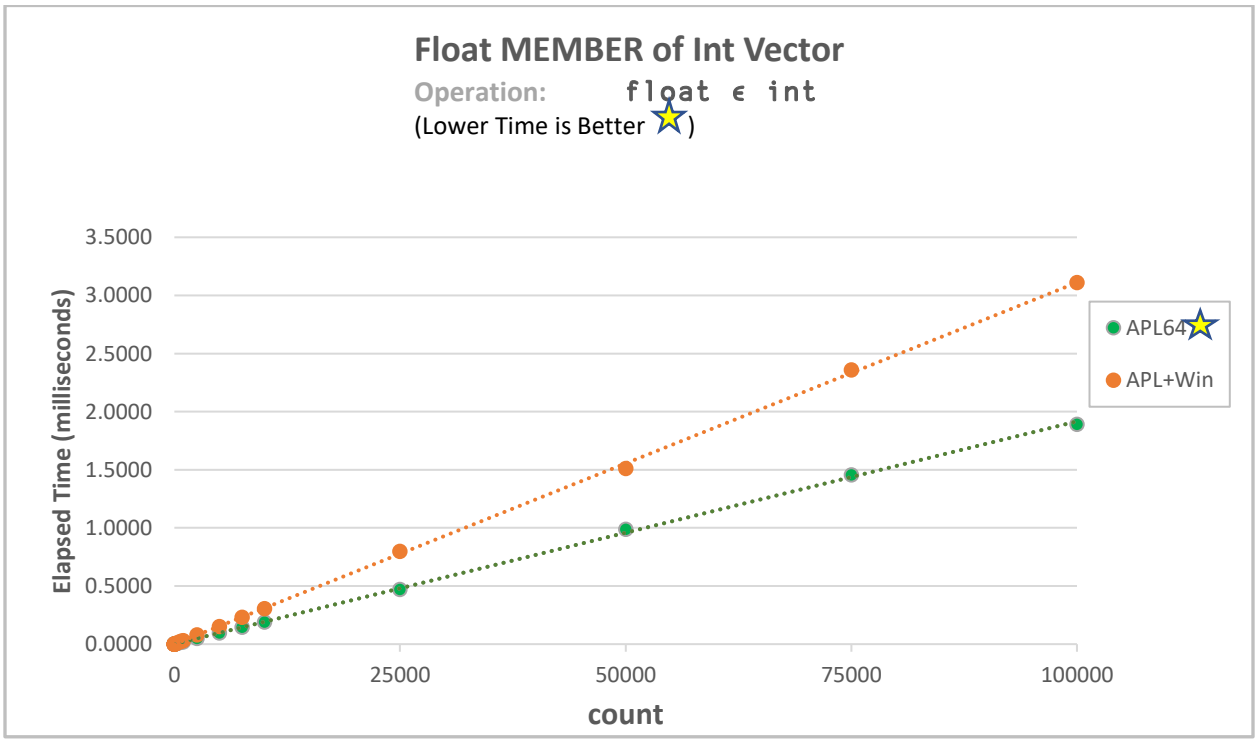
Integer INDEX of Bool Vector



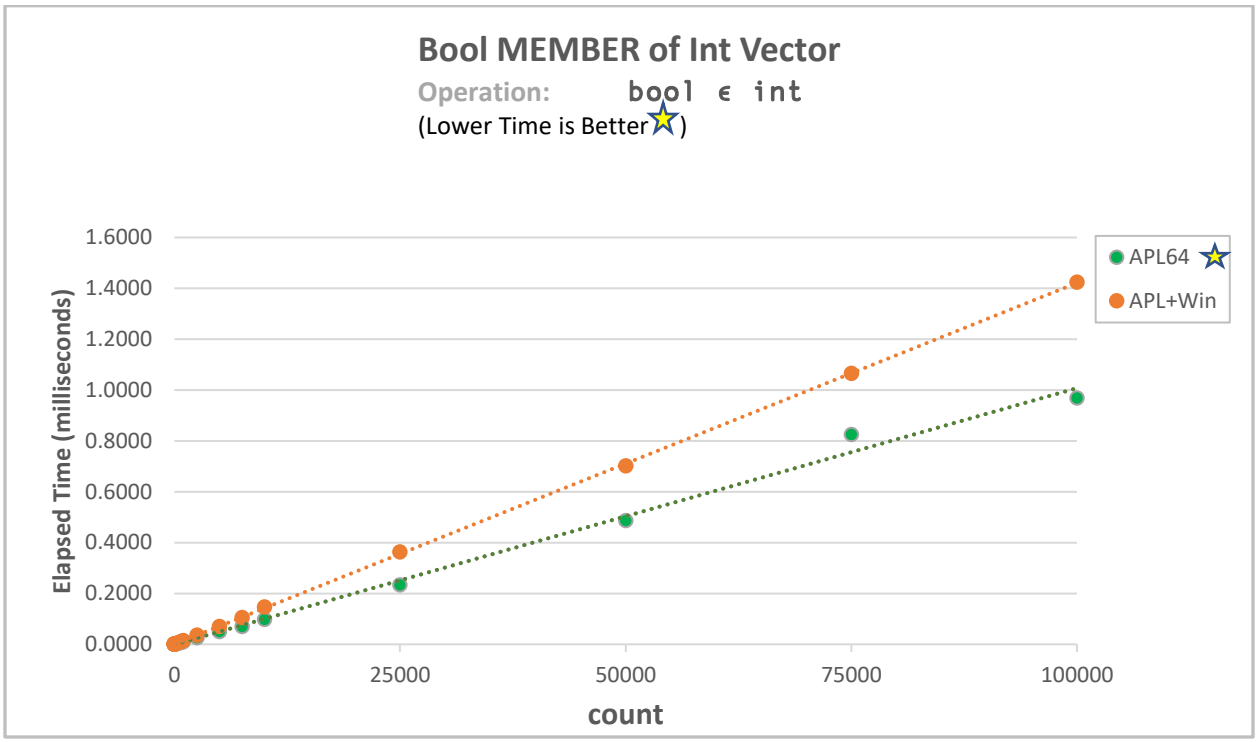
Integer MEMBER of Integer Vector



Float MEMBER of Integer Vector



Bool MEMBER of Integer Vector



Bug Fixes

The font size in the Debug pane was not affected by the Session | View Scale APL Text setting and the Ctrl+Mousewheel(up/down)

When in the Debug or State Indicator pane, typing an APL character did not switch to the command line/editable history

If the Debug/State Indicator panes are visible and the user puts the focus in the Debug/State Indicator panes and types a character which does not constitute a keyboard shortcut:

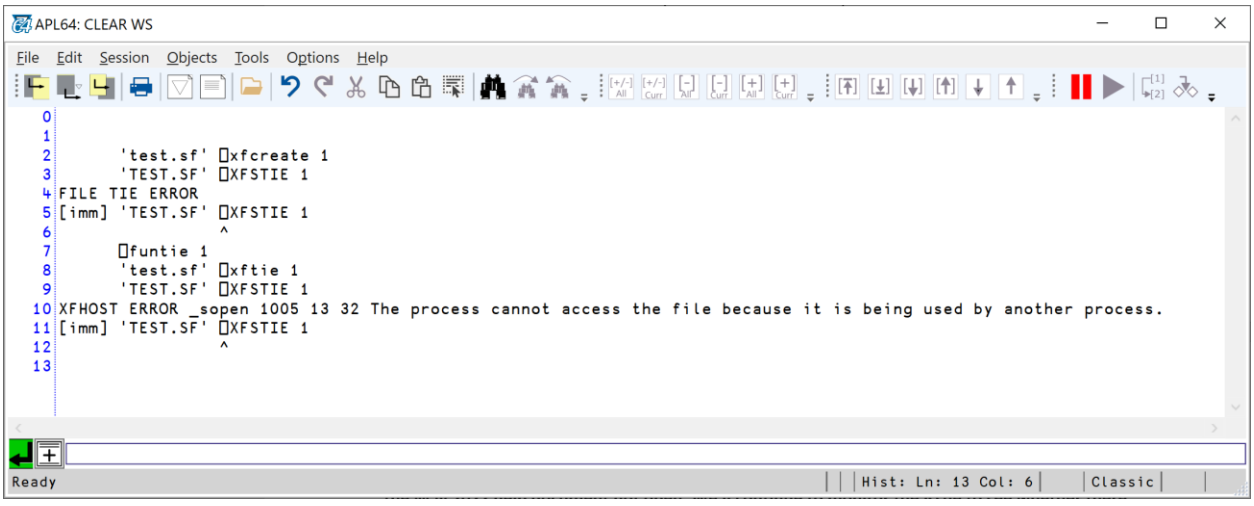
a. If the editable classic history format is used, the typed character will be appended to the text of the last line of the history.
b. If the editable classic history format is not used, the typed character will update the command line. The type of update to the command line (insert or replace) is controlled by the **Session | Command Line | Insert Text Into Command Line** option.

Help | License & Activation: #Days remaining

The number of days remaining is now consistent between the License Status dialog and the Refresh License dialog.

Tying a component file with different case letters would erroneously result in one of the following two error messages: FILE TIE ERROR or XFHOST ERROR _sopen 1005 13 32 The process cannot access the file because it is being used by another process.

Example:



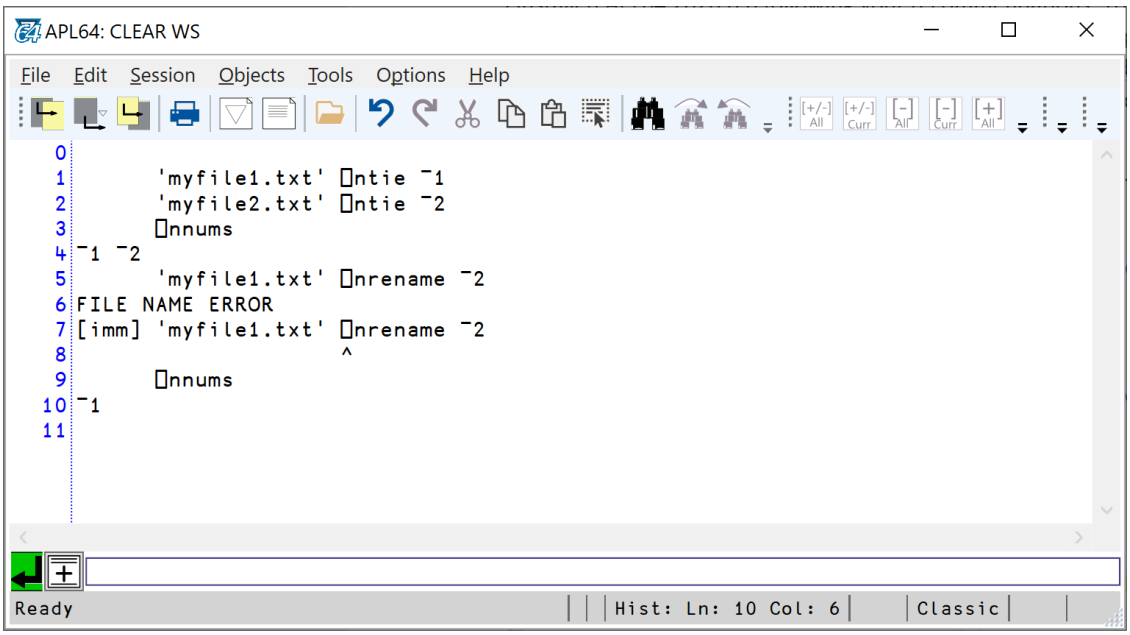
The User Tool Definitions dialog's Menu Text and Action Text edit fields did not select-all text when tabbed into

Invoking a User Tool with the del (∇) for the Action Text on an unexecuted line in the editable History could erase that line

A user command did not successfully execute in a User Tool

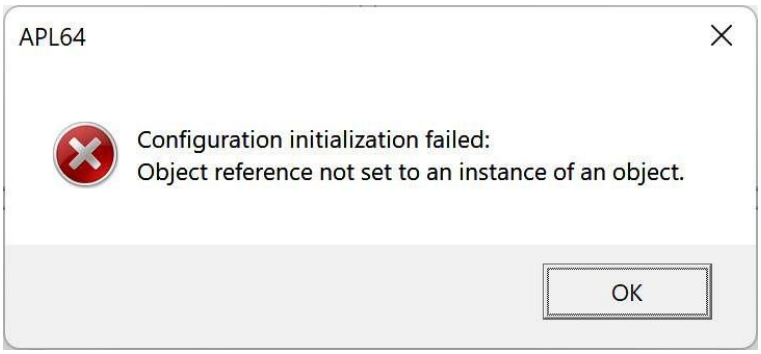
When ⎕nrename failed, the same tied file was then untied

Example:



Moving the GridSplitter Bar between the editor and debugger panes may result in the error message - Configuration initialization failed: Object reference not set to an instance of an object

Example:

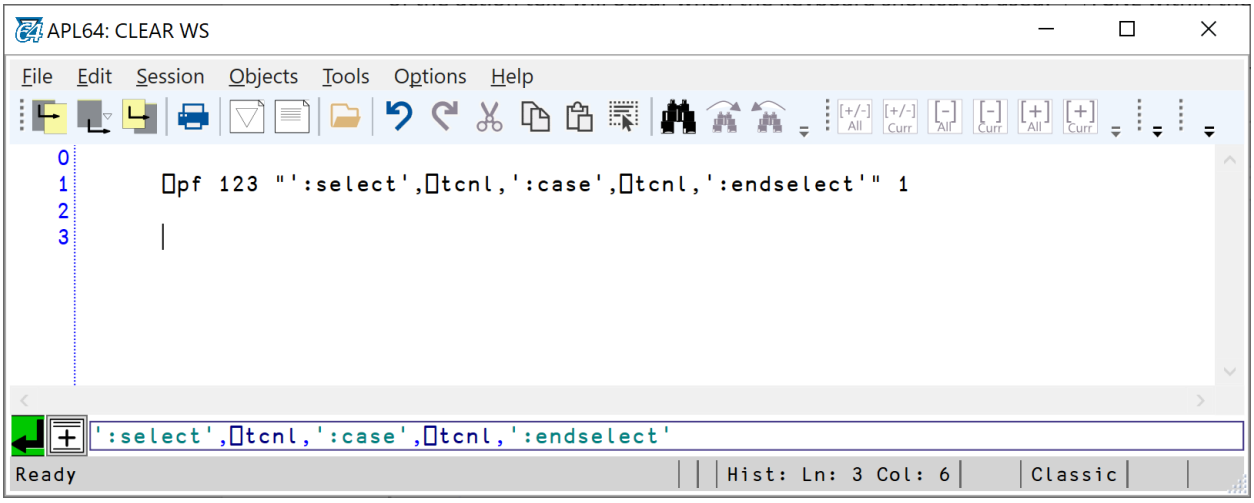


Laminate with left and right scalar arguments returned AXIS ERROR

⎕MATtoSS may result in extra blank spaces in the result

Executing the keyboard shortcut for a ⎕PF key binding did not reliably insert the associated APL statement in the editable History or the function editor

When pressing the F12 key, the APL statement appeared in the APL command line instead of the editable history.



Objects | Row Numbers | Display Row Numbers in Variable Editor

Using this CheckBox will now determine if row numbers will be presented in the variable editor.

Problem when pressing the down arrow key on a non-empty line in the editable history

With the **Session | Classic History Format | When At End of Editable History ↓ To Command Line** menu item unchecked, pressing the down arrow key on a non-empty line incorrectly placed the caret at the beginning of a new line in the history. To replicate APL+Win behavior, pressing the down arrow key now adds a new line with the caret indented six-spaces if the current line is non-empty.

The dyadic iota operation failed for negative integer values

Instead of 5⌶711 and 6⌶711 returning DOMAIN ERROR, APL64 ran in a recursive loop

The leading ⎕ (quad), Δ (delta) and ⍙ (delta underscore) glyphs were not treated as part of a token for the Ctrl+left and right arow shortcut keys

The leading ⎕ (quad), Δ (delta) and ⍙ (delta underscore) glyphs were not treated as part of a token for the mouse double-click selection

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 (APL64_2020.0.6.13405). We would like to thank the evaluators for their contribution.

"Help | APL64 Project History | May 2022" didn't display anything (initially)

I installed APL64 2020.0.6 following your recommendations. You are probably already aware of that, but selecting Help | APL64 Project History | May 2022 does not display anything. More exactly it did not display anything until and I tried multiple times until I displayed another Help file from another menu. After that selecting Help | APL64 Project History | May 2022 did work.

No. This isn't something we're aware of. This has been tested on several machines and not once did the May 2022 help document not open. We'll continue to monitor the issue to see whether there are other reports like it.

APL64 didn't start on Windows Server 2016 Data Center

The .NET Desktop Runtime 5.0.15 was installed with APL64. This was corrected by installing the .NET Desktop Runtime 5.0.16.

Moving the debugger splitter may result in: Object reference not set to an instance of an object

This is a known issue that we anticipate will be corrected in the next pre-production release (ver 2020.0.7).

What is considered to be under the caret?

i. For Ctrl + C, it is the whole line.
ii. For Ctrl + Shift + O, it is the word delimited by space at either end.
iii. For both of the above, the line or word does NOT have to be selected/highlighted.

The comment to **i.** above is true only when the menu **Edit | Copy/Cut Entire Line Without Selections** is checked.

What is considered to be under the caret for ∇ (del) in the context of user defined tools?

i. Is it the word with space at either end? What if that word includes punctuation such as : and \ in a fully qualified filename?
ii. Does the word always need to be selected/highlighted?

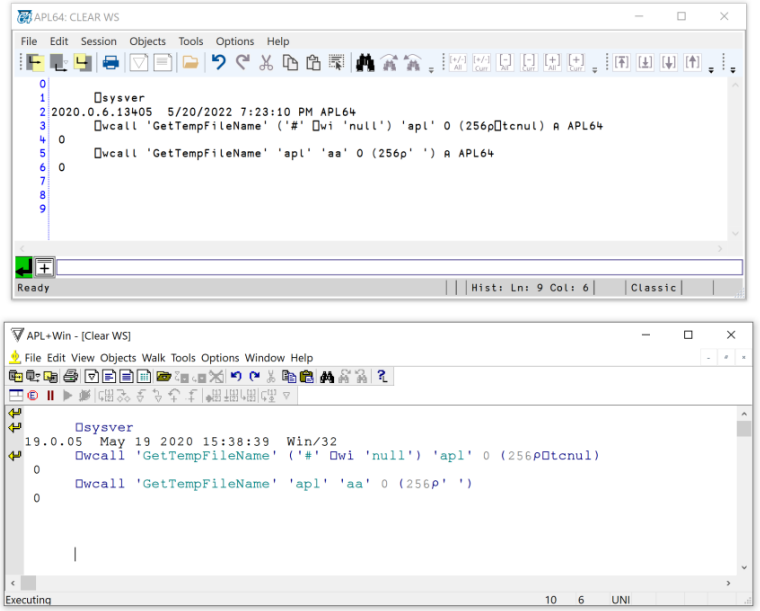
Per the documentation, the del is replaced with any text that is highlighted in the current window or with the object that is closest to the cursor in the current window if you did not highlight anything before invoking the tool.

Is the word read as ∇ (del) cached? That is, is it ever re-used internally?

No.

The APL64 GetTempFileName API call is inconsistent with its APL+Win counterpart

The image below demonstrated that we've been unable to corroborate your assertion that APL64 and APL+Win handle this API call differently for a non-existing path, nor is this expected since APL64 and APL+Win utilize a similar APL engine when executing ⎕wcall commands.

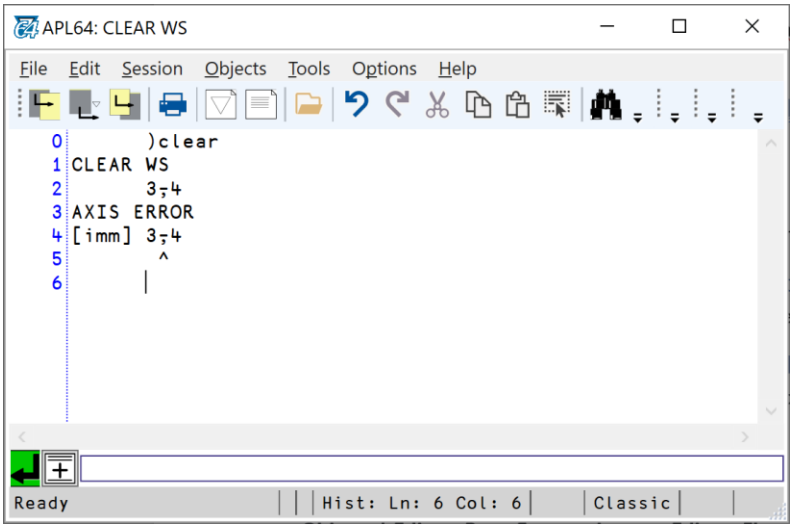


FYI. These results were repeatable in Windows 10 and Windows 11.

If you can find out what's responsible for the GetTempFileName API behaving differently in APL64 and APL+Win, we'd like to hear about it.

The following instruction: scalar⍪scalar yields AXIS ERROR

Example:



This issue will be corrected in a future pre-production release (ver: 2020.0.7).

Region markers [in function editor] are sometimes wrong

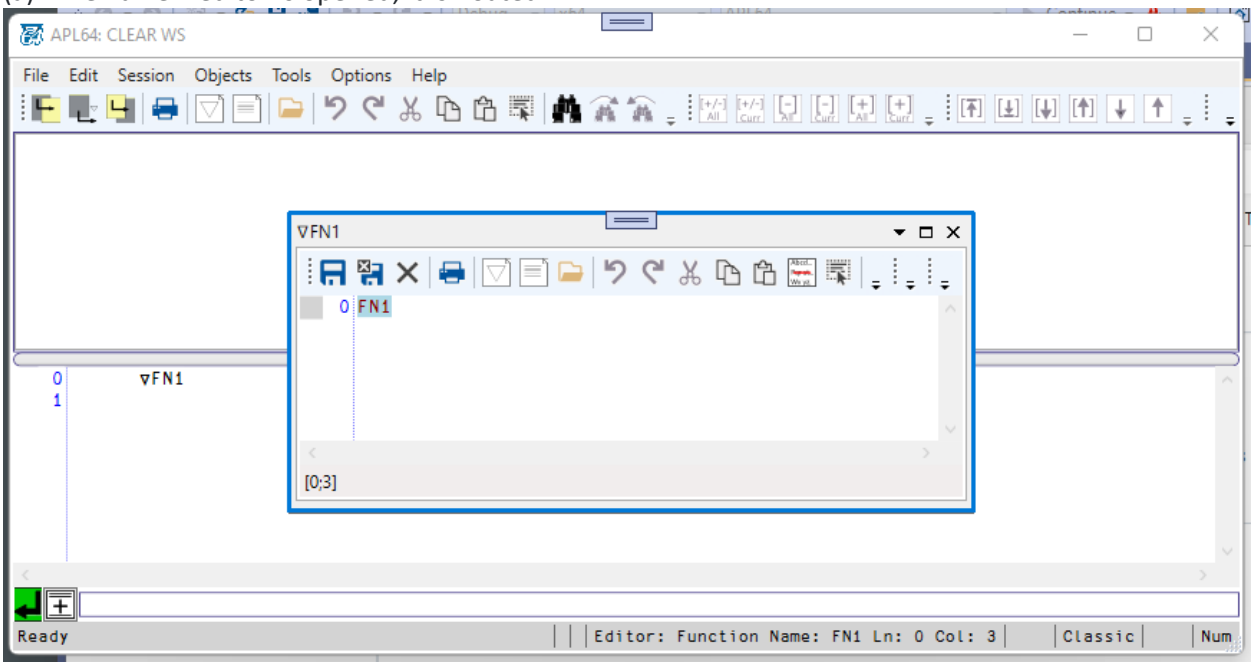
We will investigate the issue and correct it in a future release.

It is not possible to dock a floated function Editor if option Editors Floated is used

We cannot duplicate your scenario. So, we may need your assistance if you have a reproducible scenario.

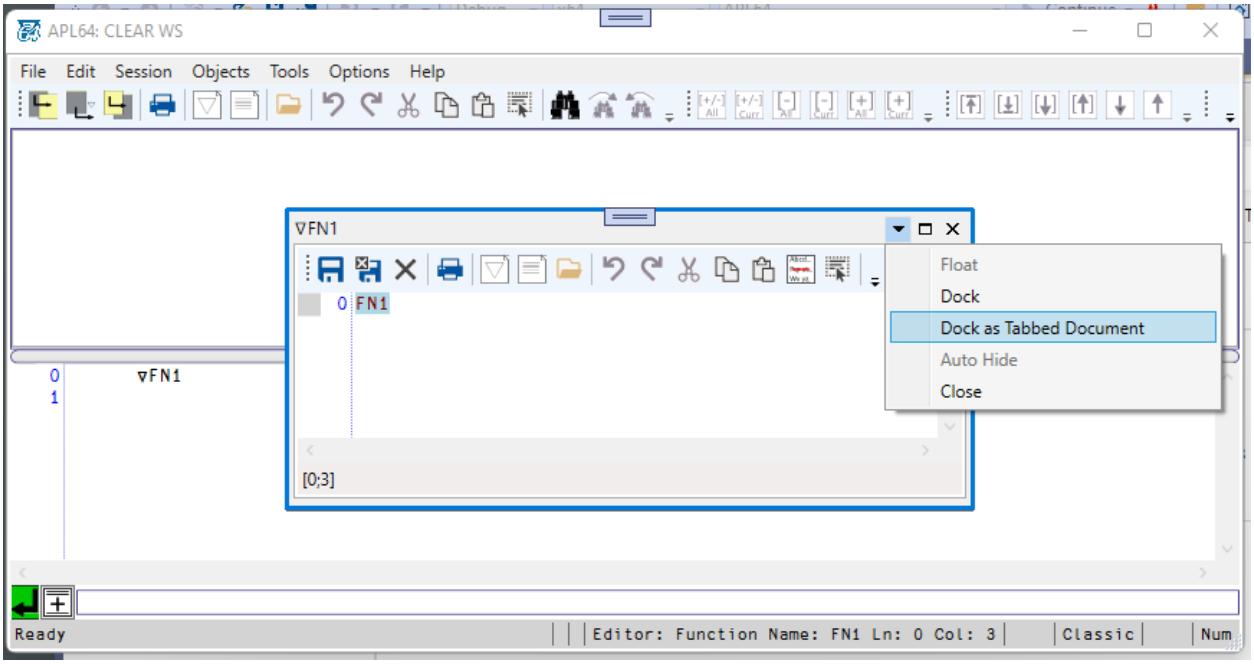
The following is what we observed:

Objects | Editors Pane Format: Layout: Editors Floated is user selected:
(a) When a new editor is opened, it is floated:

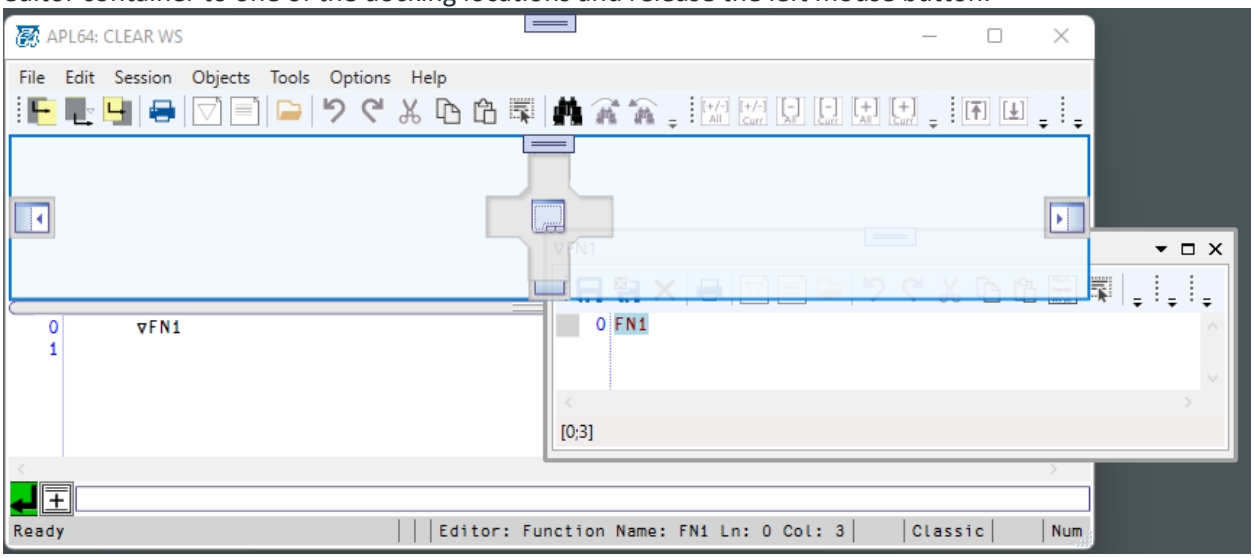


(b) The floating editor may be docked in two ways:

(1) Right click the list immediately to the left of the Full Screen and Close icons in the header of the editor container and select Dock or Dock as Tabbed Document:



(2) Put the mouse pointer on the editor container, depress and hold the left mouse button, drag the editor container to one of the docking locations and release the left mouse button:



Clicking **Objects | Dock Floating Function & Variable Editor Panes** will dock all floating editors.

If you have a reproducible scenario, please send it to us.

The down key at bottom of the APL Session should not create new session lines

In APL64 this behavior is by design. The Ctrl+End is the appropriate keyboard shortcut to move the caret to the bottom of the editable classic history. APL+Win is 'wasting' the Key.Down to do the same thing as Ctrl+End.

It is also true that Enter/Return adds an addition empty line to the history when the caret is at the bottom of the editable classic history. That is why the default APL64 implementation of Key.Down when the caret is at the bottom of the editable classic history moved the focus to the command line, rather than duplicate the Enter/Return key action.

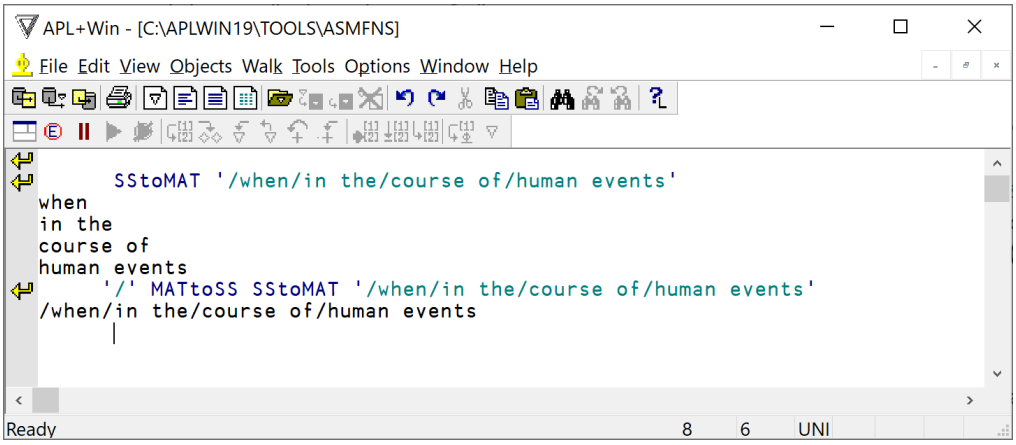
It is recommended that **Session | Classic History Format | When At End Of Editable History ↓ To Command Line** be checked by an APL64 user so Enter/Return, Ctrl+End and Key.Down have distinct actions in the editable classic history pane.

In retrospect, implementing your suggestion to provide the **Session | Classic History Format | When At End Of Editable History ↓ To Command Line** as a checkable option may have been a mistake.

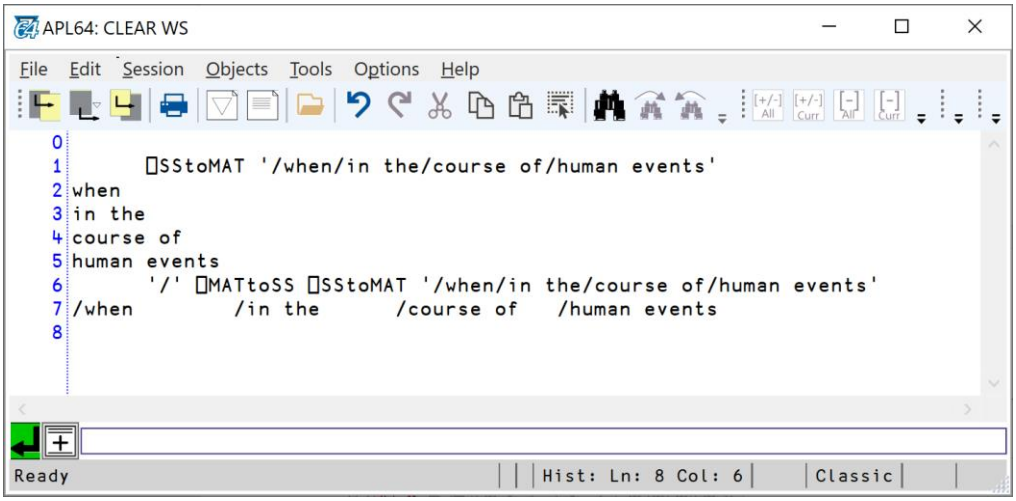
☐MATtoSS result different from MATtoSS in ASMFNS workspace in APL+Win

Example:

APL+Win:



APL64:



The extra spaces inserted by APL64 are going to cause our code to break!

This issue will be investigated, and the correction will be available in the next pre-production release (ver 2020.0.7).

Open Non-modal Object Editor in Prior Location does not remember a maximized editor

Currently the GUI toolkit for Floating/Docked panes does not provide (under program control) a way to determine if a floating pane is maximized or to maximize a floating pane. Only the user can (manually) maximize a floating pane. We will request an enhancement from the vendor of this toolkit and if they implement the enhancement, it will be added to the next available version of APL64.

Update: Corrected in the 2020.0.7 pre-production release.

Isn't a string just a character vector?

No. A string is a scalar and an array of strings has elements which are strings. A character is a scalar and a character array has elements which are characters.

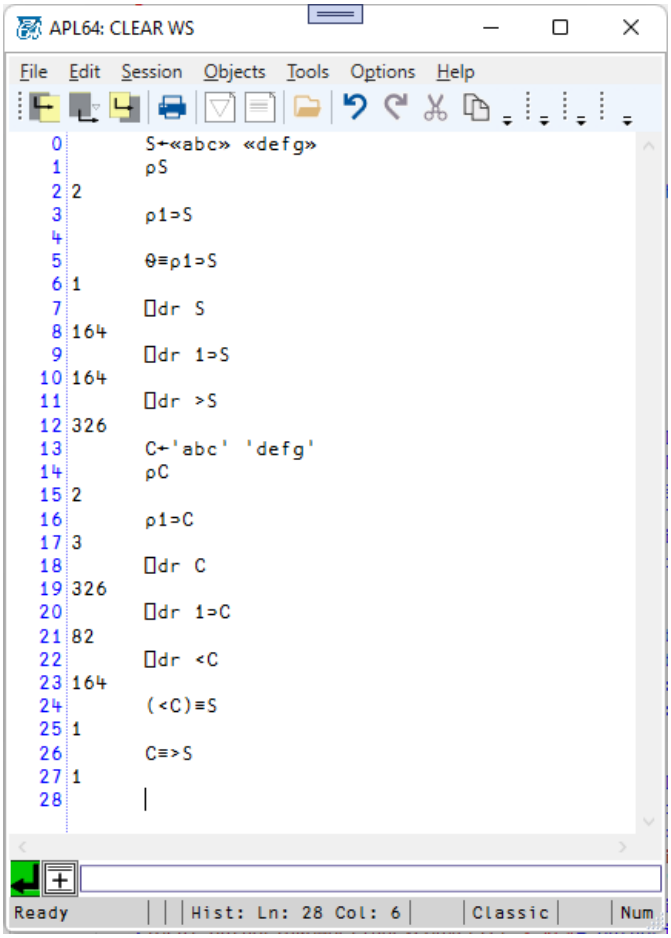
I have difficulty in seeing the purpose [of strings]. Why would APL need another data type for character vectors?

One of the keys to keeping APL a 'living' language is to enable it to interact with non-APL tools and environments. Many APL programmers want to take advantage of features not present in APL. This is important for APL applications which 'thrive' in an environment larger than APL. Often this means using .Net technology. In .Net both string and character are available datatypes, but string manipulation is overwhelmingly more common and better developed.

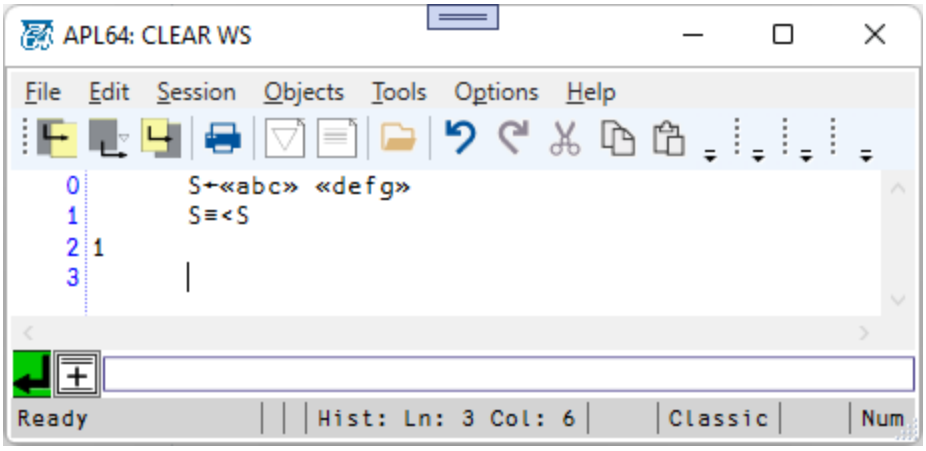
Check out the new APL64 ⎕STRING system function to see a robust implementation of .Net string features in APL64. In APL64 execute ⎕string '?' to see a summary.

The document says a string can be treated as a scalar but so can a character vector if one encloses it.

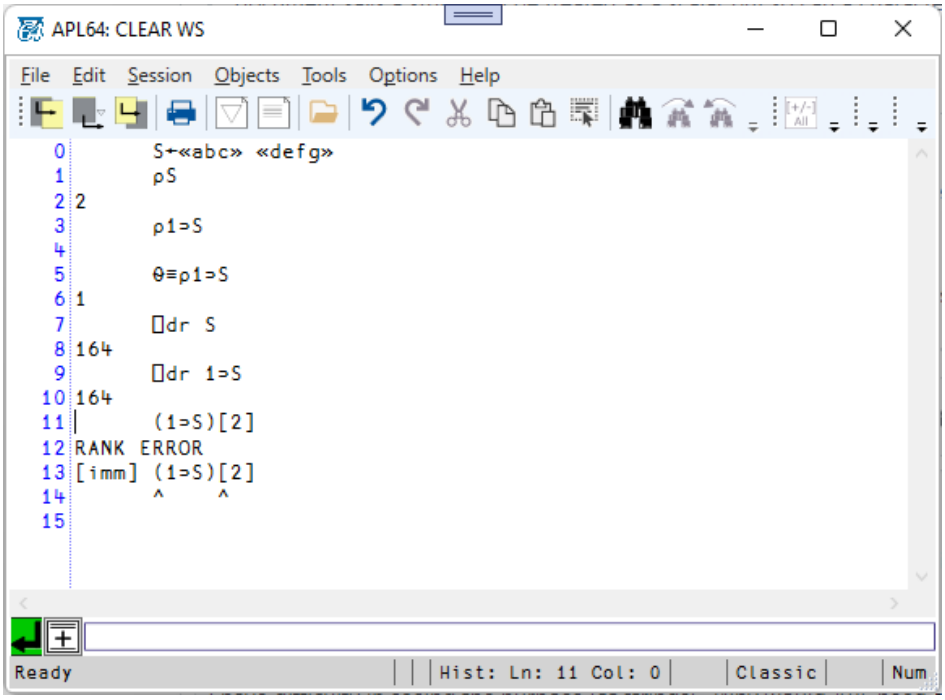
No. The operation to convert a string to a character array is destring (>String). The operation to convert a character vector to a string is Enstring (<CharVec). Destring and Enstring are distinct operations from Disclose and Enclose.
Example:



The repeated use of the Enstring operation has no effect:



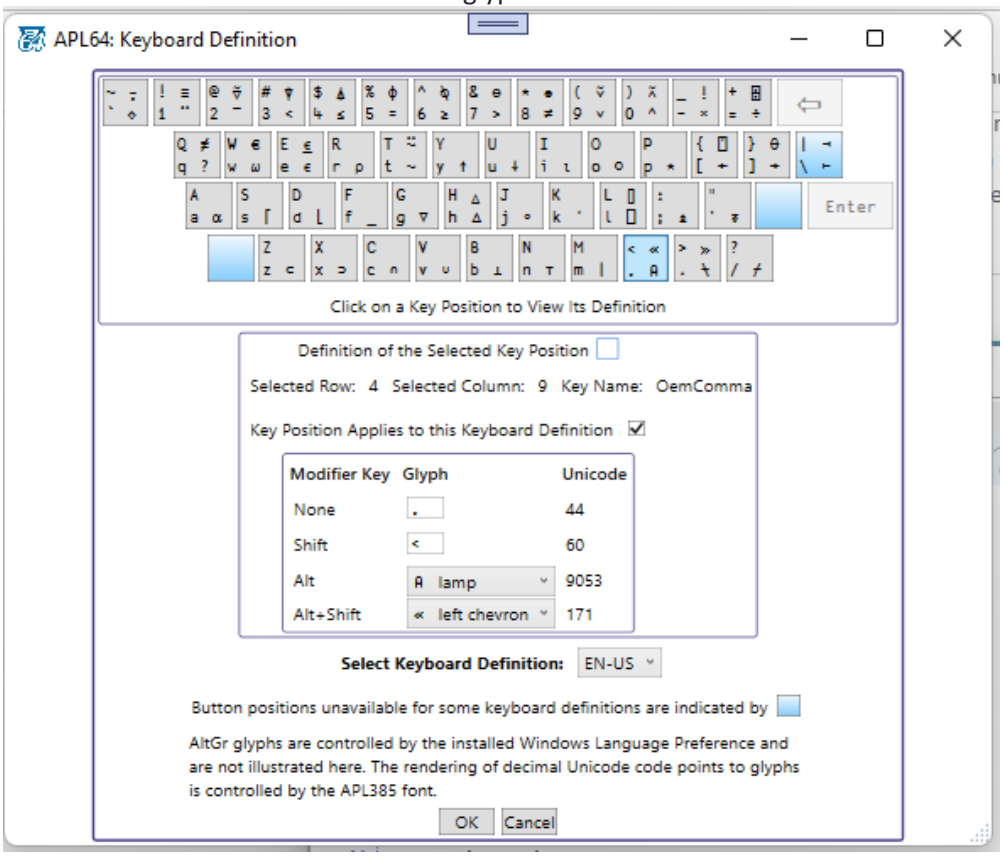
APL64 properly throws an exception:



Some APL64 system functions accept arguments that are string and string arrays in addition to characters and character arrays.

The delimiter glyphs for string in APL64 are the left and right chevrons, whereas the delimiter glyphs for character vectors in APL64 and APL+Win are quotation marks or quote mark. Use **Options| Keyboard**

Definition to see the location of these glyphs:



The ⎕XL function is intriguing, but how can it exchange data with an Excel worksheet if Excel is not installed?

⎕XL can only exchange data between an Excel-format .xls or .xlsx file.

A mechanism in ⎕XL to inquire as to the names of the worksheets and to the size (rows and cols) in a worksheet

Your enhancement request to ⎕XL was approved. Look for these two new ⎕XL action arguments: GetWorksheetNames and GetUsedRange, in the next APL64 pre-production release (ver 2020.0.7).

What one can do (from within APL) with Excel installed that one cannot do without Excel installed?

If APL and Excel are installed on the same Windows machine, it is possible to use Excel to the fullest extent (including run a macro) via the APL ⎕WI ActiveX interface.

Can one cause an Excel macro in a worksheet to execute without Excel being installed?

No.

Is there a replacement for the GetDOSHandle function from the UTILITY workspace that employs ⎕CALL?...

…The results of GetDOSHandle are being used in the GetFileTime function to get the file time in that one function

We will investigate whether it will be possible to update the GetDOSHandle function for APL64. We'll be in touch with an update when we've completed our investigation.

After some internal discussion on this issue, the conclusion was to recommend you to substitute your current solution with the ⎕NFE system function with the methods GetCreationTime, GetLastAccessTime and GetLastWriteTime.

Example:

⎕nfe 'GetCreationTime' file
⎕nfe 'GetLastAccessTime' file
⎕nfe 'GetLastWriteTime' file

There are also UTC versions of these methods, if they're necessary. For more information, refer to Chapter 8: Native Files with Encoding in the System Functions Manual.

Is there a replacement for the catbar function that employs ⎕CALL?

The user followed up with “Further testing in both APL+Win and APL64, which leads me to conclude that the catbar function is identical to ⎕OVER in APL64”.

Will the interpreters require separate activation or will the activation be unified?

A separate activation is required for both APL+Win and APL64 and they will coexist comfortably.

When I registered this new release, it told me I had 10 days left. But when it went back to APL it told me that it would expire in 8 days. Why the discrepancy?

The discrepancy was in the number of days reported in the separate message box following the initial activation. This will be corrected in the next pre-production release (ver 2020.0.7).

I think there is a blocking problem with ⎕xftie in APL64 (this problem does not happen in APL+Win)

The problem was due to file names being compared in a case-sensitive manner. This will be addressed in the next pre-production release (ver 2020.0.7).

I do not see ⎕PR in the System Functions help file for APL64

⎕PR is a system variable therefore it is described in the System Variables help file for APL64.

I defined PF12 with a code snippet and wanted to get the text replicated in the editor. It always replicates to the session window even when I have the focus in the editor window; e.g. ⎕pf 123 "':select',⎕tcnl,':case',⎕tcnl,':endselect'" 0

Per the documentation, you must replace the 0 with a 1 at the end of your APL statement to get the result you're after.

Problem with dyadic Iota in APL64 where all of the elements in the right arg that are less than zero are not found in the left arg; but are found in APL+Win

This will be investigated and corrected in a future release.

APL64 seems to be taking a lot more memory to store and process nested arrays than APL+Win did

For us to analyze your scenario, please provide a simplified workspace which illustrates the issue and the "System information.pdf" document generated from the option **System Information** in the menu **APL64 | Help | About APL64**.

⎕chdir ⍬ works with APL+Win but yields DOMAIN ERROR with APL64

This will be investigated and corrected in a future release.

APL64 APLW.WSEngine?

Your question is answered in more detail in the second **boldened** bullet appearing in the APL64 menu **Help | APL+Win Compatibility | Compatibility Modification for Existing APL+Win Applications**:

• APL64 includes the APLNow32.exe component which supports certain legacy features of APL+Win in APL64. The APLNow32.exe component does not require a separate installation or ActiveX registration on the target workstation. The configuration of APLNow32.exe is like APL+Win and its configuration is preserved between APL64 instances in an ini-format configuration file. The APLNow32.exe component is applicable only to the Windows operating system environment. Certain features of APL64 are available only on the Microsoft Windows operating system platform:

- **o The ⎕WI GUI development features, e.g., 'Fm' ⎕WI 'Create' 'Form', ⎕WCALL, etc.**
- **o Microsoft has deprecated ActiveX technology in the cross-platform .Net Core, so an APL64 ActiveX server engine does not exist. However, APL64 may operate as a client and use the APL+Win ActiveX server engine via ⎕WI. Note that the APL+Win ActiveX server engine can access only APL+Win format workspaces and not APL64 format workspaces.**

Problem with dyadic iota where in APL+Win all of the elements in movedata[;2] are found in codes[;1], but not in APL64 for elements in movedata[;2] that are less than 0

This will be investigated and corrected in a future release (ver: 2020.0.7).

5○711 and 6○711 both hang APL64

The issue will be addressed in a future release of APL64 (2020.0.7).

Ctrl+left (or right) arrow key does not treat ⎕ (quad) and Δ (delta) as part of a token

You are correct. A request has been made to the vendor of the text editor GUI control used in APL64, which replaces the Microsoft-deprecated GUI tools on which APL+Win is based, to provide a configurable definition of 'word' in the text editor. We will incorporate a modification into APL64 should such a modification become available from the vendor of the text editor GUI control.

Update: Corrected in the 2020.0.7 pre-production release.

I've encountered an array with ⎕DR 162 (for instance, the result of ⎕EDITINFO 'GetFnText'). I can edit it, but the type remains 162 after saving

Please read **Help | APL64 Project History | September 2018**, Page 2 - New String Datatype and **Help | APL Language | System Functions**, Page 229 - ⎕String system function

Note: The String datatype is more amenable to interfacing with .Net. The APL character singleton or array datatypes are retained in APL64 where it has been historically used. Some APL64 system functions support String datatype arguments in addition to other datatype arguments.

"Edit | Enable Unicode Clipboard" setting not available

In APL64, the Windows Clipboard is always Unicode enabled.

⎕WI '?property' is not available

The issue of ⎕WI '?property' not available in APL64 will be addressed in or before the production release. In the interim, you can place a copy of HELPSTRINGS.TXT from APL+Win in the same application folder where APL64.exe and APLNow32.exe reside. The application folder can be retrieved in APL64 with the command ⎕exepath.

⎕IT 'WsPack' is not available

⎕IT 'WsPack' is available in APL64. Did you try it?

When the root user command function is invoked, the leading] has been removed from the argument

Please provide a specific scenario which can duplicate the issue.

If an error occurs during a callback on an open form, focus does not go to the session manager; the form just locks up

Please provide a specific scenario which can duplicate the issue.

F6 does not move focus between session/debugger/stack

The F6 and Shift+F6 keyboard shortcuts in APL64 for navigating between the the session window, a code window that shows a function, and a state indicator window has been deprecated in APL64. Use Ctrl+Tab and Ctrl+Shift+Tab instead which is the analogous and built-in functionality when docking/floating panes are used.

Several items concerning Colors in APL64

e.g.,

- There is no color setting for "User Function"; setting for "Undefined Name" is used.
- There is no option to use session background color, all elements must have background color set individually, which is tedious. The converter should have done this automatically
- "Function Session" is listed in the Syntactic Elements rather than the General Elements
- Label color is only applied to the label itself; references to the label get "Undefined Name" color
- Setting Session Input and Session Output colors has no effect
- Somehow, the selection highlight colors got changed to black on white, so I can't see what's selected in the history. I think I remember seeing an option to change it, but can't find it now.

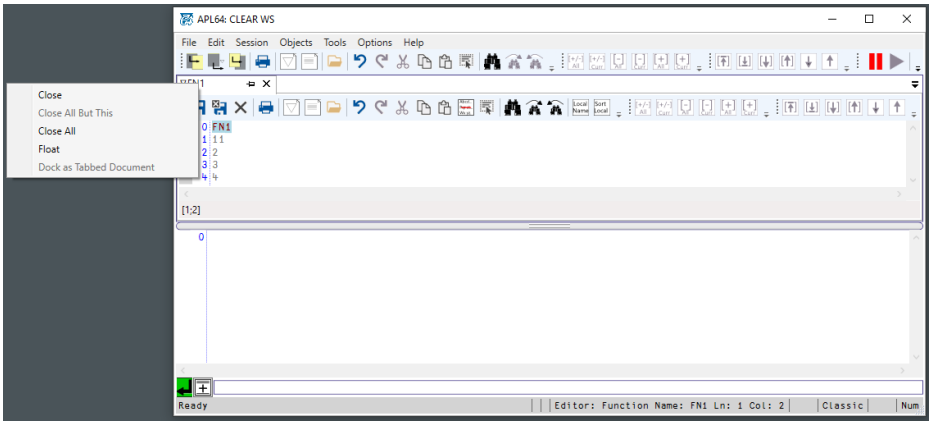
- The "Local Function" color setting seems pointless; the interpreter has no way of determining whether a local identifier is a function, and if a function is stopped after the local function is defined, the "Local Function" color is not used in the debugger

We will investigate each of the items you've raised in this issue then make a decision on if and/or how they will be addressed in a future release of APL64.

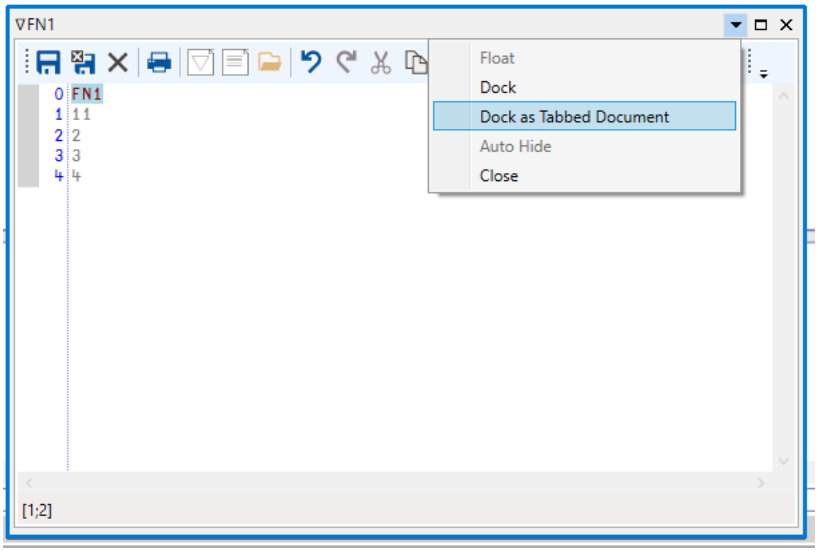
There should be a shortcut to float/unfloat a session. Also, "Float" should be an item in the context (right-click) menu in editor

The option to dock/float an existing editor pane is inherent in the pane, because docking/floating applies to the pane and not the pane's content.

To float an existing docked editor pane, right mouse click the docked editor pane's tab and select Float.



To dock an existing floating editor pane, right mouse click the floating editor pane list of actions (to the immediate left of the maximized and close buttons. There are subtle differences in the Dock and Dock as Tabbed Document options associated with parentage of editor panes. Some user experimentation is recommended.



All existing editor panes may be docked using **Objects | Dock Floating Function & Variable Editor Panes**.

All existing and subsequent editor panes may be floated using **Objects | Editors Pane Format | Layout: Editors Floated**.

Ctrl+T does nothing if most recent function session was closed

This behavior is the same as APL+Win. The Ctrl+T keyboard shortcut moves the focus only if there is an active editor. There can be only one active editor or only one GUI object with keyboard focus. If you've observed something different, please send the steps to reproduce this to support@apl2000.com.

Ctrl+F12 does not put the cursor at the point of suspension

This issue will be investigated, and any correction will be available in a future pre-production release.

When in debugger or stack panes, typing an APL character does not switch to the command line

A modification will be made to a future version of APL64 so that if the Debug/SI panes are visible and the user puts the focus on the Debug/SI panes and types a character which does not constitute a keyboard shortcut:

c. If the editable classic history format is used, the typed character will be appended to the text of the last line of the history.
d. If the editable classic history format is not used, the typed character will update the command line. The type of update to the command line (insert or replace) is controlled by the **Session | Command Line | Insert Text Into Command Line** option.

Update: Corrected in the 2020.0.7 pre-production release.

Configure User Tools - list of definitions should expand (i.e. show more rows) when dialog made larger; making everything bigger is kind of useless

The purpose of the flexibly-sized dialog is so that users with very high resolution displays can increase the size of the dialog's content. Expanding the list length when the dialog height was increased would prevent this.

Floated edit sessions get a separate entry in Alt+Tab menu (with no caption), but not on task bar

You are correct. We have reported these inter-related issues to the vendor of the Docking/Floating Panes GUI tools, which replaced the Microsoft-deprecated Win32 MFC tools on which APL+Win was based. If such a modification becomes available, we will incorporate it into APL64.

⎕ROWFIND is five times slower than raw APL (sort method)

When time permits, please send an example to support@apl2000.com illustrating this issue.

"Horizontal Tab Group" and "Vertical Tab Group" have opposite meaning than I expected

Please provide more information indicating how the horizontal and vertical tab groups have the opposite meaning than you'd expected.

Get rid of [imm] at the beginning of the penultimate line in ⎕DM; I'm repeatedly cursoring up to that line and modifying it to re-execute it

Displaying these actions in the history pane improves the accuracy of the history in APL64, which is a significant improvement over APL+Win. However, we will take your request into consideration.

⎕EDITEVENTS should use expressions (e.g. to invoke user commands), like the onEditStart etc. handlers, not function names…

…Requiring functions to be present in the workspace significantly reduces the utility of ⎕EDITEVENTS. Every time you load a new workspace, you have to bring in the support functions, then remember to erase them before saving. Is there a way to query the ⎕EDITEVENTS settings?

Your enhancement suggestion will be considered for implementation in a future version of APL64. It is not anticipated that this enhancement will be available in the first production version of APL64.

I would much prefer to enter statements in the "history" and ditch the command line…

…(which I think gains you little), or be able to hide it... Likewise, the command line seems superfluous. At least the latest version doesn't automatically put the focus on the command line after executing a line in the history; that's a real improvement.

Please read the *Help | Developer Version GUI | Editable/Non-editable Classic History* document to learn more about the classic history formats in APL64 and the classic history format in APL+Win.

There are scenarios where the Command Line is necessary in APL64. It is not anticipated that History options and developer version GUI features will be removed to limit all users to a specific historical perspective. Options have been and will continue to be incorporated into the developer version GUI, as you note, "...The latest version doesn't automatically put the focus on the command line after executing a line in the history; that's a real improvement..." This was done by providing the *Session | Classic History Format* options: *Is Editable* and Set *Focus When Editable* which are the defaults upon installation so that other users are not prevented from exploring other options and features in APL64.

Having an "auto-float" option and not having vertical real estate taken up by an empty function session pane would also be a plus. [There also] should be an option to always float edit sessions

To cause all new editors to float rather than dock use **Objects | Editors Pane Format | Layout | Editors Floated**. Changing this setting will apply to new editors only.

The horizontal GridSplitter between the editors pane and the history pane can be manually dragged to adjust the size of the editors pane. In the next APL64, pre-production version, the horizontal GridSplitter may be mouse left double clicked to minimize/maximize the editors pane, if it contains docked editors or not.

I find the interface cluttered and hard to work with a Microsoft Surface Pro, which has a 12" screen, so having function sessions take up precious vertical space is a negative

Most of the Microsoft Surface Pro machines have a screen resolution of 1920 x 1080 or better. At a Scale% of 100%, the Microsoft Surface Pro should provide the same display area as a much physically larger monitor with the same resolution. Please indicate the **Windows | Display Settings | Scale%** you

are using with the Microsoft Surface Pro 12" screen. If you are using a Scale% > 100% that will effectively reduce the Microsoft Surface Pro screen area significantly for all applications.

Ultimately all display area is precious. If the APL64 MainWindow is not presented full screen and the function editor of interest is floated without intersection with the MainWindow both can be used simultaneously. The **Objects | Editors Pane Format** menu provides several options for presenting function/variable editors.