Modifications Included Since APL64 Pre-Production Version 2020.0.8

Table of Contents

Overview 3

Enhancements 3

New ⎕nfe methods: SetLastWriteTime and SetLastWriteTimeUTC 3

New ⎕skd action: '?' and 'Help' 3

Update to ⎕SKD 'ContainsKey' 3

Update to ⎕SKD 'ContainsKeys' 3

Performance Improvements 4

Integer MEMBER of Integer Vector 4

Float MEMBER of Integer Vector 4

Bool MEMBER of Integer Vector 5

Integer MEMBER of Integer Vector 5

Float MEMBER of Bool 6

Bool MEMBER of Bool 6

Bug Fixes 7

Correction to ⎕symb system function in System Functions manual 7

A function editor tab did not update the name typed or pasted on line 0 7

Evaluators' Comments & APLNow's Responses 7

Upon VALUE ERROR due to missing function, after adding function to ws and resuming, get TYPE CHECK ERROR **(case #5329 and #5571)** 7

Could you add following methods to ⎕nfe: SetLastWriteTime and SetLastWriteTimeUTC? (case #5575) 8

The caret returns in the History too quickly before any result is displayed (case #5572) 8

⎕nfe'GetLastAccessTime' returning false information? (case #5486) 8

⎕XL: Implement a new explicit method for creating a new workbook;… 8

⎕XL: EITHER implement a property for the function that specifies the workbook,… 8

⎕XL: The FromAPL and ToAPL methods should verify the existence of the workbook and fail when found missing. (case #5504.2) 8

⎕XL: The additional methods that permit R1C1/A1 conversions are superfluous… 9

⎕XL: The GetUsedRange method appears to return a single region (a region is bound… 9

⎕XL: The GetWorksheetNames method is inadequate when the workbook is created… 9

⎕XL: The GetA1ColNameFromColNum method is misleading… 9

□XL: The parameters inclRows, inclCols are vectors of Indices of rows and columns… 9
□XL: A workbook specified as the first argument for □XL may have been created… 10
□XL: There is no safeguard against overwriting or overlapping ranges. (case #5504.9) 10
Could you please send me the original APLNow32.ini file? (case #5570) 11
Update: ' '^.= 1 2 3 returned "SYSTEM ERROR:Exception: CS0103: The name 'rfcs' does not exist in the current context" (case #5538) 11
0∈Mρ1 and 1∈Mρ1 are both very slower in APL64 than APL+Win (case #5442)..... 11
Is there a good reason why APL64 is requiring .Net 5 and not .Net 6? (case #5595)..... 11
I know □mom is deprecated in APL64. Could sneak in a rough equivalent for APL64? (case #5600) .. 11
Is it possible to open a new object editor in the same size and location as that of the previous object editor? (case ##5602) 11
Clicking on the Debug pane scroll bar does not always send focus to the debug pane (case #5601.1-3) 12
Are any of the APL+Win command line arguments like WsNotLow, etc. still valid with APL64? (case #5260.1) 12
If yes, how should one specify them on the APL64 command line? (case #5620.2) 13
Is there a way to reserve a specific amount of Windows memory for APLNow32? (case #5260.3) 13
Erasing from caret to end of session does not work correctly (case #5639)..... 13
The syntax color is wrong for comments on :endregion lines (case #5641) 13
Double right clicking a function[line] in the APL Session does not always open the function [editor] with the caret on the right line! (case #5637) 14
Double right clicking function[linenumber] prints ")ed ∇zzlcchart" in the APL64 Session (case #5638) 14
Typing in a floated editor, in a large function, can be extremely slow (case #5636) 15
I was trying to]upack the large component file and got a DOMAIN ERROR (case #5613) 15
My example with assigning the result of '#' □skd 'instances' to □wres produced "SYSTEM ERROR:Aval.Categorize encountered an uncategorized child element at index=1" (case #5649)..... 15
One can retrieve the □string documentation with □string'?', why not □skd '?' (case #5640) 15

Overview

This document describes the enhancements and bug fixes incorporated into the first production release of APL64, **2022.0.1**. This document also describes the evaluator comments and APLNow responses based on the previous pre-production release of APL64.

Enhancements

New ⎕nfe methods: SetLastWriteTime and SetLastWriteTimeUTC

Refer to **Help | APL Language | Using ⎕NFE** for their complete descriptions.

New ⎕skd action: '?' and 'Help'

Returns help information for the ⎕skd system function.

Update to ⎕SKD 'ContainsKey'

Accepts string or character key value.

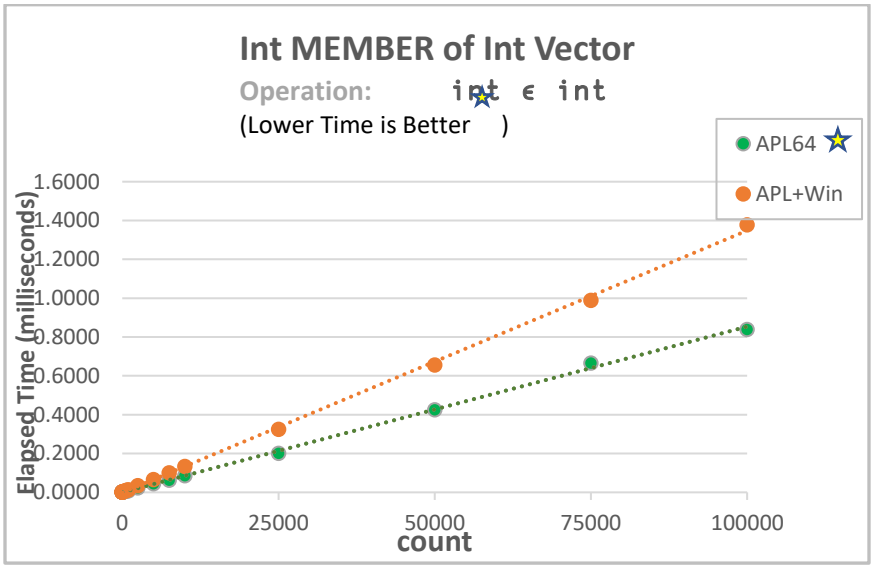
Update to ⎕SKD 'ContainsKeys'

Accepts a vector of strings or character vectors.

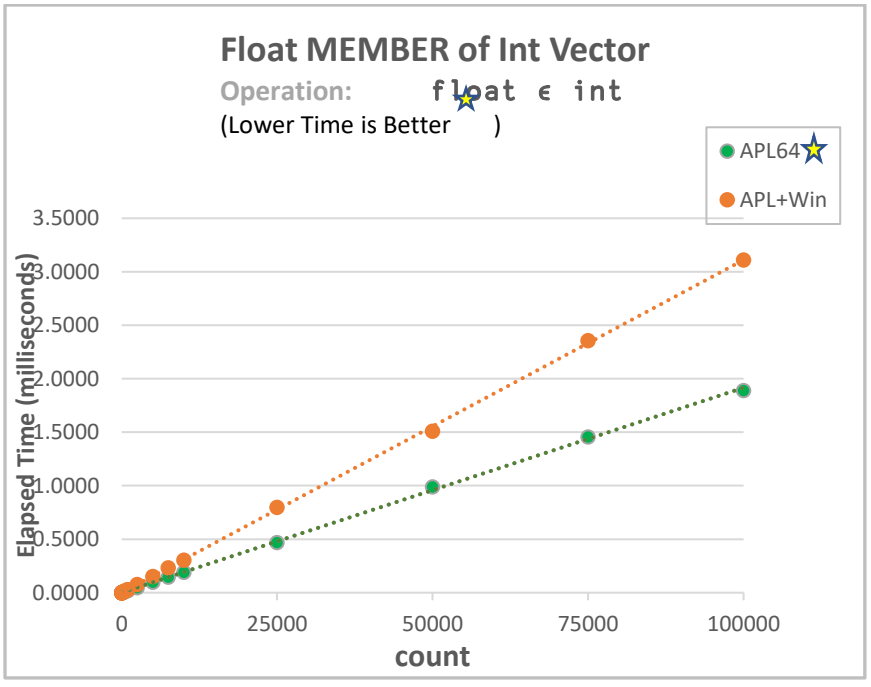
Performance Improvements

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

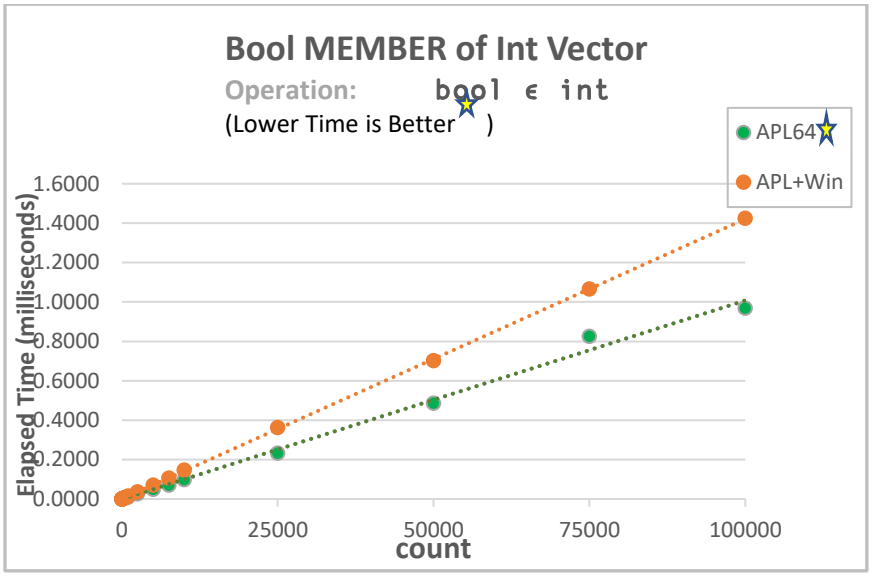
Integer MEMBER of Integer Vector



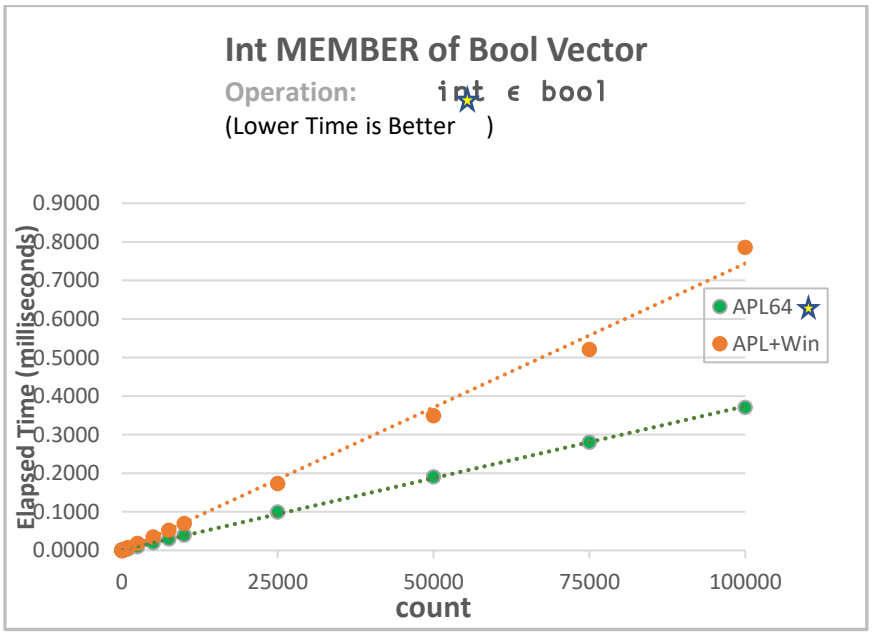
Float MEMBER of Integer Vector



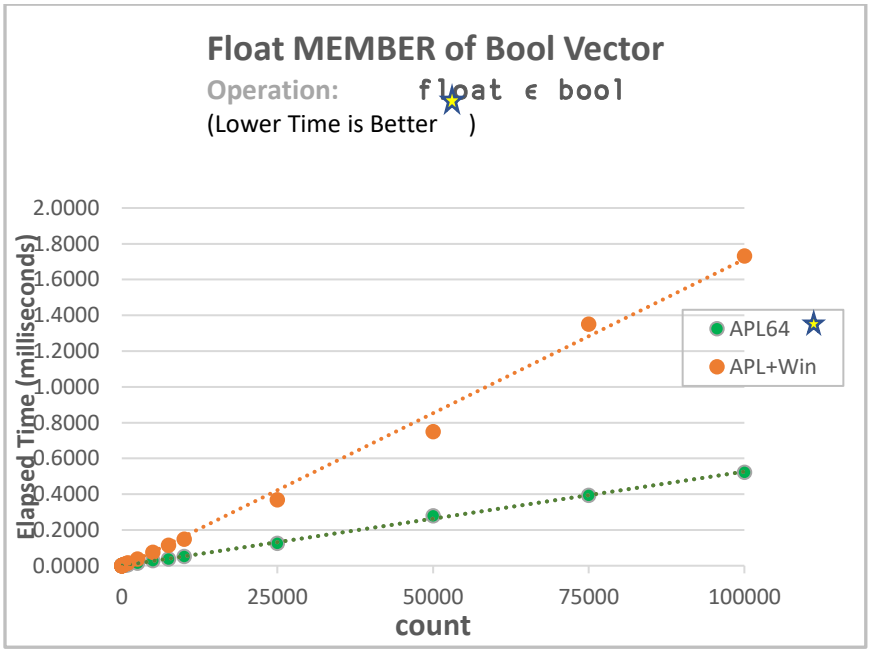
Bool MEMBER of Integer Vector



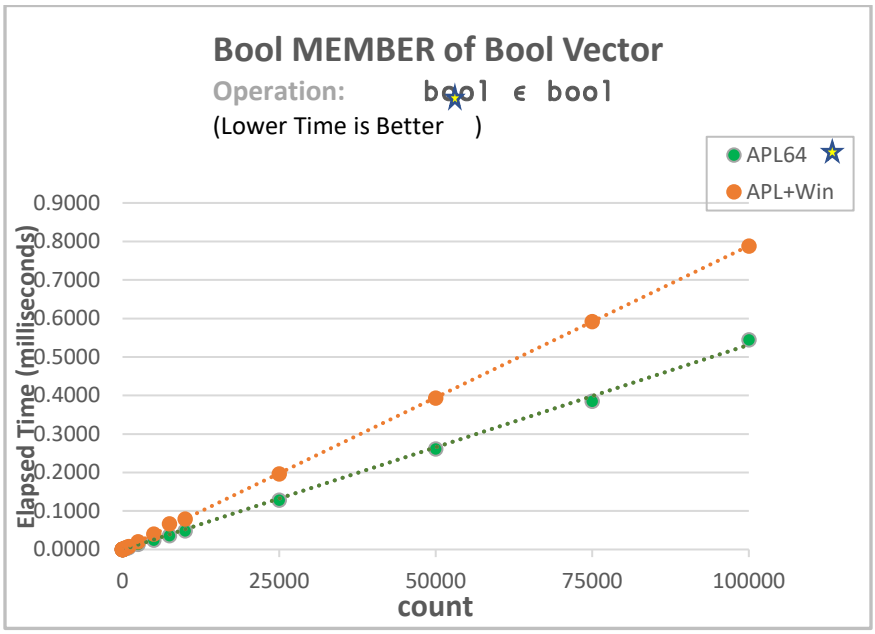
Integer MEMBER of Integer Vector



Float MEMBER of Bool



Bool MEMBER of Bool



Bug Fixes

Correction to ⎕symb system function in System Functions manual

The Results section in the documentation for the ⎕symb system function contained the erroneous statement "**you can change the size with the system command)symbols.**". The)symbols system command does not support changing the size of the symbol table.

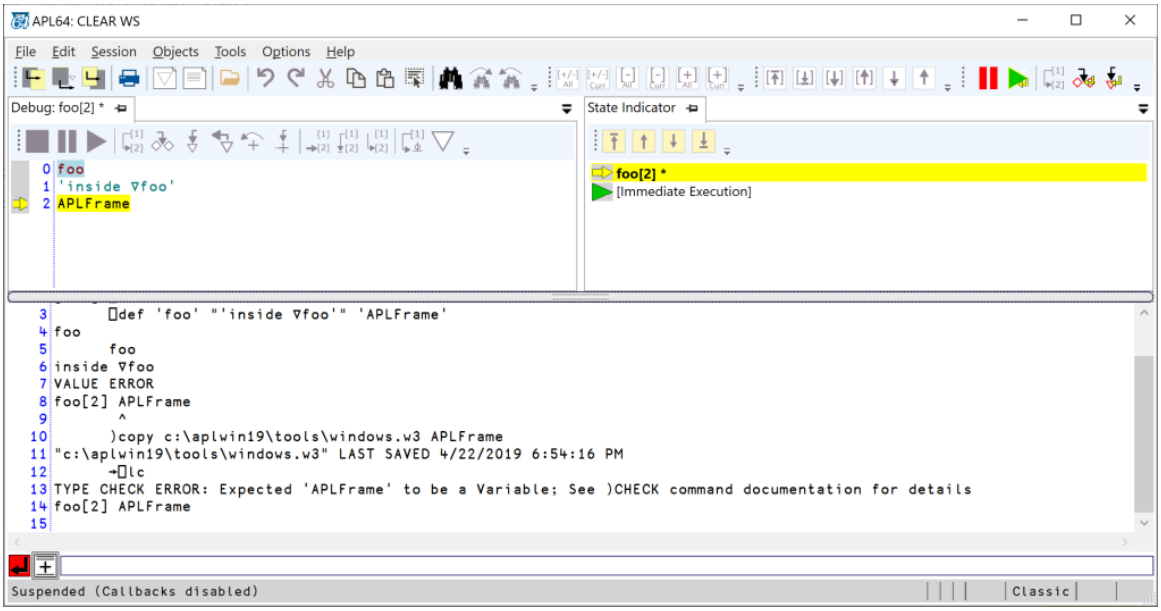
A function editor tab did not update the name typed or pasted on line 0

Evaluators' Comments & APLNow's Responses

This section contains the summary of the evaluators' comments and APLNow's responses to the Pre-Production Version 2020.0.8. 18996 of APL64. We would like to thank the evaluators for their contribution.

Upon VALUE ERROR due to missing function, after adding function to ws and resuming, get TYPE CHECK ERROR (case #5329 and #5571)

E.g.,



When this occurs, clear the state indicator with system commands **)SIC** or **)RESET** then execute the system command **)CHECK ON** to allow the APL functions to be rebuilt (reflowed) in the workspace. When this is completed, execute the system command **)CHECK OFF** to disable the command and then save the workspace.

Could you add following methods to ⎕nfe: SetLastWriteTime and SetLastWriteTimeUTC? (case #5575)

This enhancement will be included in the first production release of APL64.

The caret returns in the History too quickly before any result is displayed (case #5572)

This will be improved in the first production release of APL64.

⎕nfe'GetLastAccessTime' returning false information? (case #5486)

It appears that you are comparing disparate DateTimes in your example:

GetLastAccessTime from ⎕NFE is not the same as 'Date modified' which is the column you are illustrating in Windows Explorer. I suggest:

(a) Go to Windows Explorer and target the applicable folder
(b) Select Details view
(c) Right click on any column and add the 'Date accessed' column
(d) Retry your example using ⎕NFE 'GetLastWriteTime' and ⎕NFE 'GetLastAccessTime'

To review the definition of the ⎕NFE actions, please see: FileSystemInfo Class (System.IO) | Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/api/system.io.filesysteminfo?view=net-6.0

⎕XL: Implement a new explicit method for creating a new workbook;…

…this method will need two parameters, namely, the fully qualified name of the workbook and the number of sheets to add, default 1. (case #5504.1)

This enhancement will be considered in a future version of APL64.

⎕XL: EITHER implement a property for the function that specifies the workbook,…

…thereby defaulting to that workbook for all subsequent calls to FromAPL and ToAPL OR allow for a syntax that permits a separation of the names of the workbook and worksheet by a single character; exclamation mark (!) is used by both the Excel.Application object model and the ACE OLEDB provider. (case #5504.1)

This enhancement will be considered in a future version of APL64.

⎕XL: The FromAPL and ToAPL methods should verify the existence of the workbook and fail when found missing. (case #5504.2)

The ToAPL action checks that the user-provided workbook path is accessible and throws an exception if that is not the case.

In the first production release of APL64, the FromAPL action will create the target workbook and worksheet if it does not exist.

□XL: The additional methods that permit R1C1/A1 conversions are superfluous…

…since the intuitive way of addressing row and column positions from APL is in terms of ordinal positions. (case #5504.3)

These are not superfluous for the APL2000 customers who have requested them. Different strokes for different folks.

□XL: The GetUsedRange method appears to return a single region (a region is bound…

…by blank rows at the top and bottom and blank columns to the left and right. This is anomalous since the FromAPL method allows cell population anywhere in a worksheet. A more appropriate method might be GetUsedRegions that returns all used regions. (case #5504.4)

The UsedRange result is specifically defined by Microsoft and the 'GetUsedRange' action of □XL conforms to that Microsoft definition. There is nothing anomalous about providing a well-defined result. See here for the Microsoft definition of UsedRange: https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.tools.excel.worksheet.usedrange?view=vsto-2017

'UsedRegions' has no Excel definition, so □XL cannot provide it.

□XL: The GetWorksheetNames method is inadequate when the workbook is created…

…by Excel.Application and may contain named ranges and tables: it ought to be possible to specify those named ranges and table names with the FromAPL and ToAPL methods.

A more appropriate method might be GetSchema that returns all worksheet names, names of ranges and table names. (case #5504.5)

WorksheetNames is specifically defined by Microsoft and it is properly returned by □XL.
There is an Excel definition for 'Names', which includes Named Ranges, plus other named objects.
Adding an □XL action for 'Names' will be considered in a future version of APL64.

'GetSchema' has no Excel definition, so □XL cannot provide it.

□XL: The GetA1ColNameFromColNum method is misleading…

…It is simply an R1C1 to A1 conversion method with a tacit assumption that R1 is 1. My first impression was that it would return the header of a column from any range in the workbook. (case #5504.6)

The name of the □XL action 'GetA1ColNameFromColNum' makes it clear that it is going to return the Column Name, which is a Microsoft-defined Excel value. There is no Excel definition of 'column header'. An Excel row which contains 'column headers' is merely a user-defined thing.

□XL: The parameters inclRows, inclCols are vectors of Indices of rows and columns…

…respectively, in a worksheet and whose order is significant and index-origin sensitive.

a. An Excel workbook (which may be built with a file with CSV or TSB content) always has rows and column references in index-origin 1. APL64 documentation should make this clear and signal an error if it detects that either parameter has content based on index-origin 0.

b. The order within each of these parameters is an expensive indulgence in an array based language which is capable of reordering rows and columns in arrays with ease.

c. In the interest of simplification, consider the following:

1 This will avoid a future dilemma with respect to backward compatibility.

I. For the FromAPL method, replace these two parameters by a single one that denotes the top-left cell of the rectangular block of cells and determine the cell addresses from the shape of the values being written to the worksheet.

II. For the ToAPL method, either have a consistent syntax and determine the number of rows and columns to read by determining the populated region from the top-left cell specified or two arguments, respectively specifying the top-left cell and the shape (number of rows and columns) of the desired content. (case #5504.7)

Thank you for your suggestions, however your suggestion is not the design established for ⎕XL. If may be possible to implement an alternative design, in addition to the current design, in a future version of APL64.

⎕XL: A workbook specified as the first argument for ⎕XL may have been created…

…by Excel.Application and may contain formulae and array formulae. This system function adds handling for empty values and dates (because APL does not handle dates) but not for formulae.

a. Should this function eliminate the user option for handling empty values and return such values using some default? Perhaps in the same way as the COM interface does?

b. Should this function return date columns as text? Or should this function implement ToOADate and FromOADate methods as found in System.DateTime? (case #5504.8)

A cell formula is merely text prefixed by '='. ⎕XL can be used to put/get a formula into/from an Excel cell.

As indicated in the documentation and examples for ⎕XL, when an empty Excel cell is encountered, the result value in APL64 is a default value and that default value is user-defined.

There are two ⎕XL options for DateTime handling: 'AplDateTime' and 'XlDateTime' (OADateTime).

⎕XL: There is no safeguard against overwriting or overlapping ranges. (case #5504.9)

Your observation is correct. This is analogous to standard APL, i.e. there is no safeguard to prevent: A[5]←1234

Could you please send me the original APLNow32.ini file? (case #5570)

Beware that APL64 does not install an APLNow32.ini file comparable to the APLW.ini file in APL+Win. It is expected the user will create it if one is necessary.

Update: ' '^.= 1 2 3 returned "SYSTEM ERROR:Exception: CS0103: The name 'rfcs' does not exist in the current context" (case #5538)

The fix for this is in the first production release of APL64.

0∈Mρ1 and 1∈Mρ1 are both very slower in APL64 than APL+Win (case #5442)

There will be a marginal performance improvement for the Bool∈Bool operation in the first production release of APL64.

Is there a good reason why APL64 is requiring .Net 5 and not .Net 6? (case #5595)

Yes. The first production version of APL64 is designed for .Net 5 and updating to .Net 6 would have disrupted and delayed the scheduled release of it. It is the intention of the APL64 development team to update APL64 from .Net 5 to 6 for future releases, however there are potentially some modifications to the APL64 source code that may be required, which will require further analysis, design and testing.

I know ⎕mom is deprecated in APL64. Could sneak in a rough equivalent for APL64? (case #5600)

⎕MOM will not be implemented in APL64. If you are asking for some syntax like A.B in APL64 analogous to ⎕MOM, that will also not be implemented in APL64. The 'dot' notation syntax is being reserved in APL64 for accessing class members, both those in external .Net objects and those in APL64 class objects. Implementing a 'dot' notation restricted to the ⎕MOM design in APL64 now would restrict future development of APL64. The 'dot' notation concept also exposes some potential conflicts between APL syntax modernization efforts and legacy APL syntax. The timing of such an OOP oriented 'dot' notation has not been established for APL64 at this time.

Is it possible to open a new object editor in the same size and location as that of the previous object editor? (case ##5602)

These are the potential problems with your proposed design: (a) There may be no prior object editor and (b) if the prior object editor is still active, the new editor would obscure the prior object editor.

With the current design, the user knows that every new editor will be presented in the upper left in a 'standard' size. The new editor location and size could be made user-editable settings in the *Objects | Editors Pane Format* menu, but this could not be deemed an urgent enhancement.

Clicking on the Debug pane scroll bar does not always send focus to the debug pane (case #5601.1-3)

Your scenario (#5601.1):
(a) Function suspended in debugger with more lines than can be displayed in the available space
(b) Click on the vertical scroll bar of the debug pane
(c) Focus does not go to the Debug pane

A scrollbar is a focus target which is separate from the content which it scrolls.

In .Net WPF there is a distinction made between the local focus and keyboard focus. Every pane in an application can have a local focus point within that pane and it will become the caret location when that pane gets the keyboard focus. However, only one GUI object (pane, button, scrollbar, etc.) can have the keyboard focus, so clicking on a scrollbar puts the keyboard focus on it and not elsewhere. It would not be reasonable to move the focus somewhere else every time the user clicked on a scrollbar because then the user would not be able to manipulate that scrollbar.

In this scenario put the keyboard focus within the pane by clicking within it and not on the scrollbar or by using Ctrl+Tab.

Your scenario (#5601.2)
(a) Function suspended in debugger with more lines than can be displayed in the available space
(b) Click on the SI pane
(c) Click on the vertical scroll bar of the debug pane
(c) Focus goes to the Debug pane

Based on the Microsoft specifications for .Net WPF, we believe that the focus should not move to the Debug pane. We will ask the vendor of the Dock/Float panes for an explanation of this behavior.

Your scenario (#5601.3):
(a) Function suspended in debugger with more lines than can be displayed in the available space
(b) Click on the History pane
(c) Click on the vertical scroll bar of the debug pane
(c) Focus sometimes goes to the Debug pane or not.

We observe that the focus remains on the History pane and does not move to the debug pane. Please provide a scenario which reliably duplicates your Scenario (3).

Are any of the APL+Win command line arguments like WsNotLow, etc. still valid with APL64? (case #5260.1)

APL+Win configuration options may be used with APL64. There are some APL32 configuration options which do not apply and will be ignored, e.g. color specifications.

If yes, how should one specify them on the APL64 command line? (case #5620.2)

First read the APL64 *Help | Command Line Arguments* documentation about the APL32CONFIGSRCS (A32CS) command line argument.

Method (1): Read the APL64 *Help | Command Line Arguments* documentation about the APL32 command line argument. There can be multiple APL32= command line arguments for APL64 which specify a specific section, key and value, e.g. apl32=[Config]Wssize=20M

Method (2): Create an ini-format APLNow32 configuration file with the desired configuration specifications in a manner analogous to the APLW.INI file format and use the APL32CONFIGPATH (A32CPS) command line argument to specify its path.

Is there a way to reserve a specific amount of Windows memory for APLNow32? (case #5260.3)

The same options for memory arrangement in APL+Win apply to APL64. The Win32 memory limits apply to the APLNow32 component of APL64.

Erasing from caret to end of session does not work correctly (case #5639)

This behavior is by design in APL64, although it may be different than in APL+Win which may be inconsistent in its handling of different types of user changes to the editable classic history (scratch pad).

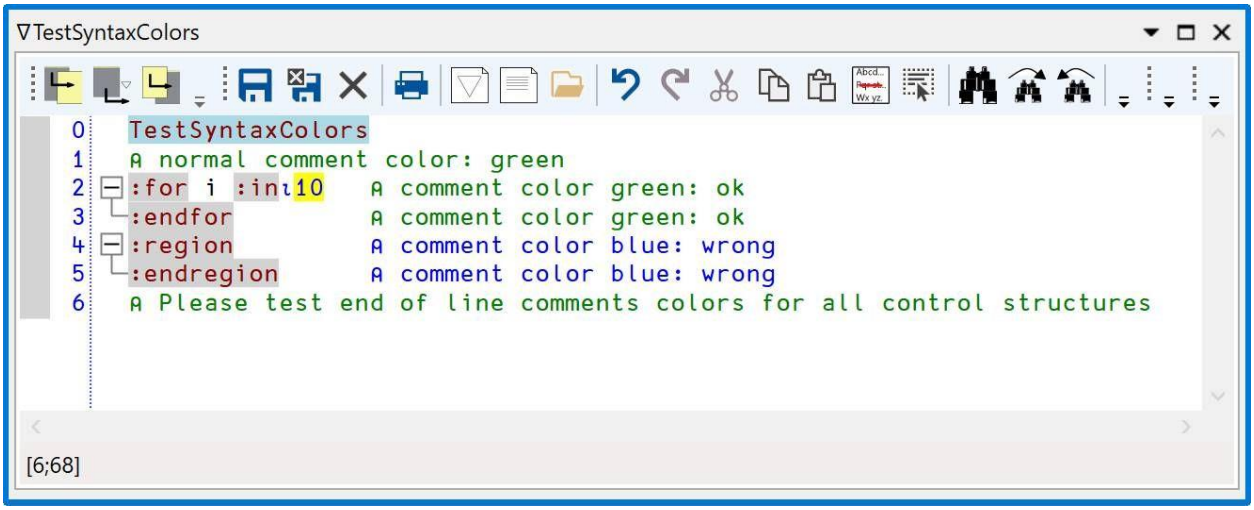
Here is the design in APL64 for the editable classic history:

- **(a) When the caret is placed on a history line, the text on that line is captured prior to any user editing of that line.**
- **(b) Because this is the editable classic history, the user is free to edit that line in any applicable manner, including typing, selecting (linear and rectilinear), pasting deleting and in this case Alt+Shift+End.**
- **(c) APL64 will process all such user-editing of the line in a consistent manner, no matter how it was accomplished by the user.**
- **(d) If, after editing the line, the user keeps the caret on that modified line and clicks the Enter/Return key, the user-edited text of that line is captured, and the pre-user-edited text is restored to that line (analogous to APL+Win).**
- **(e) The user-edited text which was captured (a new APL executable statement) is sent to the interpreter for execution and this results in a new line in the history containing the new APL executable statement followed by any interpreter output generated by the execution of that new APL executable statement.**

In this scenario, the new APL executable statement was empty text, so the modified line was restored, and two empty lines were added to the history, which is illustrated by the last user-provided screen capture.

The syntax color is wrong for comments on :endregion lines (case #5641)

Example,



We'll be in touch with an update after investigating your issue.

Double right clicking a function[line] in the APL Session does not always open the function [editor] with the caret on the right line! (case #5637)

Currently in APL64, the caret offset in a function editor is not being saved when that function editor is closed. We will investigate if this is a possible enhancement. If so, it will be considered for inclusion in a future version of APL64.

Double right clicking function[linenumber] prints ")ed ∇zzlcchart" in the APL64 Session (case #5638)

This issue was addressed in the evaluator's comment and response section in the recent July 2022 APL64 Project History document included with the 2020.0.8 pre-production release. Below is the excerpt.

Comment/Response:

User comment: I pressed Ctrl + Shift + O with the cursor on quadXL and)ed ∇quadXL appears in the session window (case #5506)

This behavior in 2020.0.7 is by design and documented so in the last bullet item in the H*elp | APL+Win Compatibility | Compatibility Modifications for Existing APL+Win Applications*:

To maintain an accurate history in the APL64 Developer version, any modal editor initiated from the APL64 Developer version will update the history pane with the appropriate APL executable statement.

The fact that APL+Win didn't do this was an oversight.

Typing in a floated editor, in a large function, can be extremely slow (case #5636)

We have now seen observed this ourselves when the editor is floated and docked. The issue will be investigated and if there is a solution, it will be considered for inclusion in a future version of APL64.

I was trying to]upack the large component file and got a DOMAIN ERROR (case #5613)

This is a bug in]UPACK and will be addressed in the first production release of APL64.

My example with assigning the result of '#' □skd 'instances' to □wres produced "SYSTEM ERROR:Aval.Categorize encountered an uncategorized child element at index=1" (case #5649)

□wres is an APLNow32 system variable, so it cannot be assigned a value with datatype unsupported by APL+Win, e.g., a string.

In the next APL64 version, the result of '#' □SKD 'instances' will be a vector of character vectors instead of a vector of APL64 strings. This will enable the result of the □SKD 'instances' action to be assigned to □wres. Where possible, the key argument to applicable □skd actions can now be a character scalar, character vector, or string datatype. For □skd actions which return keys, they will continue to be returned as string datatypes.

One can retrieve the □string documentation with □string'?', why not □skd '?' (case #5640)

The enhancement you requested will be in the first production release of APL64 release.