Modifications Included in APL64 Pre-Production Version (2020.0.4)

Table of Contents

Overview

This document describes the enhancements and bug fixes incorporated into the fourth pre-production version of APL64 released in January 2022. This document also describes the evaluator comments and APLNow responses based on the prior pre-production versions of APL64.

Enhancements

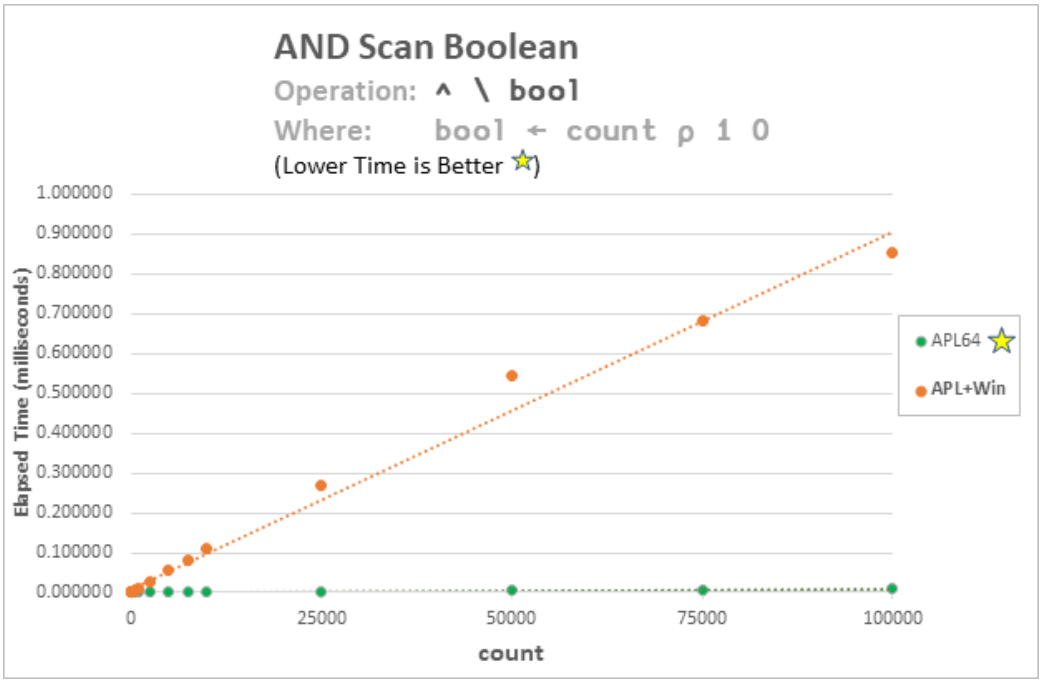
Boolean Data Type Enhancements

For Boolean data types, the operation performance and memory (storage) requirements have been significantly improved.

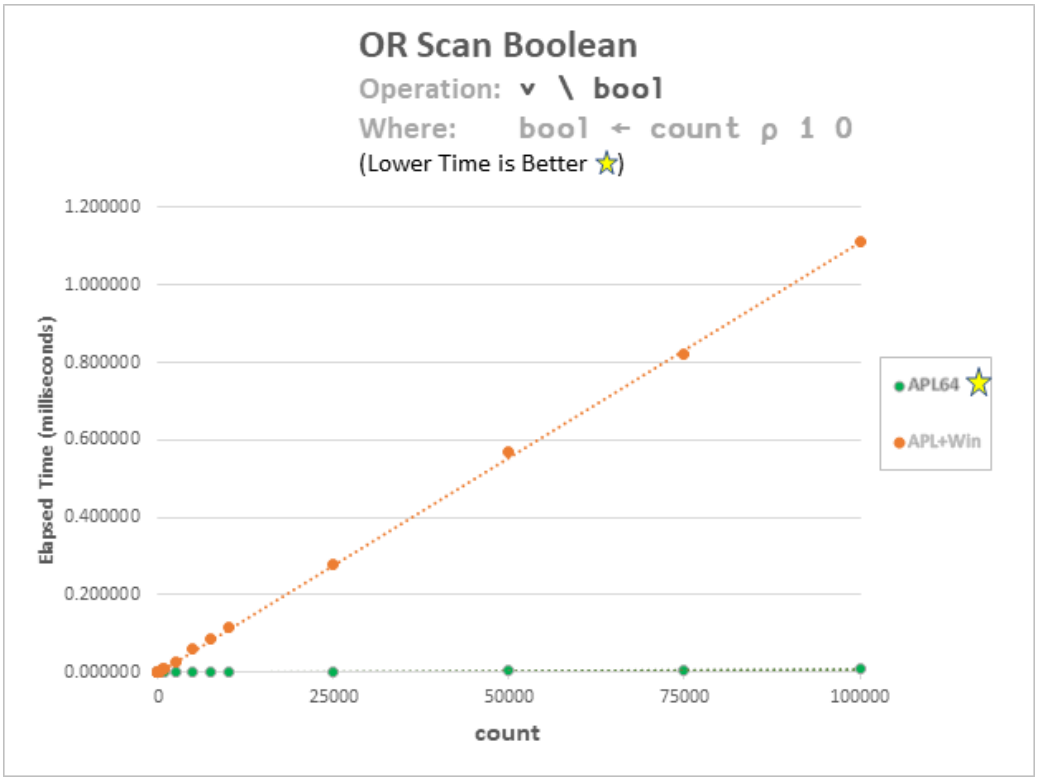
Operation Performance Improvements in APL64

The graphs below provide a glimpse of the significant performance improvements for Boolean data in APL64 compared to APL+Win 19.

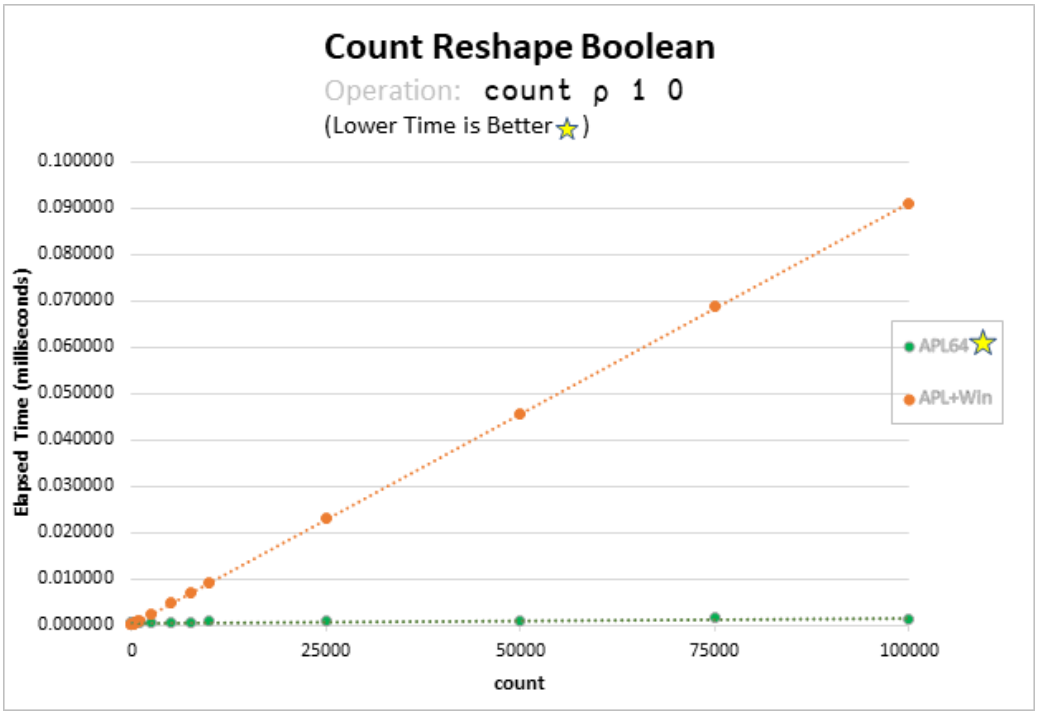
AND Scan Boolean



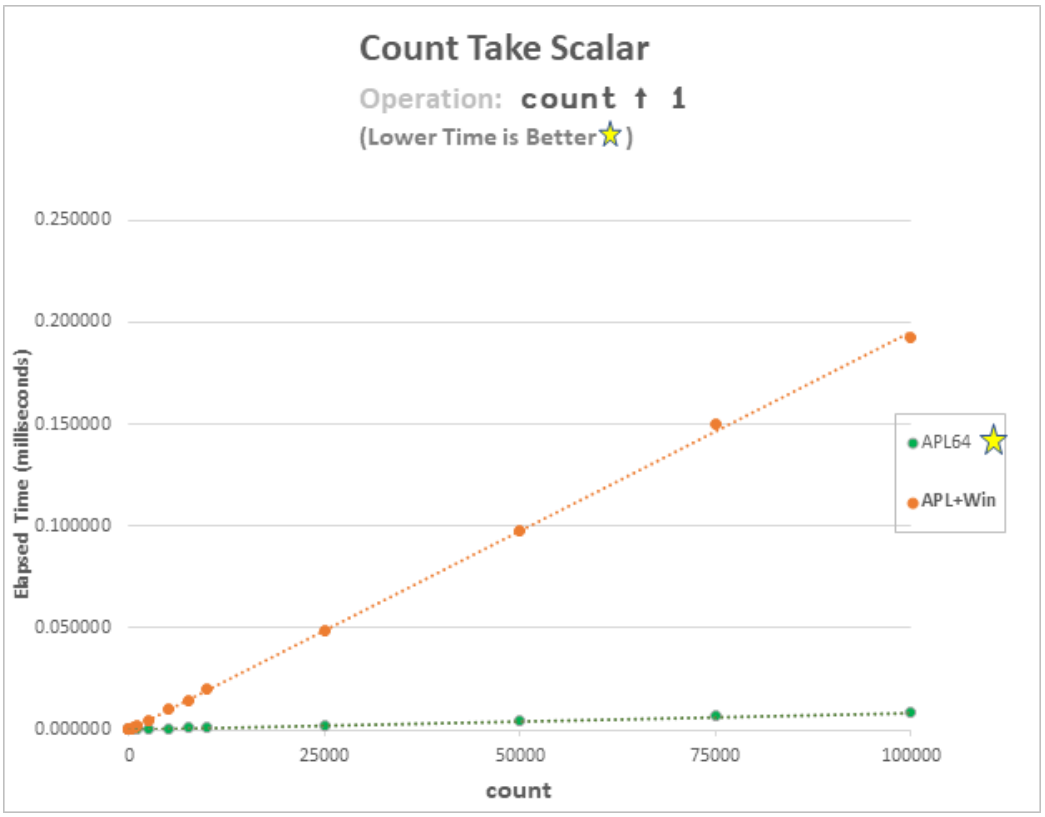
OR Scan Boolean



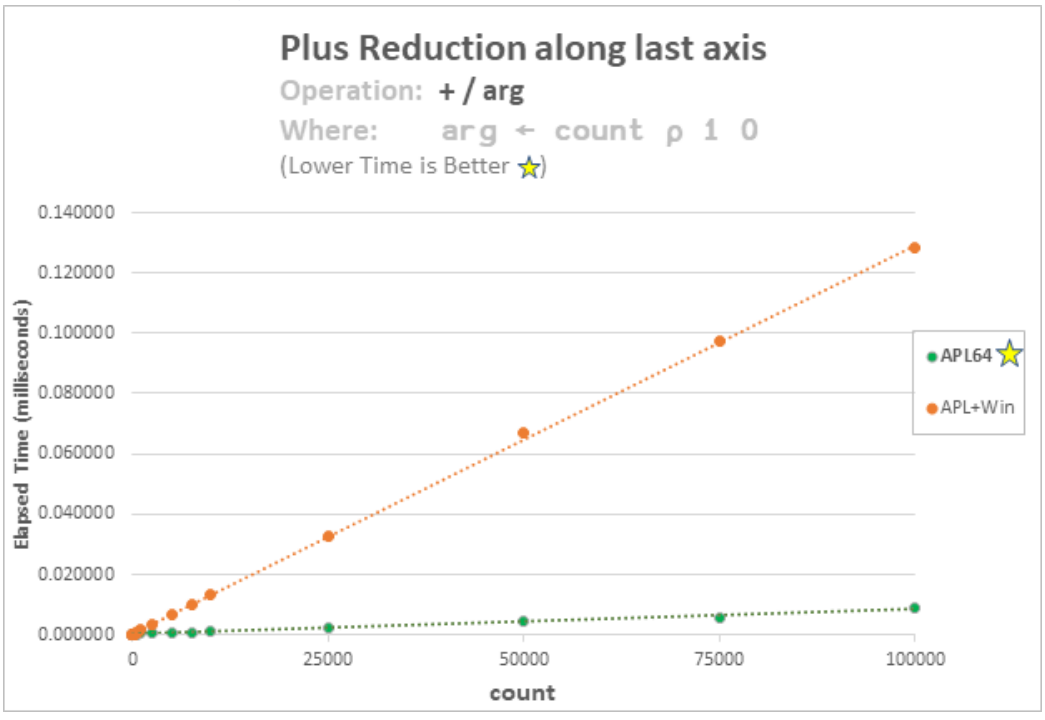
Count Reshape Boolean



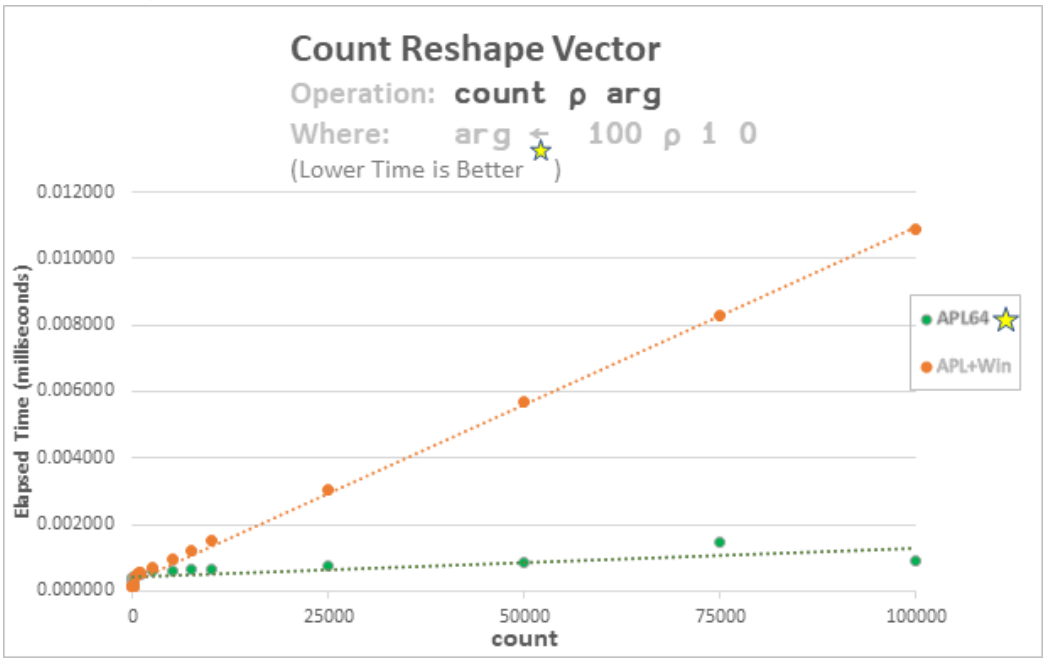
Count Take Scalar



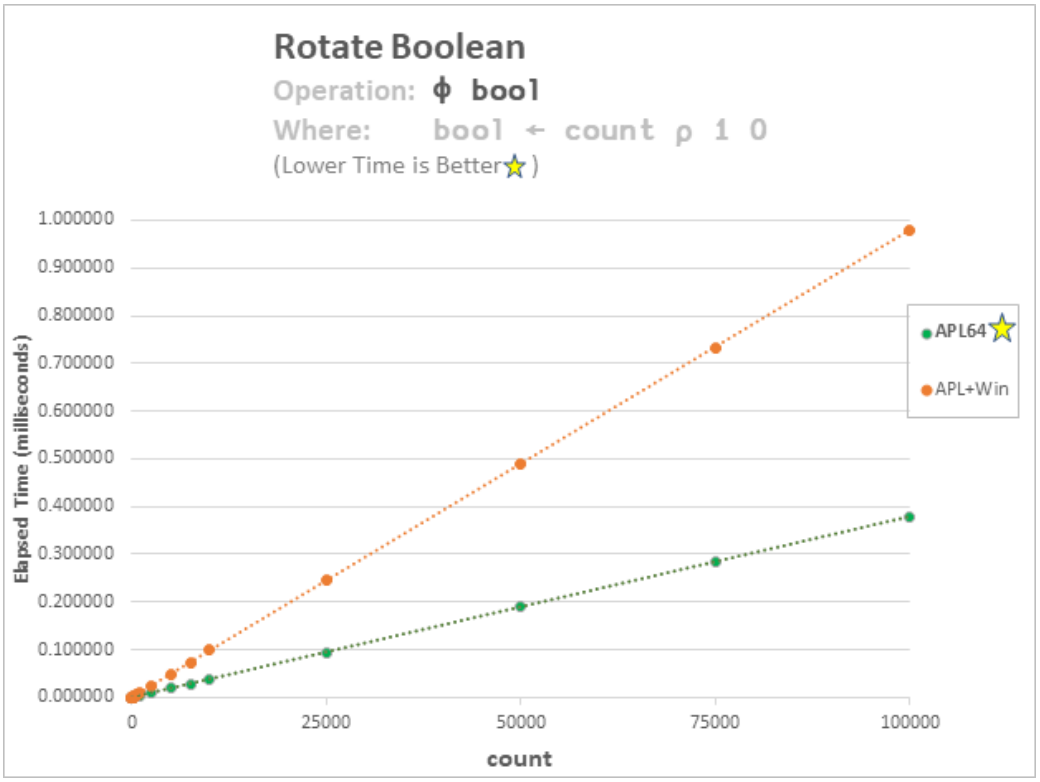
Plus Reduction Along Last Axis



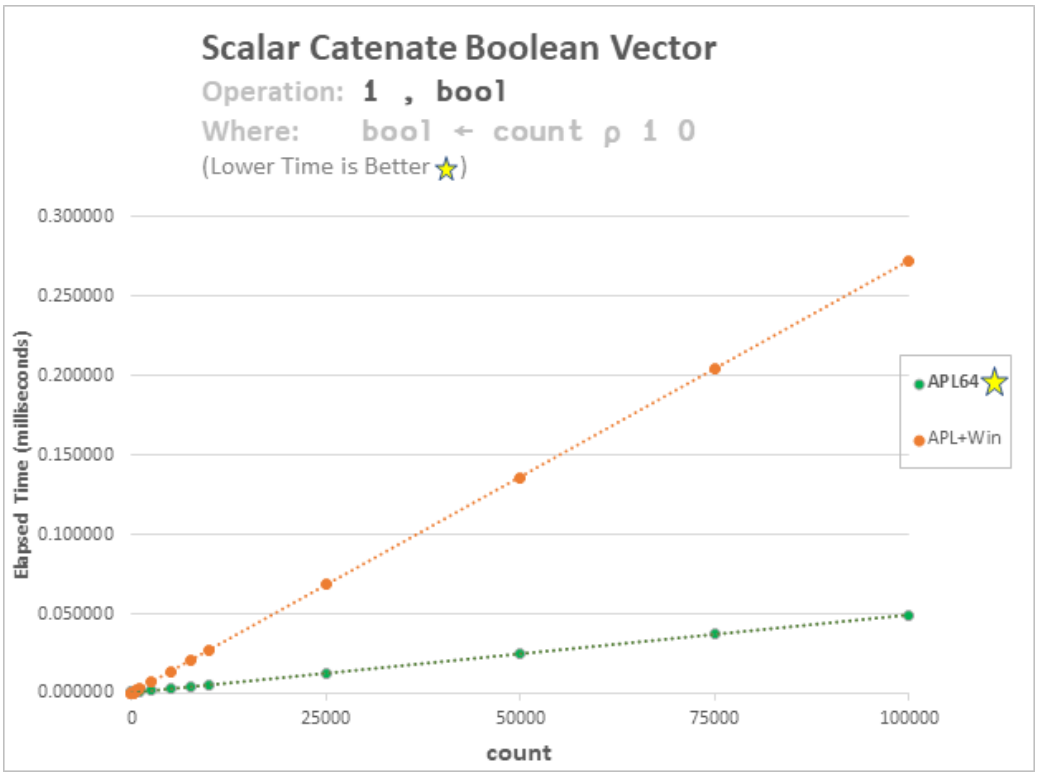
Count Reshape Vector



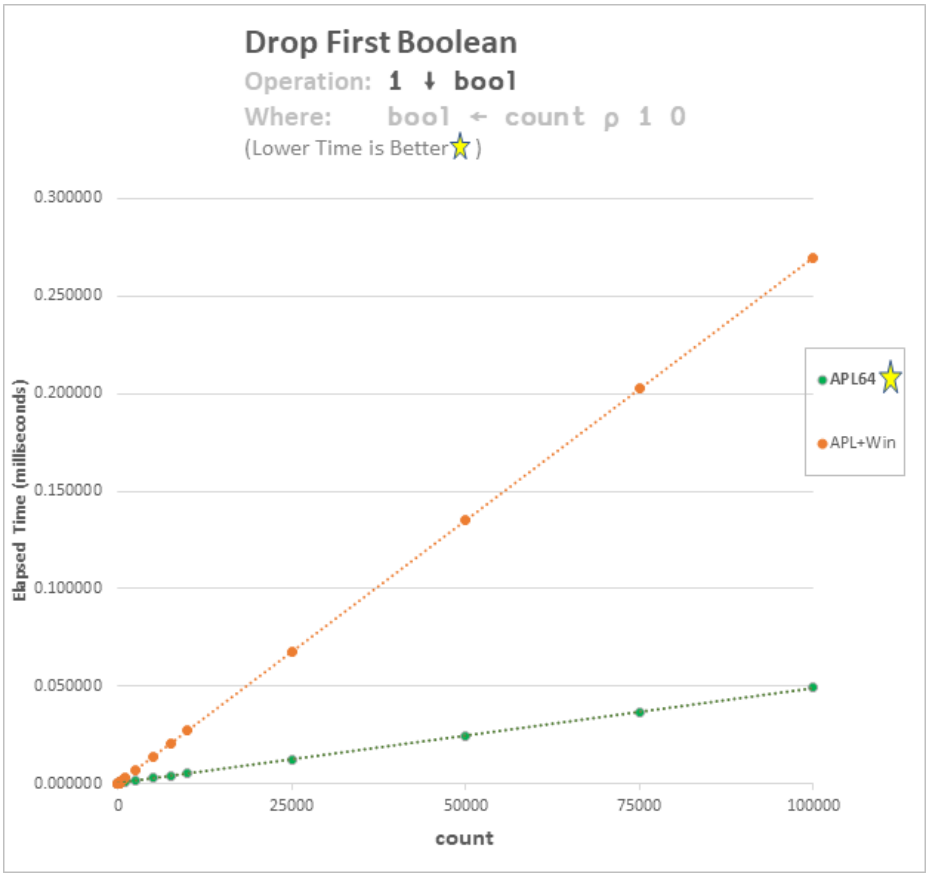
Rotate Boolean



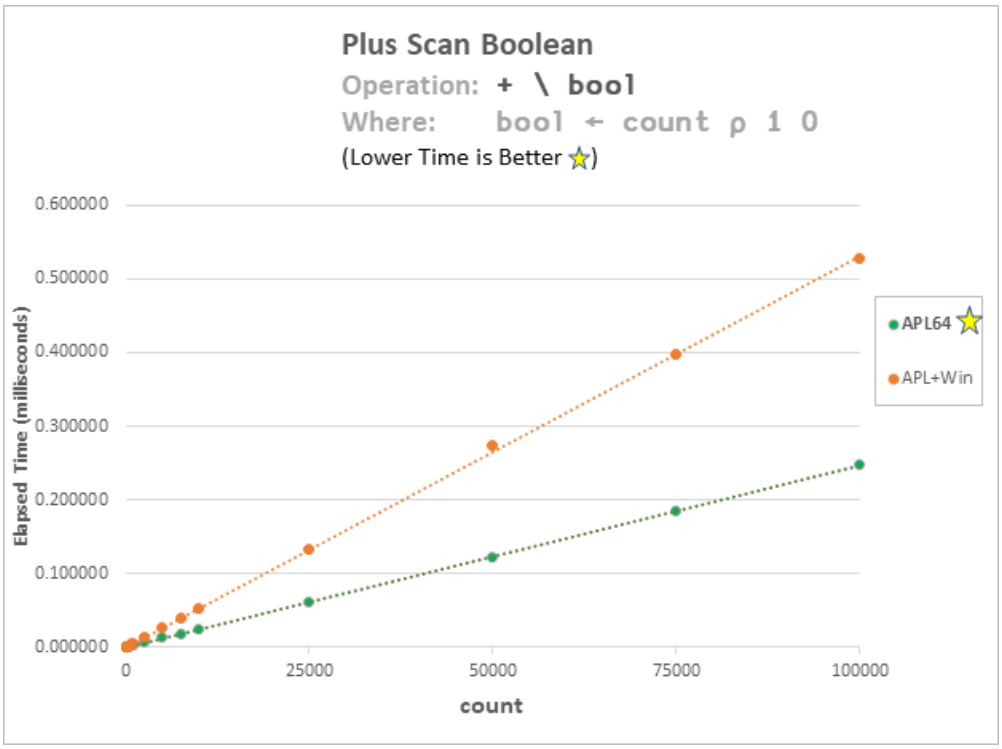
Scalar Catenate Boolean Vector



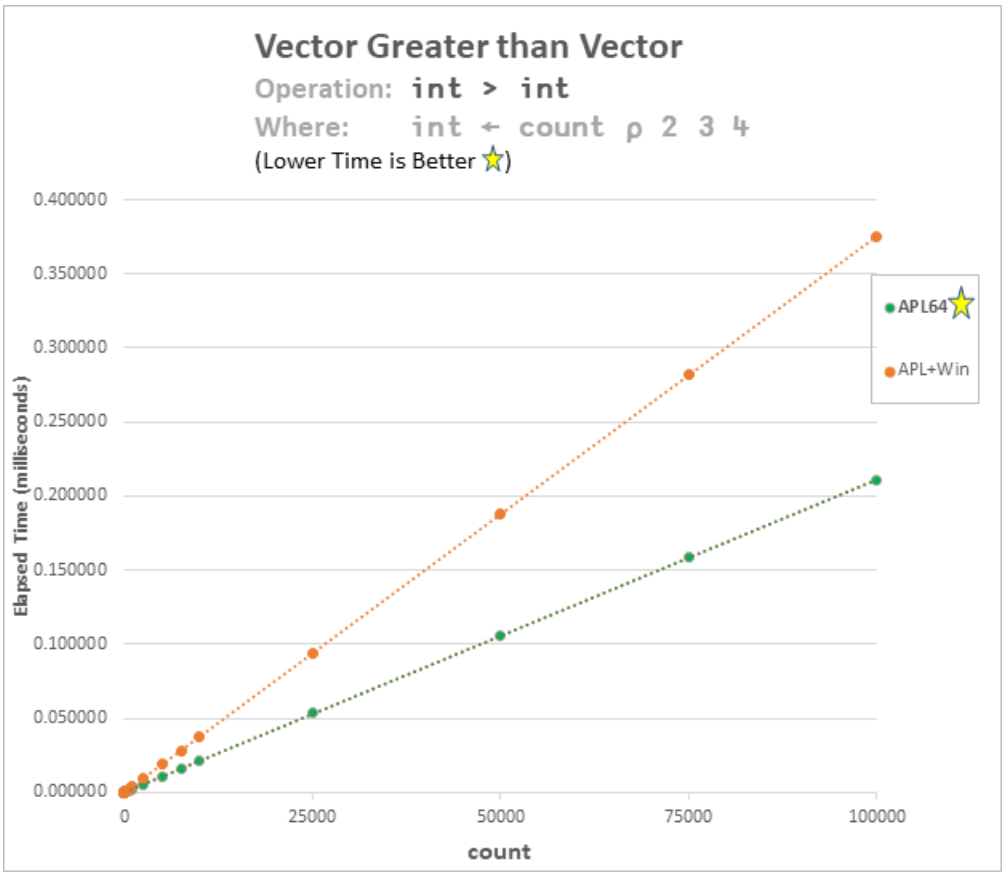
Drop First Boolean



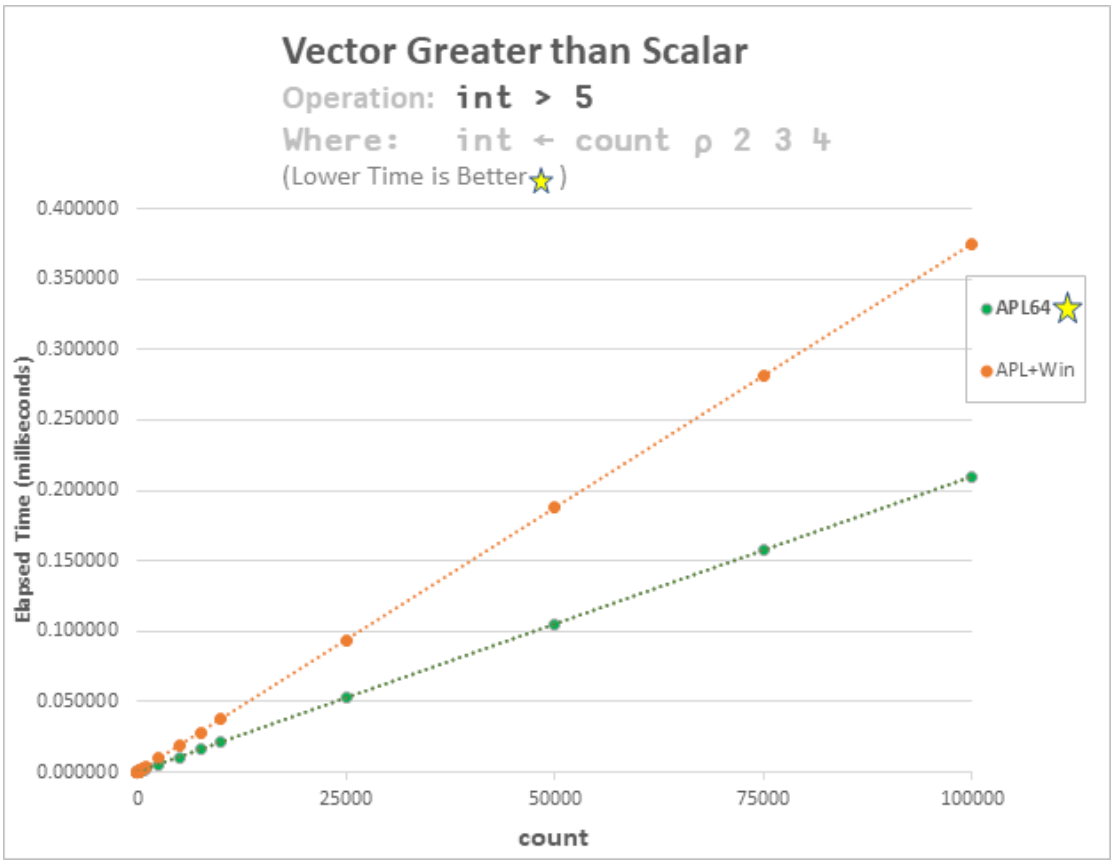
Plus Scan Boolean



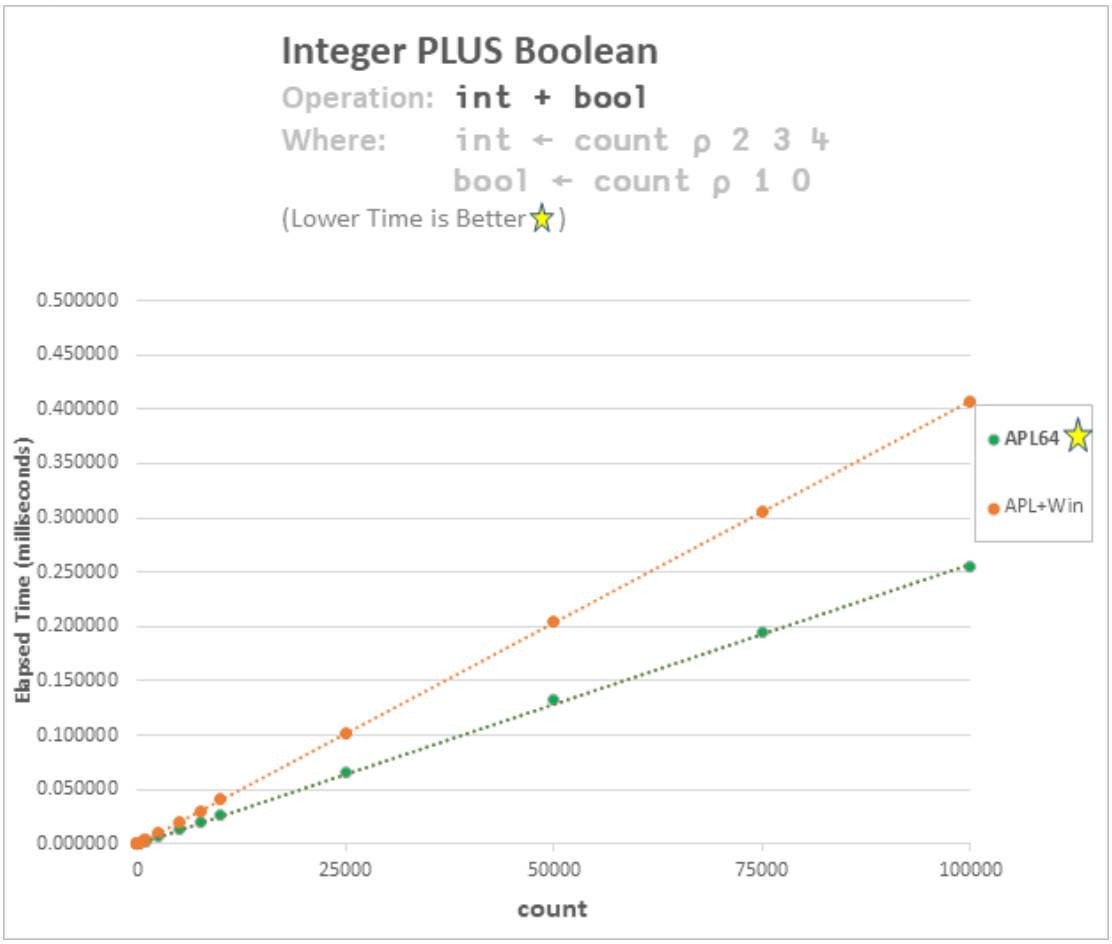
Vector Greater than Vector



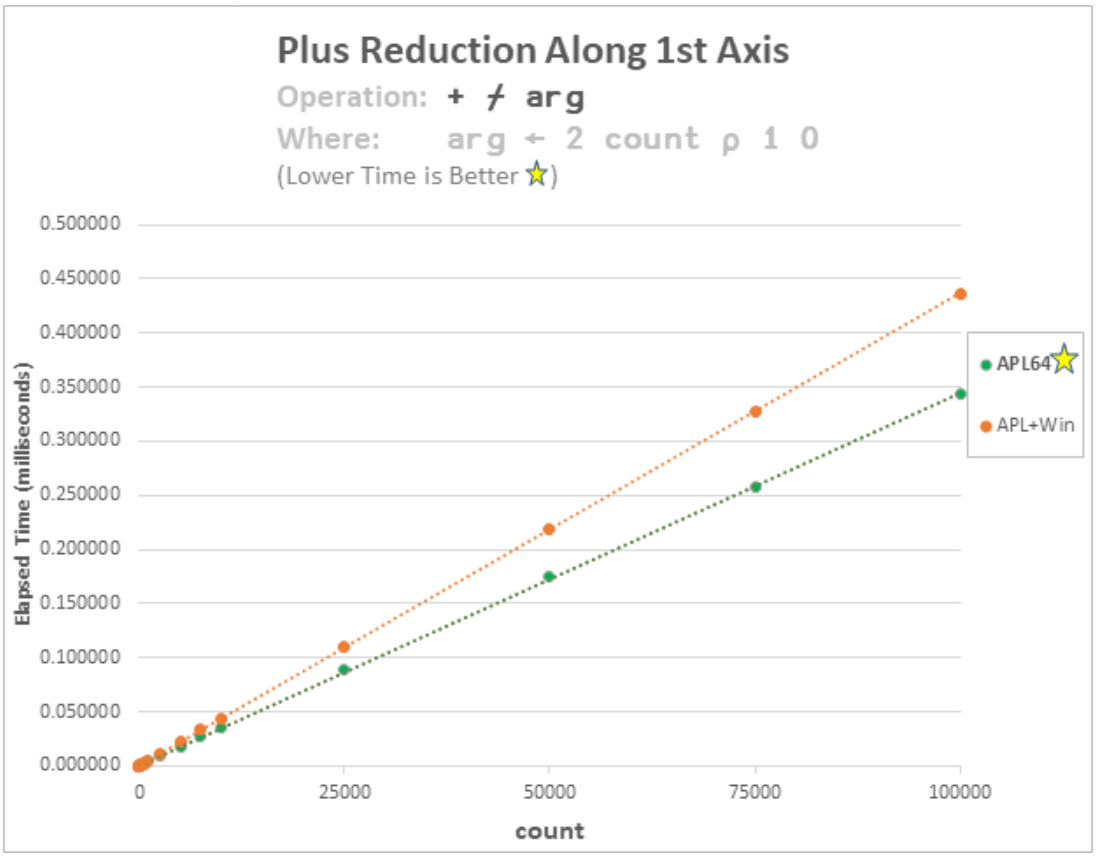
Vector Greater than Scalar



Integer PLUS Boolean

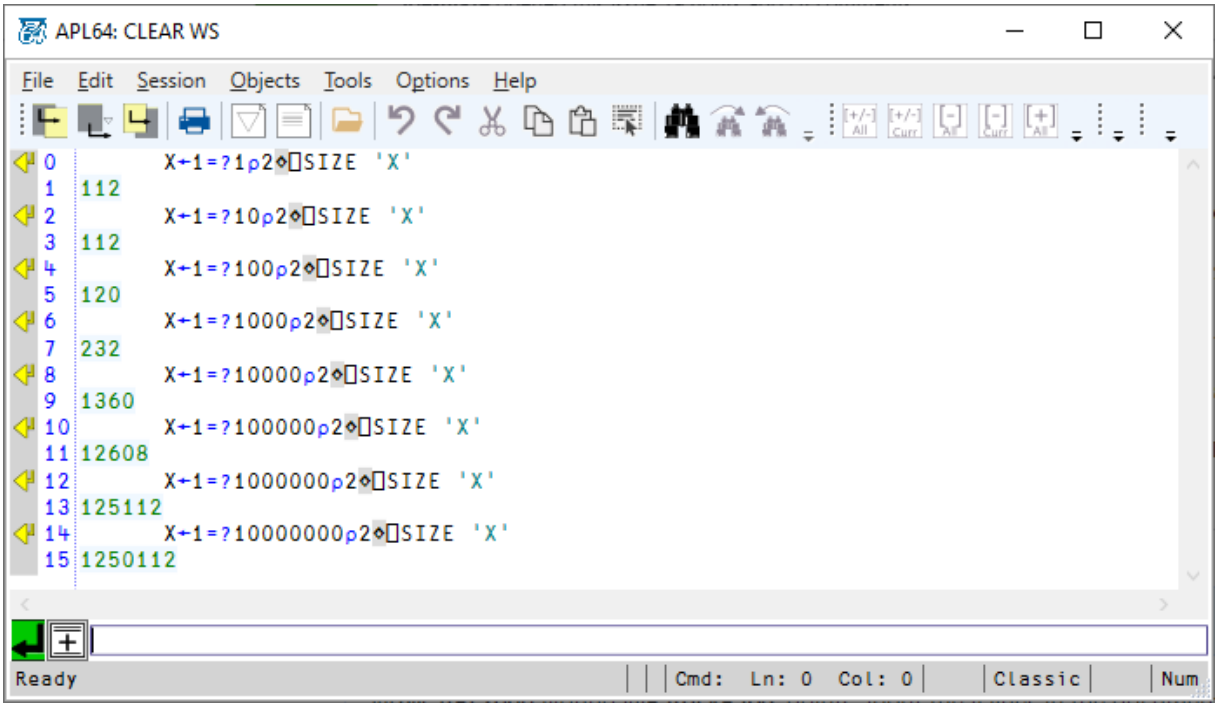


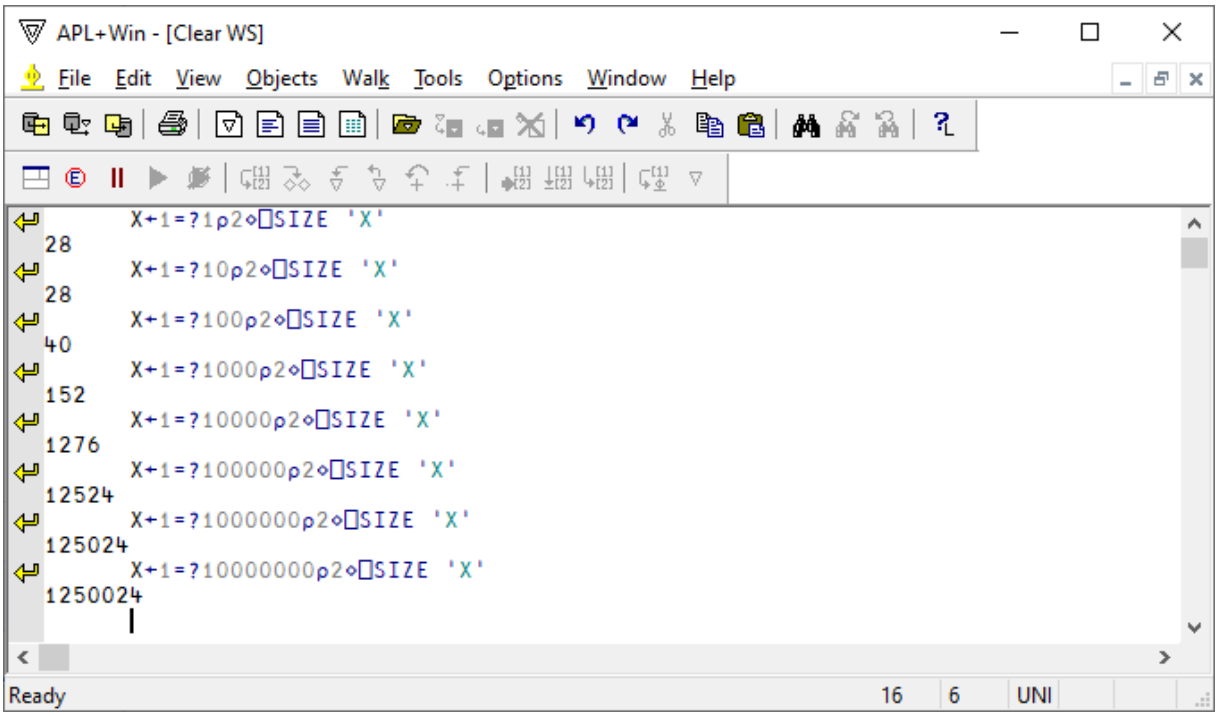
Plus Reduction Along 1st Axis



Memory Requirements for Boolean Arrays in APL64 Reduced

For Boolean data types, the storage requirements and operation performance have been significantly improved. The memory requirements for Boolean arrays in APL64 are now analogous to the requirements in APL+Win:

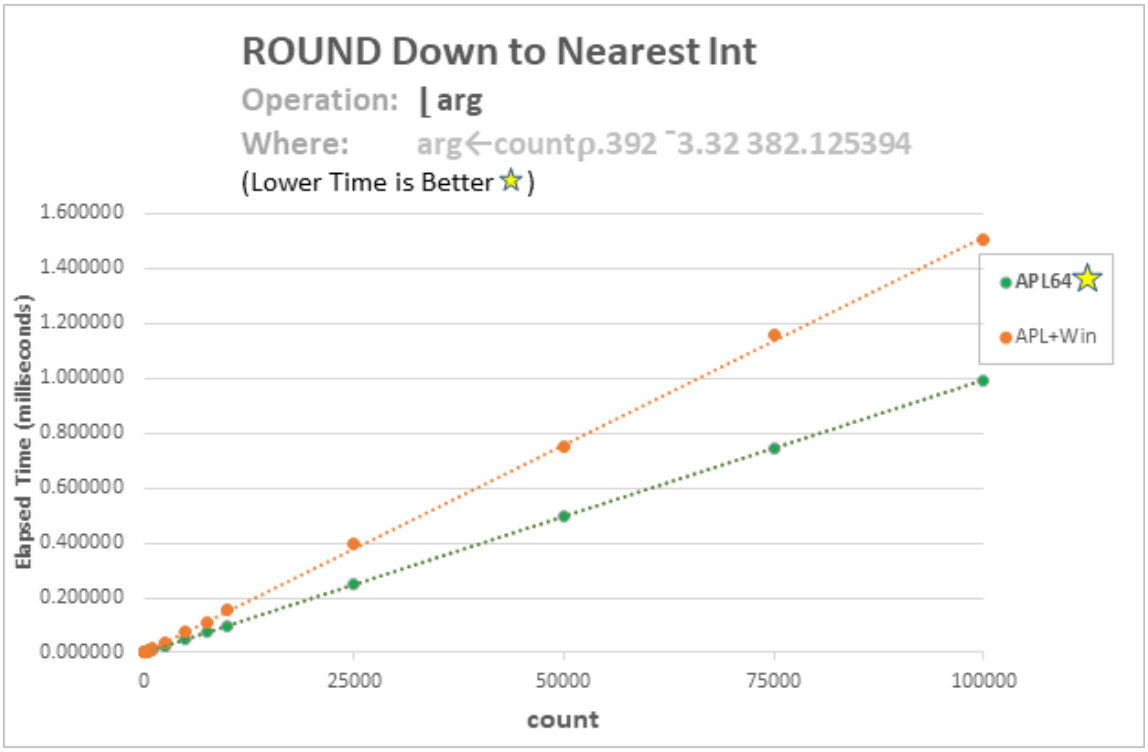


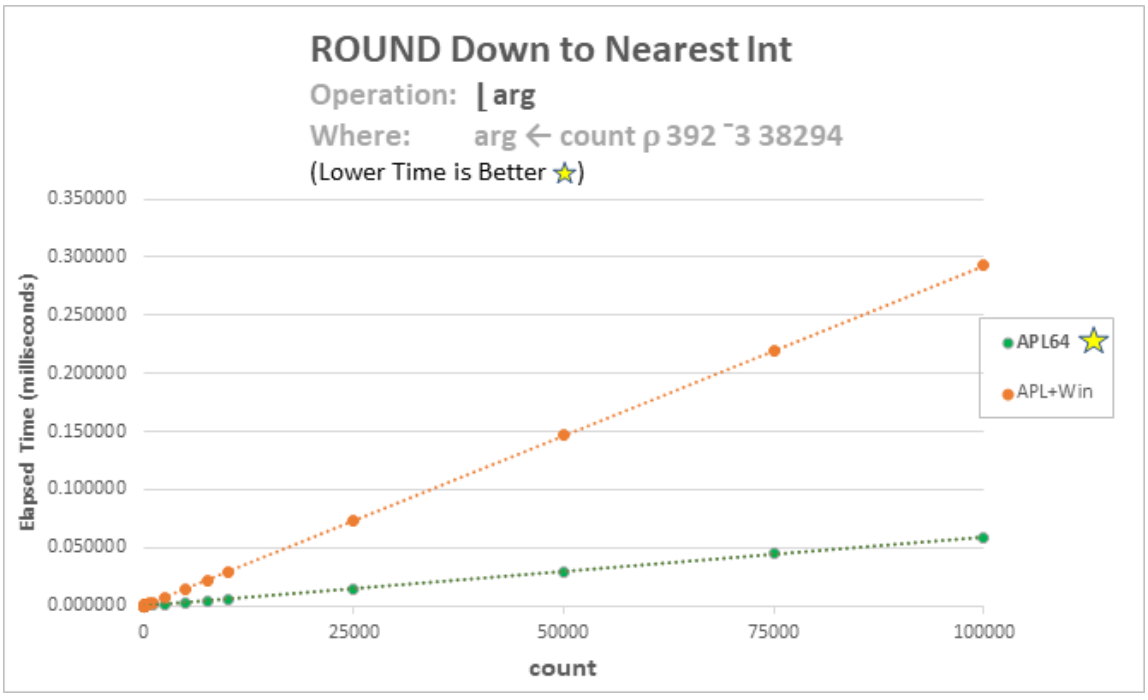


Performance Improvements for Several Primitive Functions

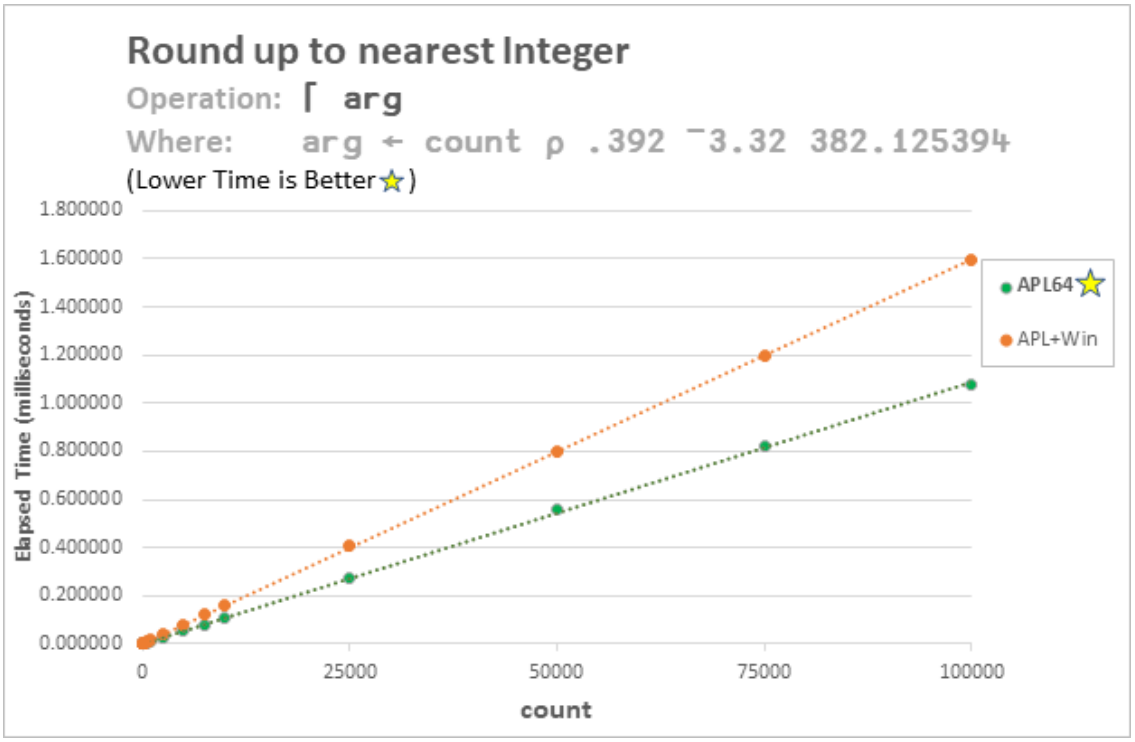
Below are graphs with comparisons between APL64 and APL+Win for the primitive functions Floor, Ceiling, Index, and Ravel:

Floor primitive function

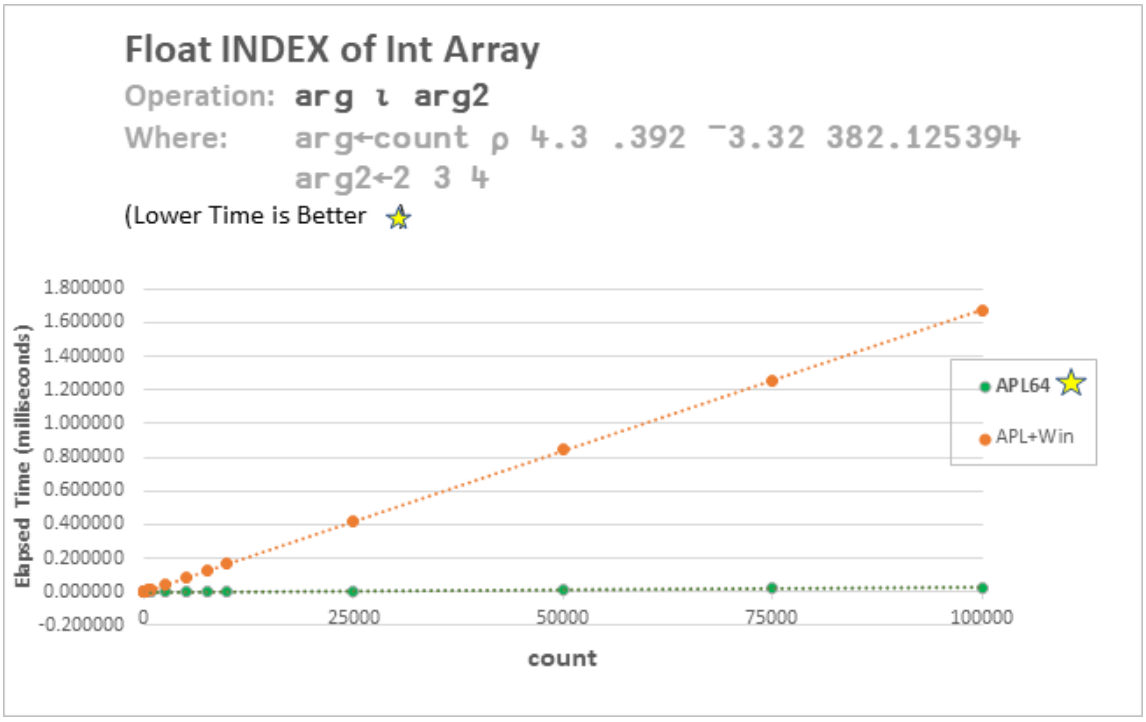




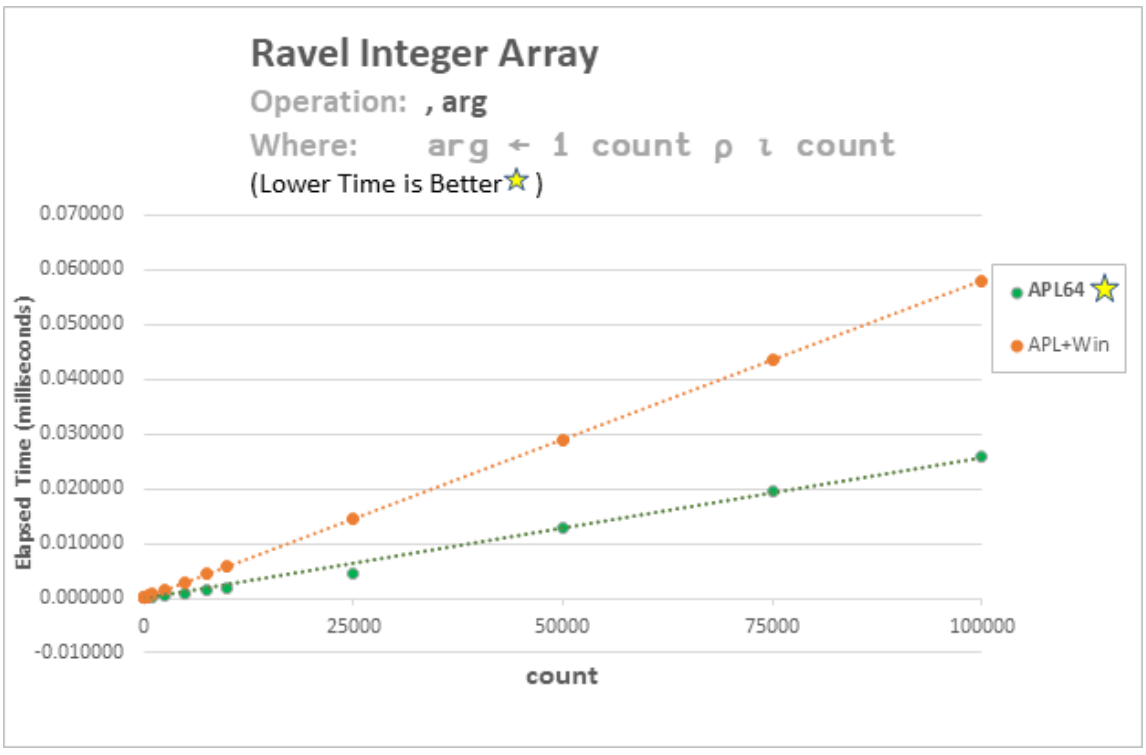
Ceiling Primitive Function

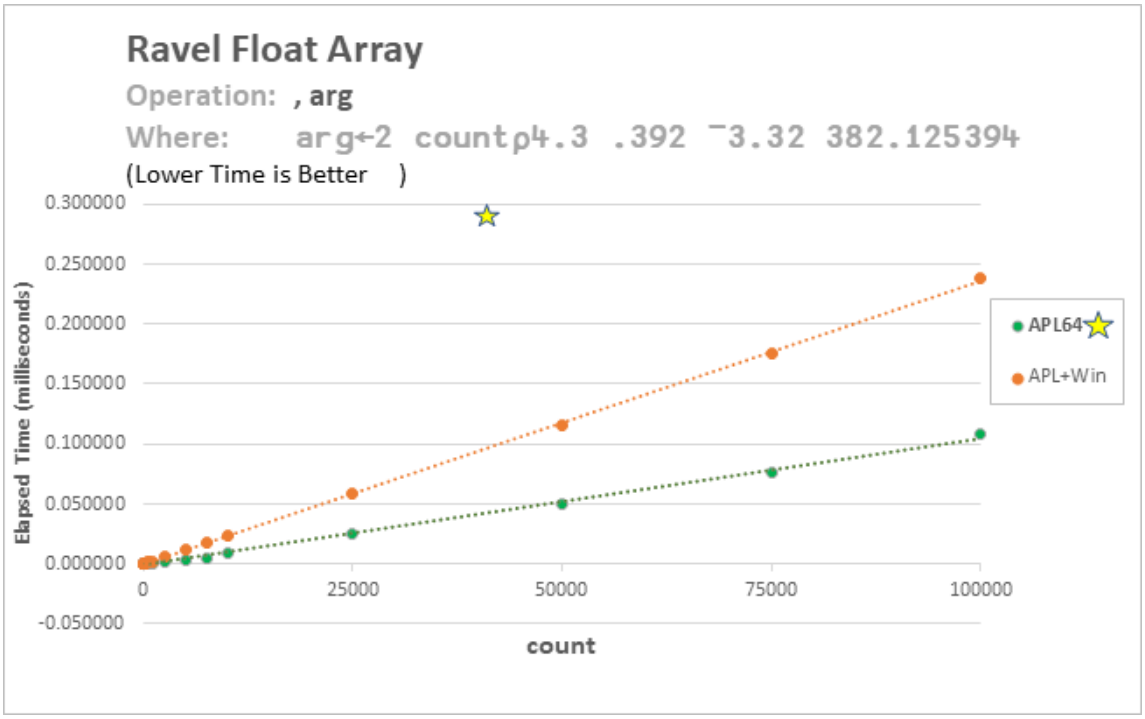


Index primitive function



Ravel primitive function





Grouped and Separated History Formats Render Results in Document or Row Style

The history format for the APL64 Developer version may be selected using the menu *Session | History Format*. The grouped and separated history formats render the APL executable and APL result statements separately. For the grouped and separated history formats, the APL executable statements are rendered in row style. For the grouped and separated history formats, the APL result statements may be rendered in document or row style.

Refer to the menu *Help | Developer Version GUI | Grouped/Separated History Document or Row Style Rendering* for additional information.

Editable Classic History Format

The classic history format now supports some editable features like in APL+Win.
Refer to the menu *Help | Developer Version GUI | Editable/Non-Editable Classic History* for additional information.

□XL Exchange Data between APL64 and Excel Worksheet without Excel.exe

Data in an APL variable can be used to fill selected rows/columns of an Excel worksheet. Data in selected rows/columns of an Excel worksheet can be used to create an APL variable. The □XL APL64 system function does not require the use of Excel.exe on the target workstation.

Syntax: Result ← □XL Action [ActionArg1] …

Actions (not case-sensitive):

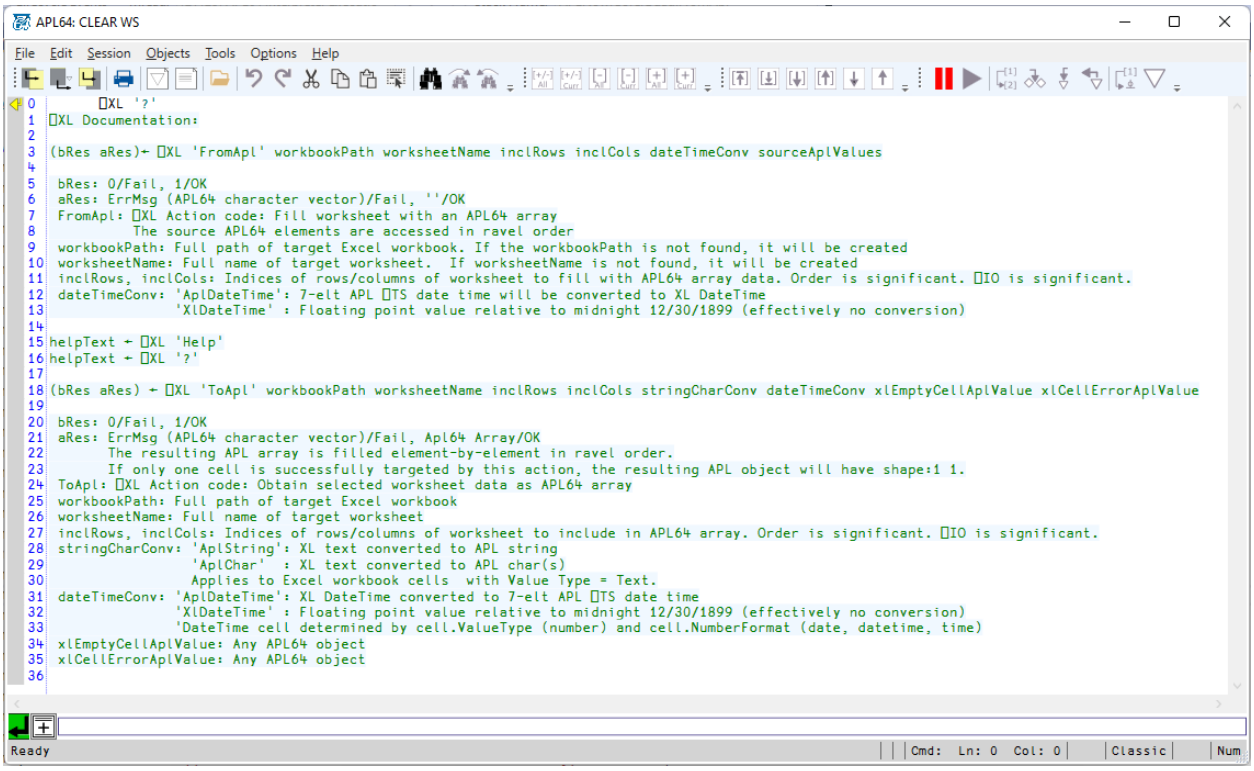
Action	Description	Reqd. Action Args.
'?'	□XL Detailed syntax	None

‘Help’	⎕XL Detailed syntax	None
‘ToAPL’	Create APL variable from selected worksheet rows/columns	1, 2, 3, 4, 5, 6, 7, 8
‘FromAPL’	File selected rows/columns of worksheet from APL variable	1, 2, 3, 4, 6, 9

Action Arguments:

#	Action Argument Name	Case-Sensitive	Options	Applicable Action
1	workbookPath	OS-dependent	Full path to XL workbook	‘ToApl’, ‘FromApl’
2	worksheetName	N	Name of target worksheet	‘ToApl’, ‘FromApl’
3	inclRows	N/A	Integer vector	‘ToApl’, ‘FromApl’
4	inclCols	N/A	Integer vector	‘ToApl’, ‘FromApl’
5	stringCharConv	N	‘AplString’, ‘AplChar’	‘ToApl’
6	dateTimeConv	N	‘AplDateTime’, ‘XLDateTime’	‘ToApl’, ‘FromApl’
7	xlEmptyCellAplValue	N/A	APL variable	‘ToApl’
8	xlCellErrorAplValue	N/A	APL variable	‘ToApl’
9	sourceAplValues	N/A	APL variable	‘FromApl’

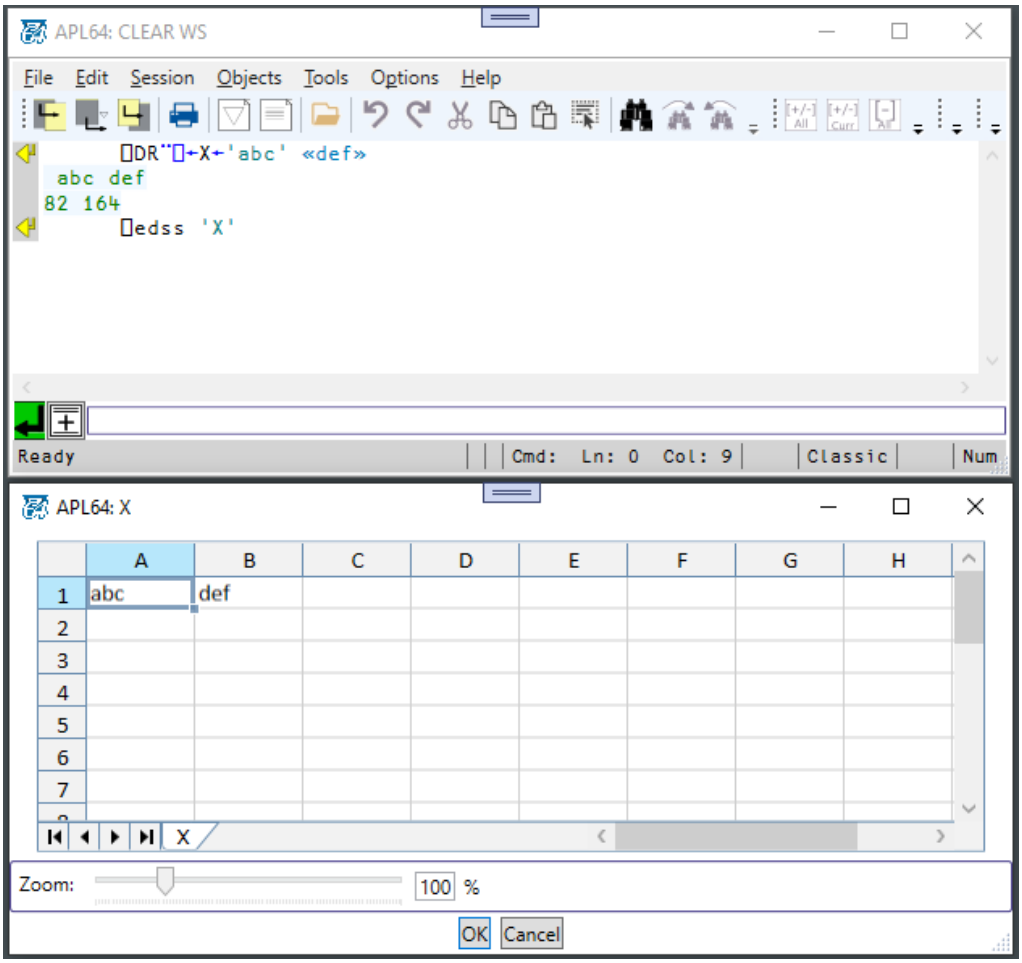
‘?’ or ‘Help’: Provides detailed syntax text in the form of an APL character vector:



⎕EDSS Edit an APL Array in a Worksheet Editor

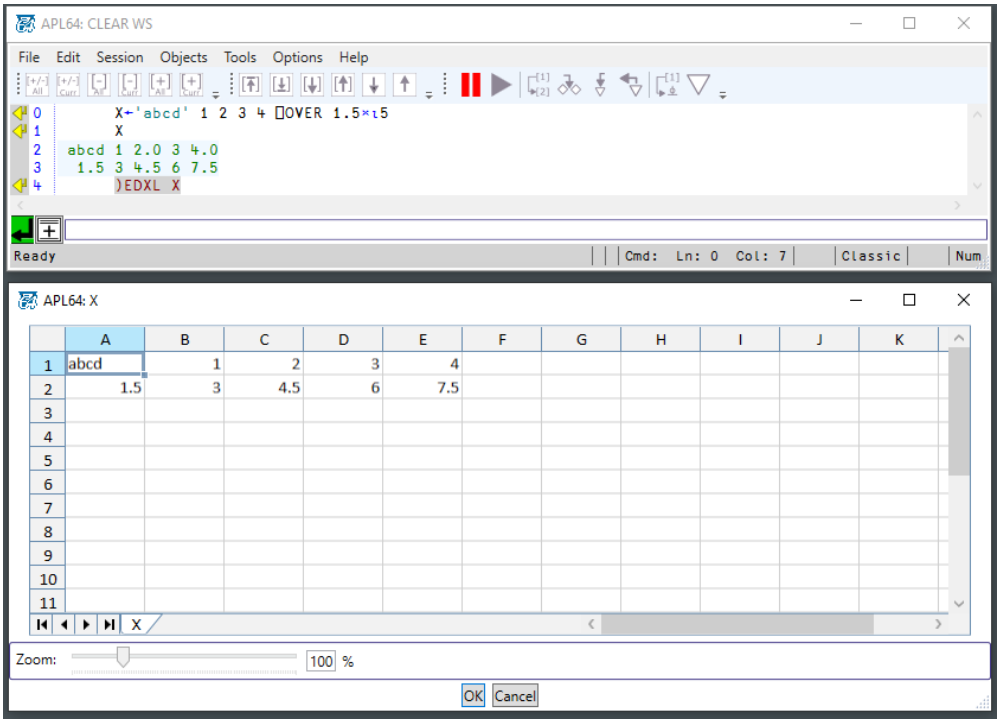
⎕edss is a new system function for editing an APL array of rank 2 or less in a graphical worksheet editor. Data in an APL variable can be used to fill selected rows/columns of a worksheet. Data in selected rows/columns of a worksheet can be used to create/update an APL variable. A ⎕edss editor is not modal, so multiple ⎕edss editors may be created and data may be pasted between a ⎕edss editor and

other areas of the APL64 developer GUI including other ⎕edss editors, the history pane and the command line as well as other Windows applications.



)EDSS Edit an APL Array in a Worksheet Editor

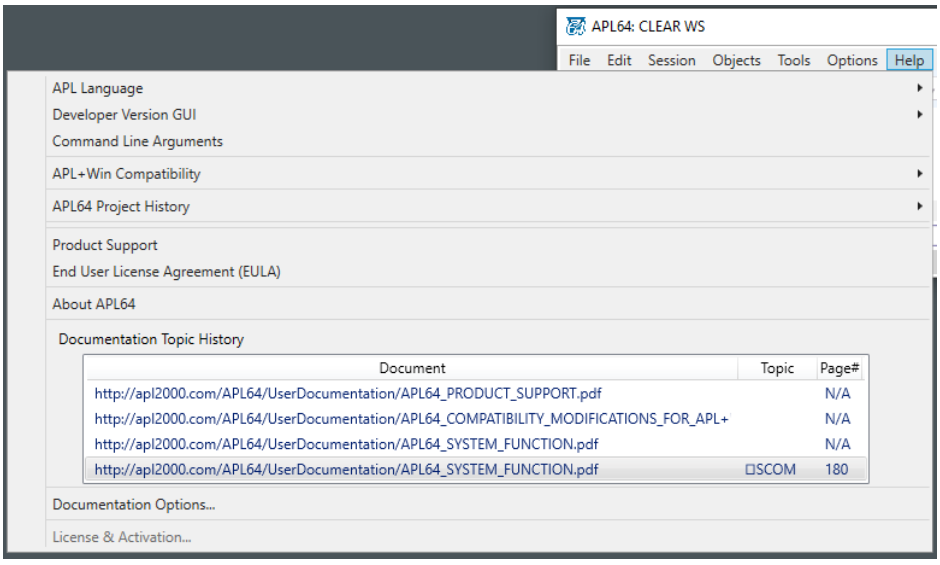
)edss is a new system command for editing an APL array of rank 2 or less in a graphical worksheet editor. Data in an APL variable can be used to fill selected rows/columns of a worksheet. Data in selected rows/columns of a worksheet can be used to create/update an APL variable.



Microsoft WebView2 Component No Longer Required

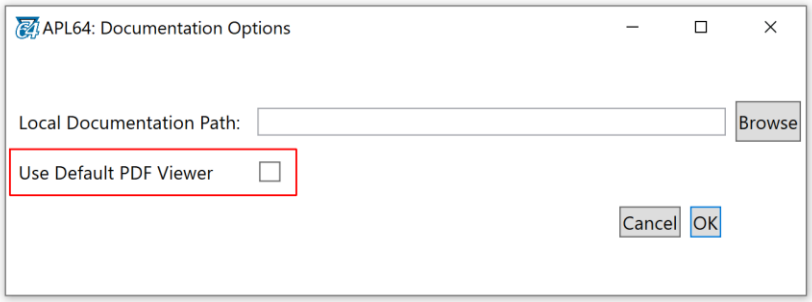
The Microsoft WebView2 component for displaying the APL64 online documentation is no longer required by APL64. The user-selected pdf-format document viewer will now be used by APL64. Users may uninstall Microsoft WebView2 provided it is not being used by another program on the machine.

The Documentation Topic History during an APL64 developer version instance is now available in the Help menu item:



Help | Documentation Options | Use Default PDF Viewer check box eliminated

The option that was removed is identified by the red rectangle in the figure below:



Drop Workspace Confirmation Dialogue Box

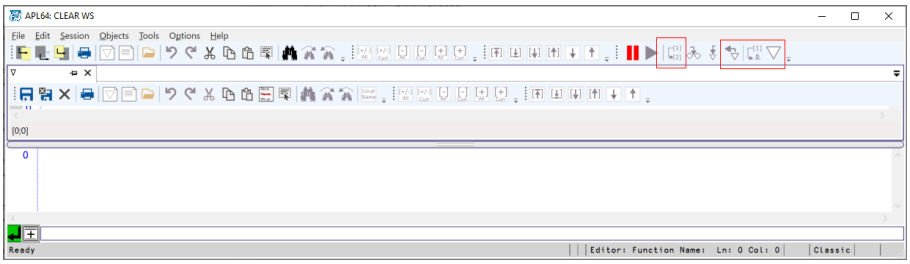
A confirmation dialogue now displays when a workspace is dropped from the menu: *File | Drop*

Description of Session History's context menu item 'Delete' has been clarified

The tooltip text for context menu item 'Delete' states:

Copies the current History row text.
If there is no selection in the text, deletes the character after the caret offset from the text. Inserts the text into the Command Line at the Command Line caret location.

Debugger icons have been added to Main Window toolbar

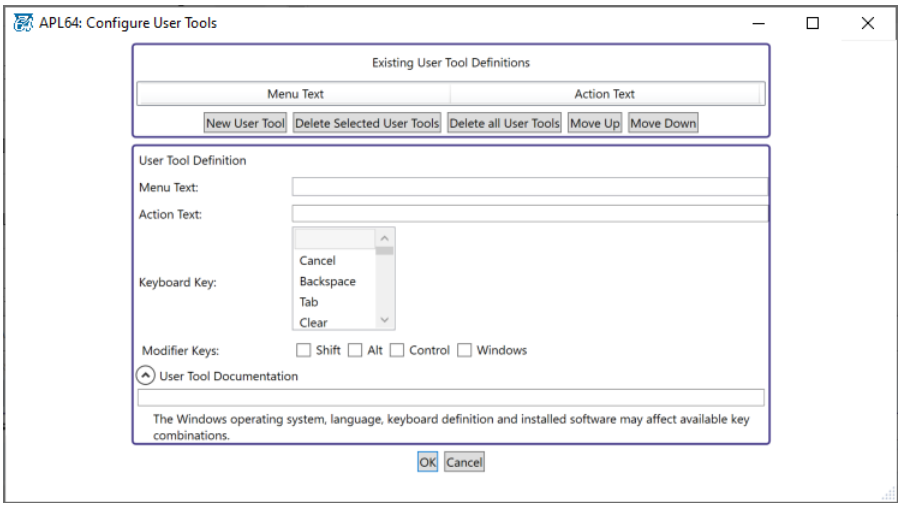


- Step to next LINE
- Step OUT of function
- Copy pending code to APL
- Edit Pending Function

Note: The following debugger icons were unsupported and therefore removed:

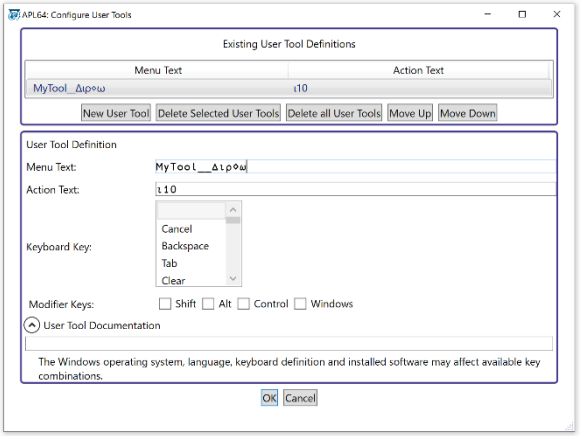
- Step by primitive OVER function
- Step by primitive INTO function

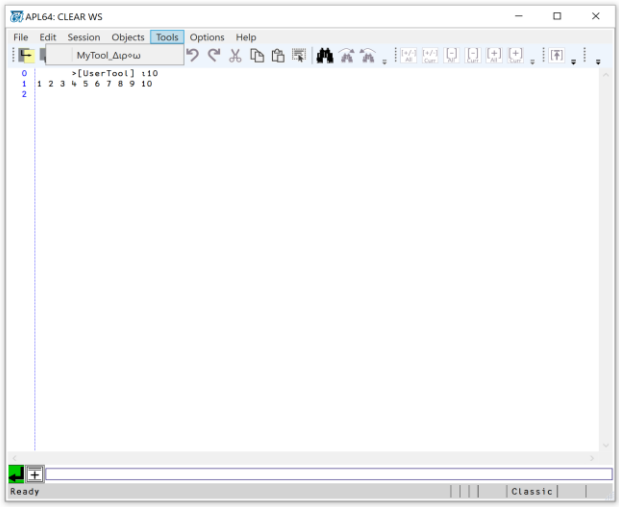
Repositioned OK and Cancel buttons to bottom center in Configure User Tools Dialogue.



Support for APL glyphs in Menu Text in Configure User Tools dialogue

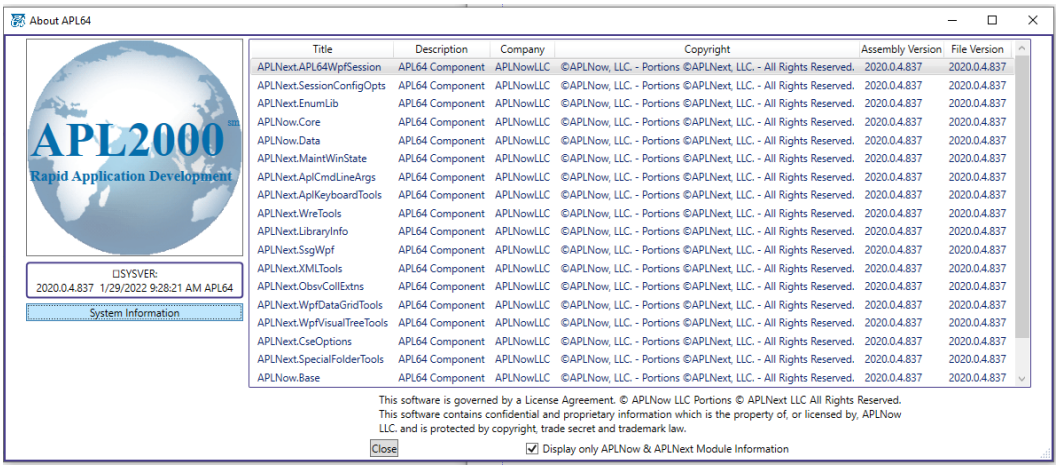
APL glyphs can be entered in the Menu Text field in the Configure User Tools dialogue. The APL glyphs will also be displayed in the Tools menu.

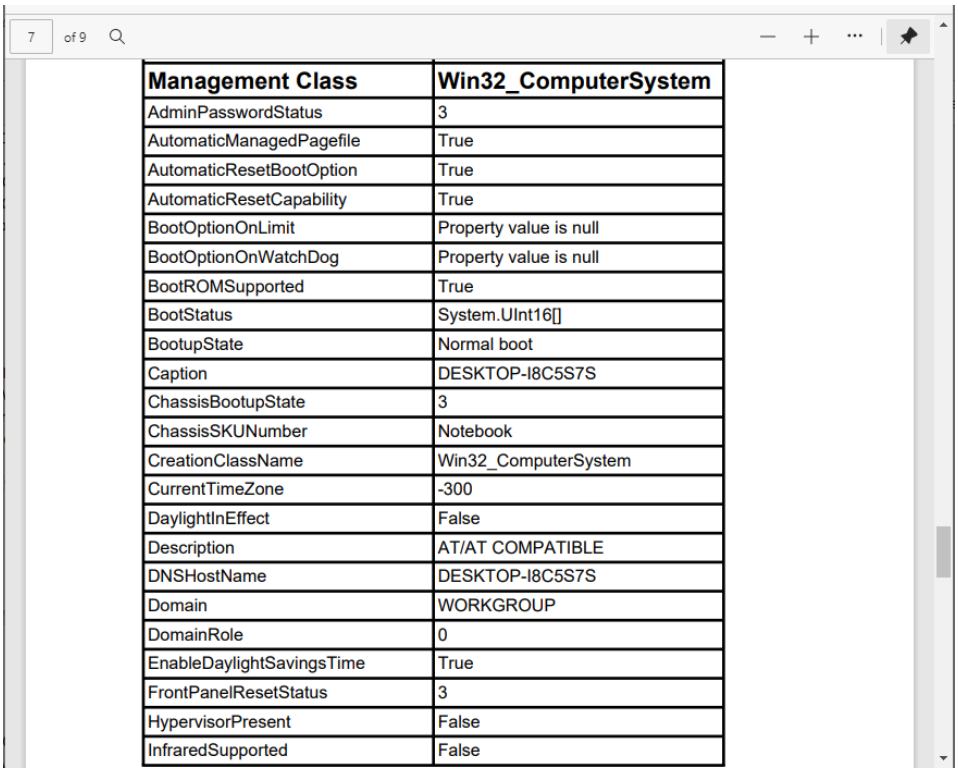




Added document to APL64 Developer version: Help | About APL64 | System Information

The Win32_ComputerSystem class is represented in the help documentation. When reporting APL64 issues to support@apl2000.com, this document should be included.





Management Class	Win32_ComputerSystem
AdminPasswordStatus	3
AutomaticManagedPagefile	True
AutomaticResetBootOption	True
AutomaticResetCapability	True
BootOptionOnLimit	Property value is null
BootOptionOnWatchDog	Property value is null
BootROMSupported	True
BootStatus	System.UInt16[]
BootupState	Normal boot
Caption	DESKTOP-I8C5S7S
ChassisBootupState	3
ChassisSKUNumber	Notebook
CreationClassName	Win32_ComputerSystem
CurrentTimeZone	-300
DaylightInEffect	False
Description	AT/AT COMPATIBLE
DNSHostName	DESKTOP-I8C5S7S
Domain	WORKGROUP
DomainRole	0
EnableDaylightSavingsTime	True
FrontPanelResetStatus	3
HypervisorPresent	False
InfraredSupported	False

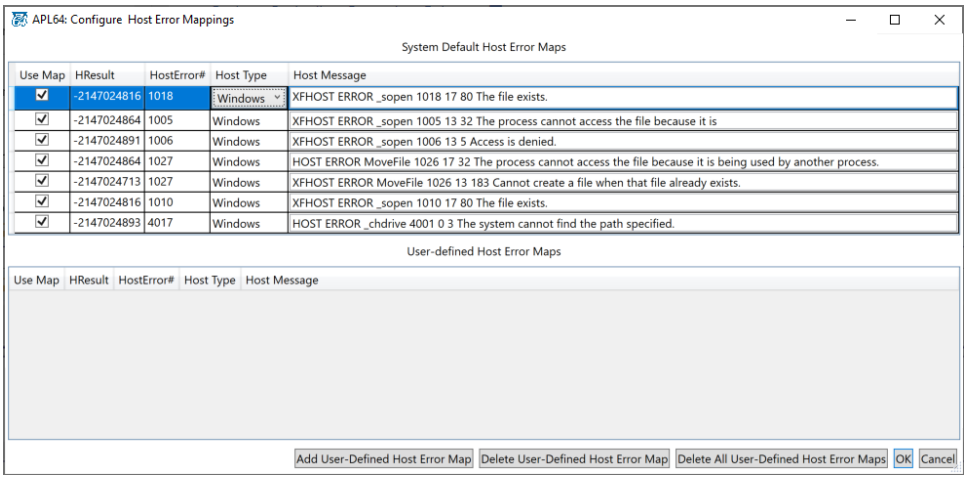
Execute symbol has been added to the debugger

The execute symbol appears in the Run, Step INTO function and Step to next STATEMENT toolbar debugger icons when the focus is in a non-empty session command line.

Context menu behavior for row-style history formats is analogous to the Classic history format.

Right-click in the history pane is used to present the context menu and any context menu selection is made or Esc is clicked, the focus will move to the TextArea within the Datagrid cell on which the mouse was right clicked.

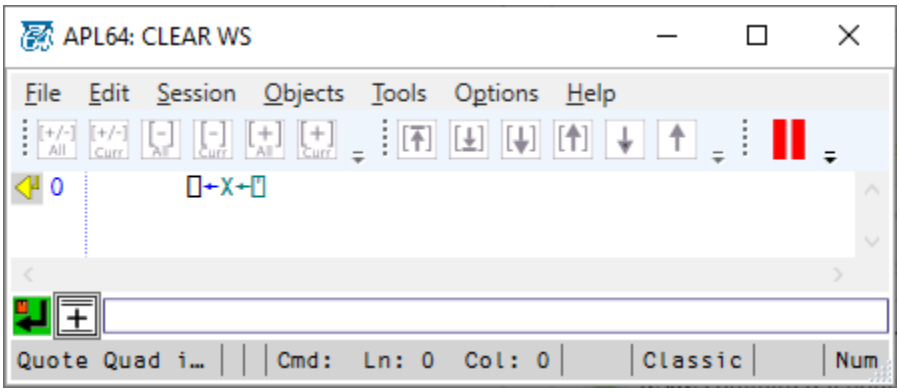
Improved readability of Configure Host Error Mappings Dialogue

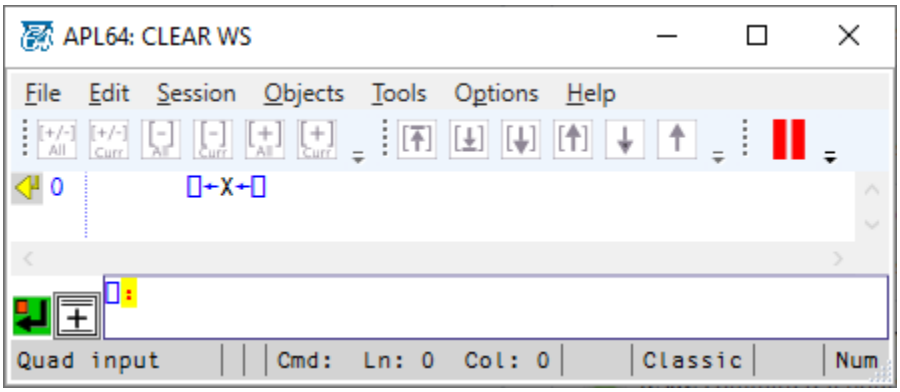


Scroll bars are not affected when the Session | View Scale APL Text property value is user-modified for the Classic and Row-style history formats.

For the Row-style history formats, this modification makes modification of the View Scale APL Text property value machine intensive.

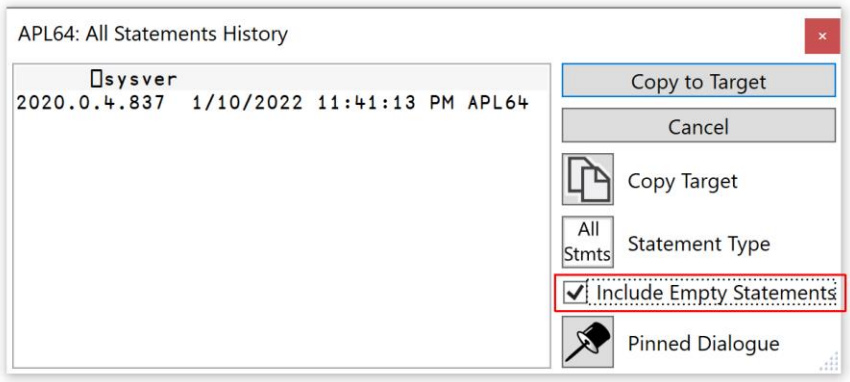
The interpreter state glyphs for quad and quote quad input have been improved





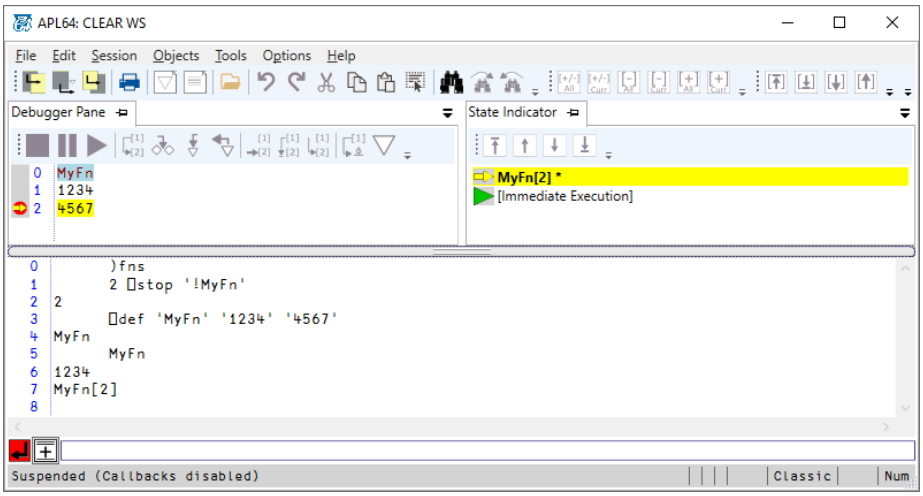
Include Empty Statements option added to Executed Statements History

When the Include Empty Statements option is enabled, the execution history dialogue displays blank rowed APL executable or results statements. The option setting is unchecked by default to not include the blank rows. The option is identified in the red rectangle in the figure below:



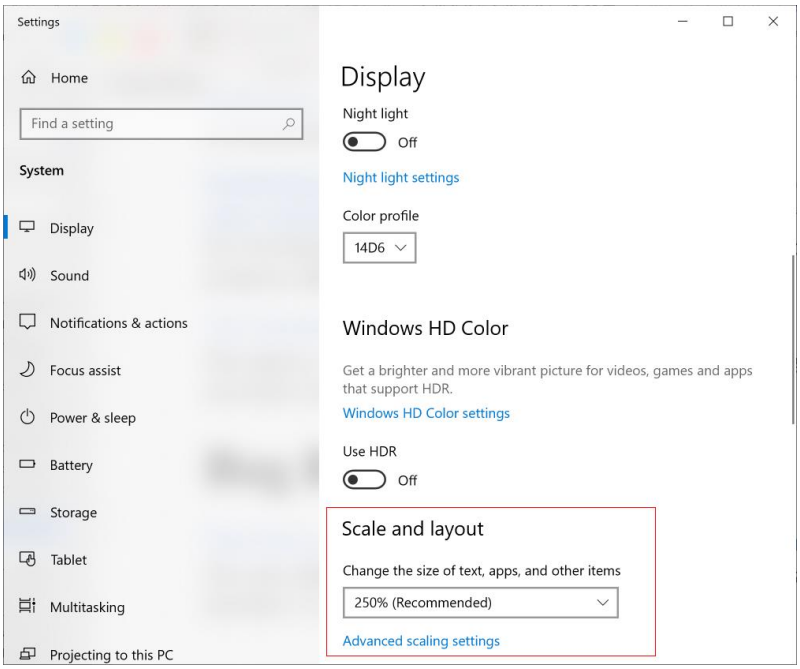
Added latent ⎕STOP settings

When creating a function stop using ⎕stop, the name of an unlocked function may be prefixed with an exclamation mark (!) to indicate a latent stop. The latent stop will be effective when the named function is defined, such as defining it from a file, using ⎕def, ⎕defl or ⎕fx.



The menu *Session | Scale Entire GUI* has been deprecated

The option conflicts with the value of Windows' Display Setting 'Size of text, apps, and other items ', shown in the red rectangle in the diagram below, is not set to 100%:



Renamed menu items with *Parens* in the name to *Glyphs* in the Edit menu

The change was necessary because the menu option also enabled you to find a matching bracket ([]), single ('') and double quotes (""), and the string («») glyphs in an APL expression.

Below is the list of the changes to the Edit menu:

Edit | Match Parens -> Edit | Match Glyphs
Edit | Match Parens | Match Parens -> Edit | Match Glyphs | Match Glyphs
Edit | Match Parens | Match and Tag Parens -> Edit | Match Glyphs | Match and Tag Glyphs

Renamed Menu: Objects | Function Editor | Sort Local Variables

This is renamed to *Objects | Function Editor | Sort Local Variables Now*. When selected, the local variables in the function header will be sorted.

Reordered menu items in Session | Grouped History Format:

- Switch Position of Vertical Scroll Bar and Row Headers in Results Grid
- Scroll to Last Result Statement
- Maximum Height of Results Grid

New menu options are provided for displaying the first or last result in history format

Below are the separate menu settings to select if the last or first result statement will be visible when the results pane is updated for the separated and grouped history formats:

- Session | Grouped History Format
- Session | Separated Session History Format

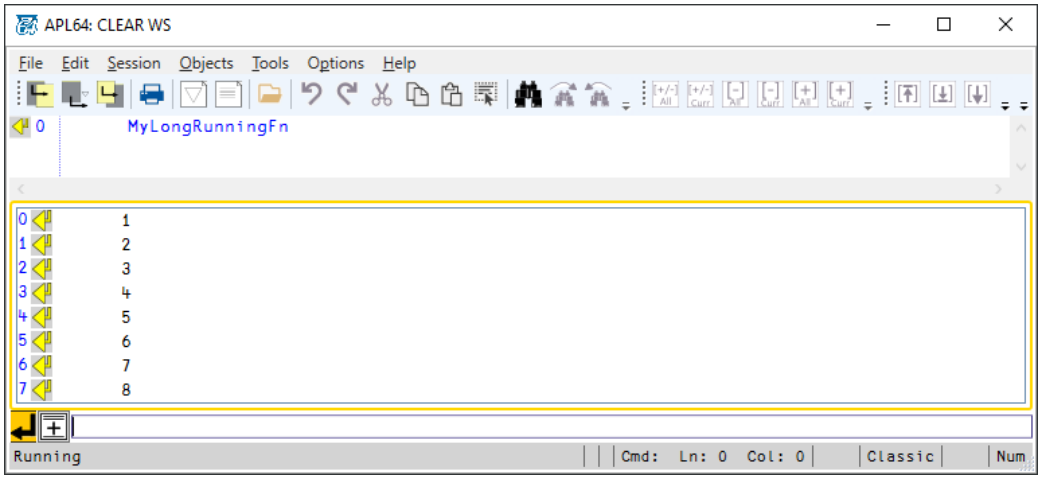
New Menu: Objects | Function Editor | Automatically Sort Local Variables

If enabled, when adding local variables local variables, the locals list in the function header will be sorted.

New Menu: Session | Display Pending APL Executable Statements

APL executable statements which have been user-entered but not yet executed are pending APL executable statements. While the current APL executable statement is processed by the APL64 interpreter, subsequently entered APL executable statements are pending execution and can be transiently displayed in the pending statements list.

When this checkbox is checked, a pending APL executable statement will be displayed in the pending statements list until execution of it commences. This option can be useful when additional APL executable statements have been entered while a long-running APL executable statement is being processed by the APL64 interpreter.



After execution is commenced on an APL executable statement, it will be displayed in the history pane. The associated APL result statements, if any, will be displayed in the history pane when the APL64 interpreter emits them.

New Menu: Session | Debugger | Show Debug Toolbar on Main Window

The menu was previously available as Session | Debug Toolbar.

New Menu: Session | Debugger | Automatically Show/Hide Debugger

If this checkbox is checked, the debugger and state indicator panes will be automatically hidden when the APL64 interpreter is not suspended.

New Menu: Session | Debugger | Show Debugger

APL64 provides the Ctrl+D shortcut to Check/Uncheck the Show Debugger and State Indicator panes, if the APL64 interpreter is not suspended and the *Session | Debugger | Automatically Show/Hide Debugger* check box is unchecked.

Bug Fixes

SYNTAX ERROR: No digits following dot

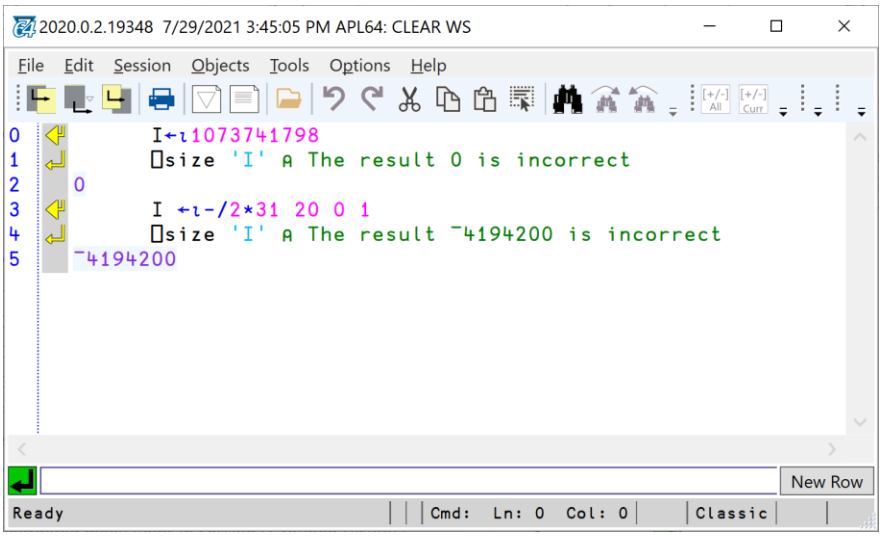
The display of the string argument following SYNTAX ERROR in the error message was an error.
e.g.

```
      ¯¯2
SYNTAX ERROR : No digits following dot
[imm] ¯¯2
       ^
```

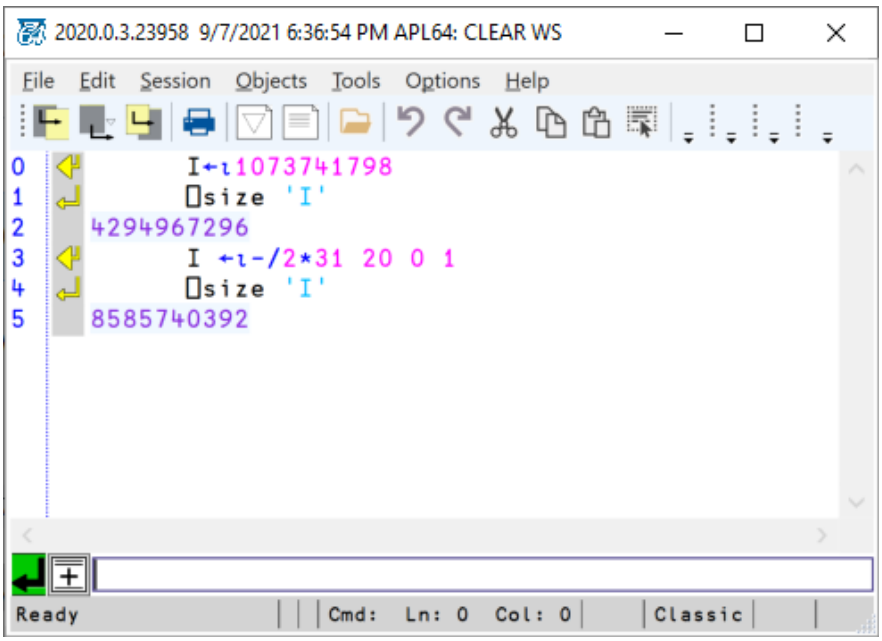
The ⎕size system function returned incorrect results for large integer arrays

Note: The fix for this appeared in the third pre-production release but wasn't documented.

Below was the incorrect result in pre-production releases 2020.0.1 and 2020.0.2:



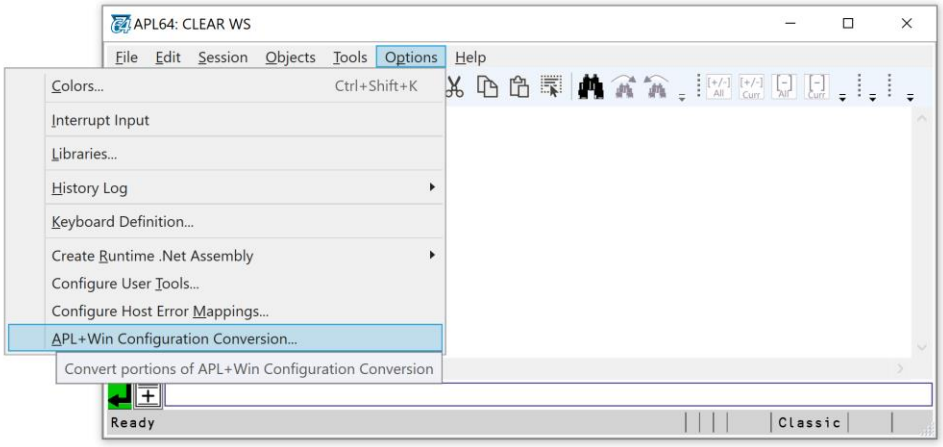
Below is the correct result in pre-production release 2020.0.3:



The Print action uses colors for printed line numbers as specified by the user

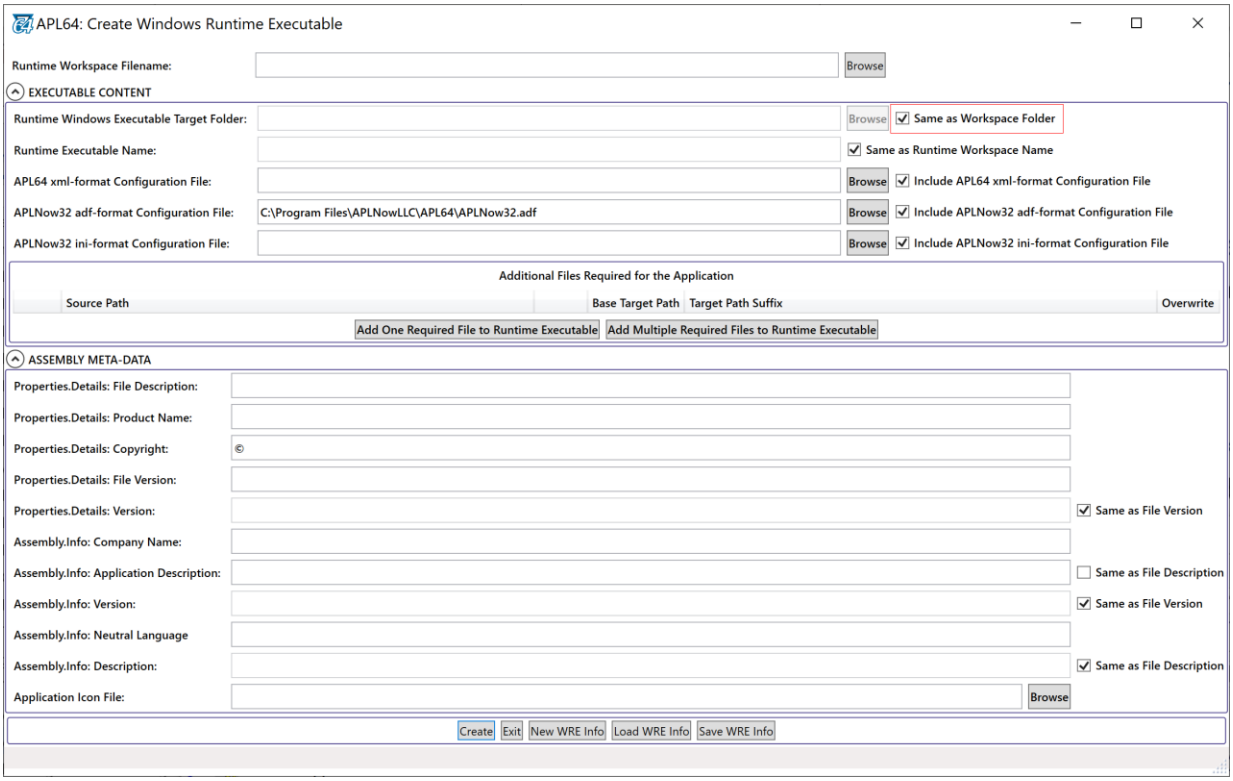
The user-selected values for foreground and background colors in Session Colors for element 'Line Numbers' is used during printing.

Added accelerator key for menu Options | APL+Win Configuration Conversion…



Create Windows Runtime Executable utility: Block using Target folder == Source Folder

The option identified by the red rectangle is removed from the WRE configuration dialogue in the latest release.



In Create WRE dialogue, the number of clicks to select Base Target Path was reduced

The number of clicks went from 3 to 2. This behavior is the expected result of the Microsoft-designed WPF Datagrid GUI control.

Binding Issues in Configure User Tools dialogue have been resolved

Text entered in User Tool Definition for Menu Text and Action Text automatically update respective fields in Existing User Tool Definitions section.

An exception is avoided when clicking the Enter key in the session history pane after clearing the history

The callback prefix displayed for the □PF system function only displays after clearing the session history

Importing of Configuration Settings now correctly updates session settings

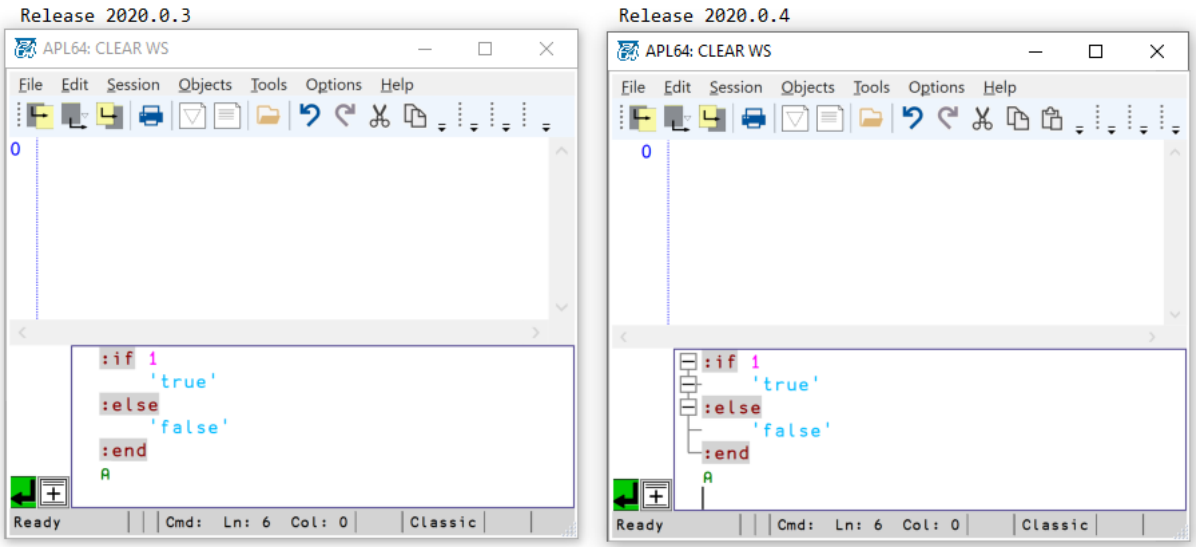
Scroll bars improperly sized when Windows | Display Settings | Scale and layout | Size of text, apps and other items is not set to 100%.

Removed Stretch property setting from Viewbox container control; this is a workaround for a Microsoft WPF bug.

Windows affected: License, About, Session Log Options, User Tool Args, Configure APL+Win Conversion, Configure Host Error Mappings, Configure User Tools, Create WRE, Define Keyboard, Edit Colors, Help Options, Import Settings, Jump to line, Libraries, Search Dialogue, Variable Editor: Variable name

Gutter Icons and Control Structure Blocks in Session Command Line

The gutter icons that typically display for a multi-line control structure block were temporarily visible in the session command line until a non-control structure statement was entered.
E.g.



Syntax color issue with previously executed system commands

There were some examples where executing a previously executed system command did not display in the session with the correct syntax colors when it involved the displaying of a dialogue box like confirming the creation of a duplicate variable name.

Syntax colors were not functional for parenthesis and brackets

Syntax coloring for matching parenthesis and bracket pairs were not available in the editors and history in the prior releases.

The APL Result Statements pane was a floatable pane

The APL Result Statements pane for the separated history format is no longer floatable to behave similarly with the other session history panes.

Two presses of the F5 key were required to resume execution of a debugged program

While suspended debugging a function and the focus was in the function or variable editors, two key presses of the F5 key were necessary to resume execution of the program.

The Edit | Copy & Cut Current Text Without Selection menu was mis-labeled

The menu has been updated to *Edit | Copy/Cut Entire Current Line Without Selection*.

The Session | Clear Current Workspace menu will no longer clear the Command Line

The menu no longer clears any commands that are in the Command line that haven't been executed.

Objects | Function Editor | Sort Local Variables was always enabled

The local variables in the header were sorted even when the menu *Objects | Function Editor | Sort Local Variables* was not enabled.

The caret was positioned in the wrong location after de-localizing the function name

When the caret was on the name of the function in the function header that was localized, un-localizing the function placed the caret where the function name appeared in the list of local objects instead of the function name.

A valuetip displayed for undefined names in the history

Valuetips were displayed as "(name) is undefined" for undefined names displayed in the history.

Possible bug with □mf applied to inner (nested) functions

The execution time reported in □mf for Inner functions were largely incorrect and/or misleading. This has been corrected in the latest release.

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 (2020.0.3.23958). We would like to thank the evaluators for their contribution.

What is the meaning of "SYNTAX ERROR: No digits following dot"

The display of the string argument following SYNTAX ERROR in the error message was an error.
e.g.

```
      ¯¯2
SYNTAX ERROR : No digits following dot
[imm] ¯¯2
       ^
```

Shouldn't the EN-UK keyboard be labeled EN-GB?

Because the keyboard layout is from Microsoft, the keyboard name will remain EN-UK.

APL+Win wraps the output in the session window and APL64 does NOT

The □pw system variable used to specify the line width for output in APL+Win needs to be user-increased beyond the default value of 80. Note the default value is 200 in APL64.

APL+Win returns the expected string data types in □cse; APL64 does NOT

There is no inconsistency in the operation of □CSE in APL64 or APL+Win. C# is a strongly typed language. The String.Format method manipulates String data types. If an argument of the String.Format method is not of the string data type, the String.Format method uses the argument's ToString() method to obtain a string value. This behavior of the C# String.Format method is the same in the APL64 CSE and the APL+Win CSE.

In your APL64 example, you provided Char[] data type arguments to the String.Format method. Therefore the String.Format method used the argument's ToString() method. The result of ToString on a Char[] object is "System.Char[]", which you probably didn't expect. For your APL64 example, you should have provided APL64 String data type arguments to the String.Format method.

APL+Win does not support the String data type, so in APL+Win the CSE 'textransfer' property was made available so the APL+Win programmer could select how an APL+Win Char data type argument would be passed to or received from the APL+Win CSE. In your APL+Win example, the texttransfer property is the default value, so Char data types are converted to strings in the APL+Win <=> CSE interface.

In APL64 since both the Char and String data types are supported, there is no need for the CSE 'texttransfer' property. This is documented in the last chapter of the APL64 CSE manual available from the APL64 Help menu item.

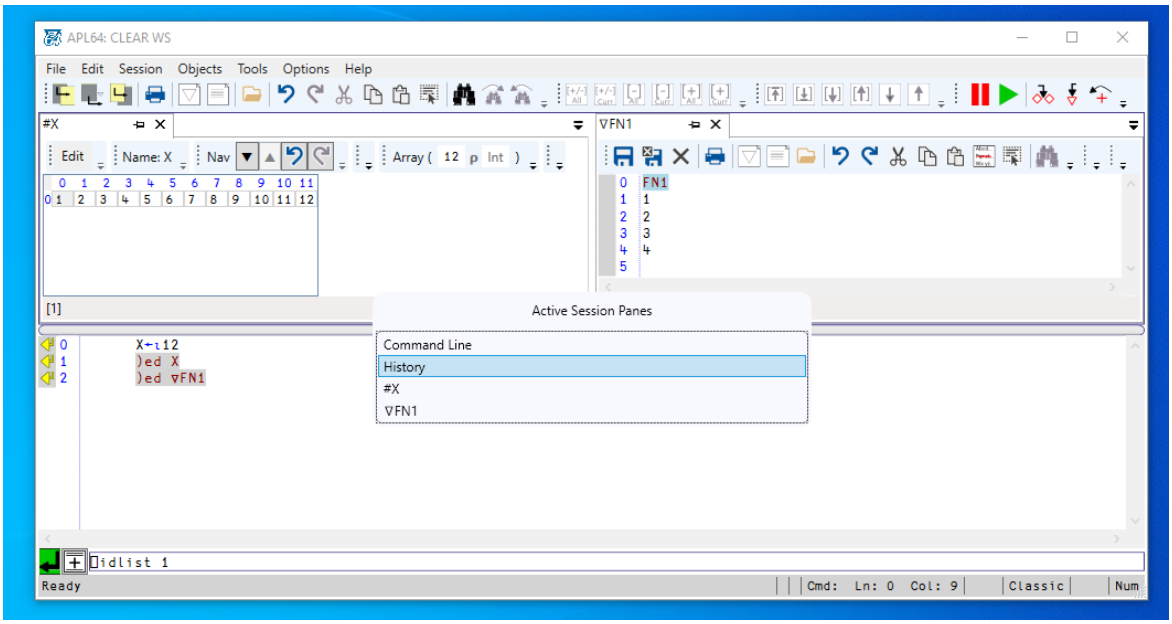
An analogous scenario applies when using a C# method which required Char data type arguments.

Should there be an APL64 Shortcuts help file?

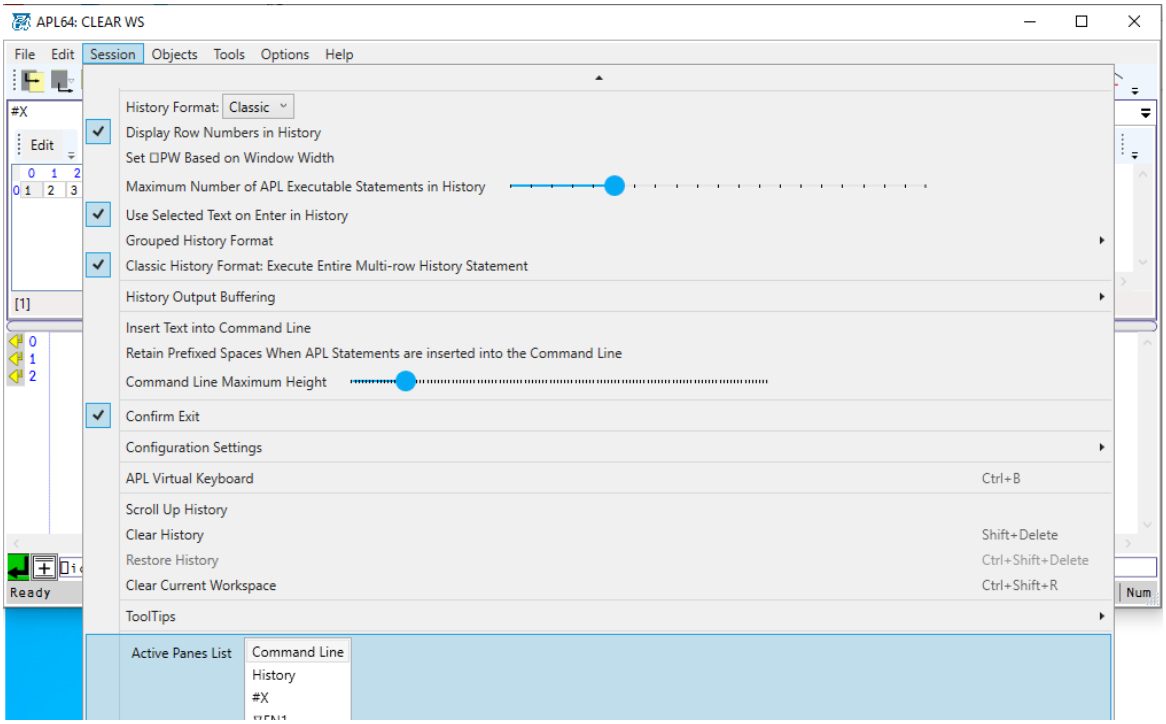
A future version of APL64 will include the new menu: *Help | Developer Version GUI | Menu Item Shortcuts*.

Is there a shortcut key to navigate to different windows in APL64?

APL64 provides the Ctrl+Tab shortcut to move the focus to any available pane in the APL64 developer version GUI:



The Session | Active Panes list is also available for this purpose:



Migrating APL+Win APLW.WSEngine to perform application calculations in APL64

For an application with a GUI implemented in .Net/C#/System.Windows.Forms application which currently uses the APLW.WSEngine to perform the application calculations, what are options to transition to APL64 for those calculations?

The design of APL64 incorporates a special runtime .Net assembly component for this purpose. This component may be used by an APL64 programmer to combine workspaces, native and component files and the APL64 interpreter engine into a single .Net Core assembly (.dll) file. Any number of such runtime .Net assembly components may be developed by an APL64 programmer. This APL runtime .Net assembly component will expos APL64 programmer-specified APL functions as .Net methods which can be used by any .Net language.

Here are the limitations:

- Windows only elements of APL64 cannot be used, e.g. ⎕WI, ⎕WCALL
- APL64 developer version only elements of APL64 cannot be used, e.g. ⎕WRE, ⎕EDIT, ⎕EDITEVENTS, etc.
- The resulting .Net assembly is not an executable (.exe), so it must be referenced and used by a .Net executable
- The .Net executable which uses the APL64 .Net assembly must be compatible with .Net Core, .Net Standard and .Net 5, so the .Net executable should target .Net Core or .Net5.
- The APL64 programmer must appropriately consider that the .Net executable may use datatypes which are not available in APL64. APL64 datatypes are Boolean, Int32, Double, Char, String.

- If the APL64 .Net assembly is going to be used cross-platform using a file system, it is the responsibility of the APL64 programmer to appropriately define filenames and paths based on the operating system in use, e.g. case-sensitivity, path separators, extensions, etc.

Here are the steps for the APL64 programmer:

- Prepare and debug all application-specific native or component files and APL64 workspaces necessary for the APL portion of the overall application.
- Designate one APL64 workspace as the primary workspace
- In the primary workspace create one or more APL64 programmer-defined functions which will be exposed to the referencing application.
- Mark each of these functions as 'public', specify the data type of the arguments and result and add appropriate xml-format Intellisense documentation for the public APL functions. All other objects in the primary and other application-specific workspaces embedded in the APK64 runtime assembly will be 'private' and not exposed to the referencing application. No public function can include '⍙' in its function or argument names.
- In the APL64 developer version use the 'Options | Create Runtime .Net Assembly | Create Platform-independent Runtime Library' dialogue to create the APL64 dll. The result will be a single .Net assembly (.dll) targeting the current .Net version on which APL64 is based. Currently .Net Core 4.6.1 is targeted, but eventually .Net 5 will be targeted.
- In the application's .Net executable source code add a reference to the APL64 dll. as a Nuget package
- In the application's .Net executable source code the APL64 public functions will now be exposed as .Net methods and may be used like any other .Net methods.

As of Sept 2021, this feature of APL64 is under development and not yet available for testing.

Support for including negative values in the left argument in replicate to match the number of items in the right argument?

For example, the expression ¯2 3 2 / 10 20 30 results in Length Error instead of 0 0 20 20 20 30 30.

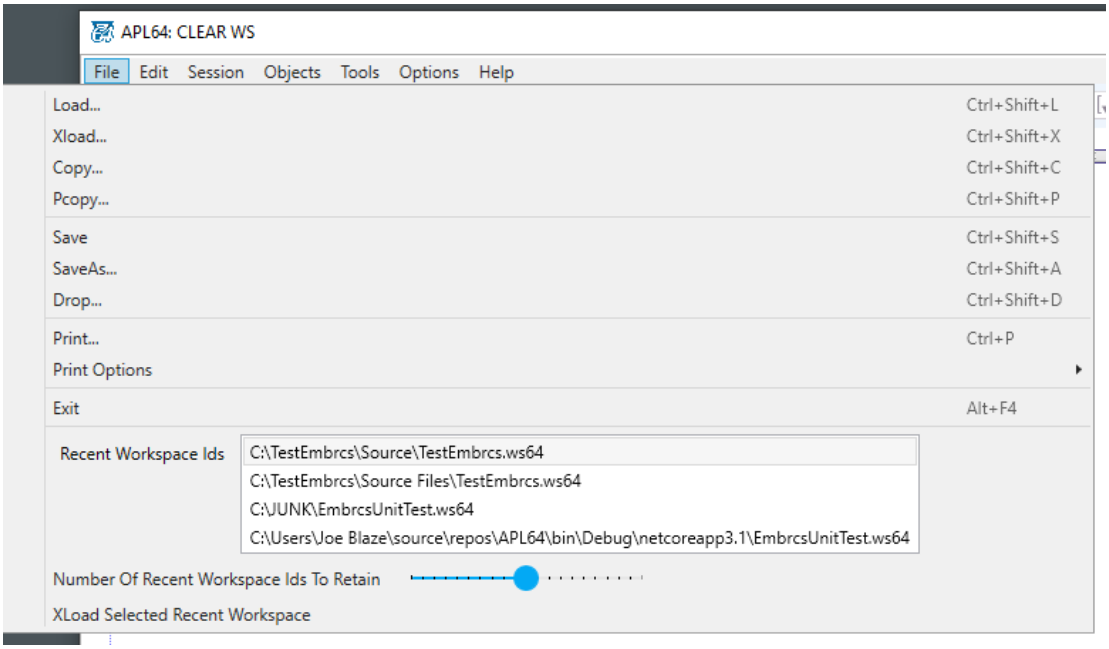
Thank you for your observation and research. Your suggestion has been recorded. If this enhancement to APL64 is deemed consistent, it is not expected to be included in the first production version of APL64, but may be included in a future production version of APL64 as time permits.

Windows 11 Support?

Testing of APL64 in Windows 11 Home and Pro has not revealed any issues.

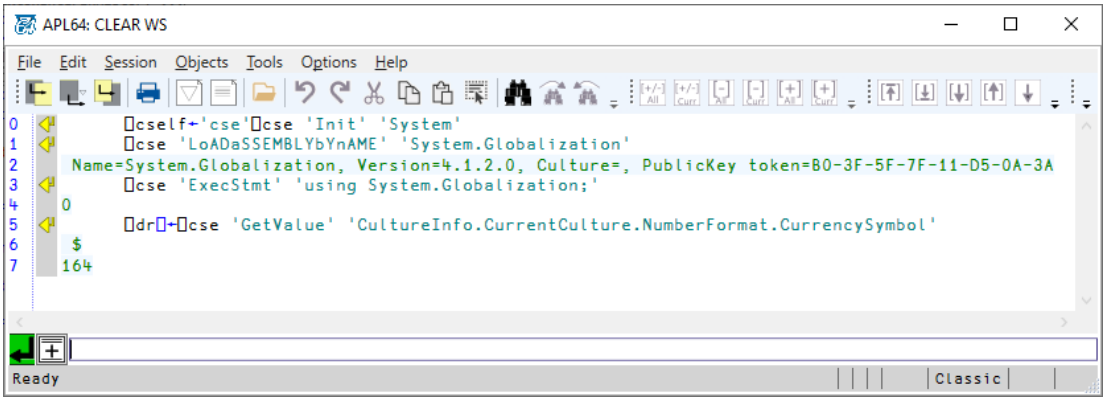
Add new menu "Options | Workspaces" for workspaces managed by the user

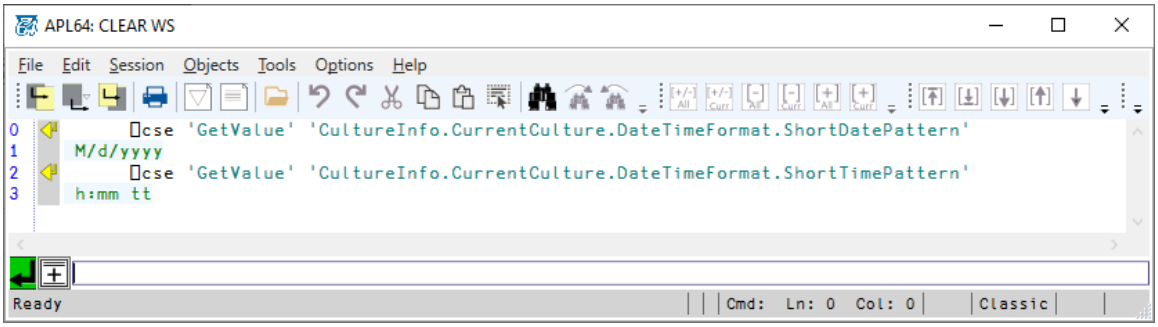
Please refer to the APL64 Developer version File | Recent Workspace Ids menu item:

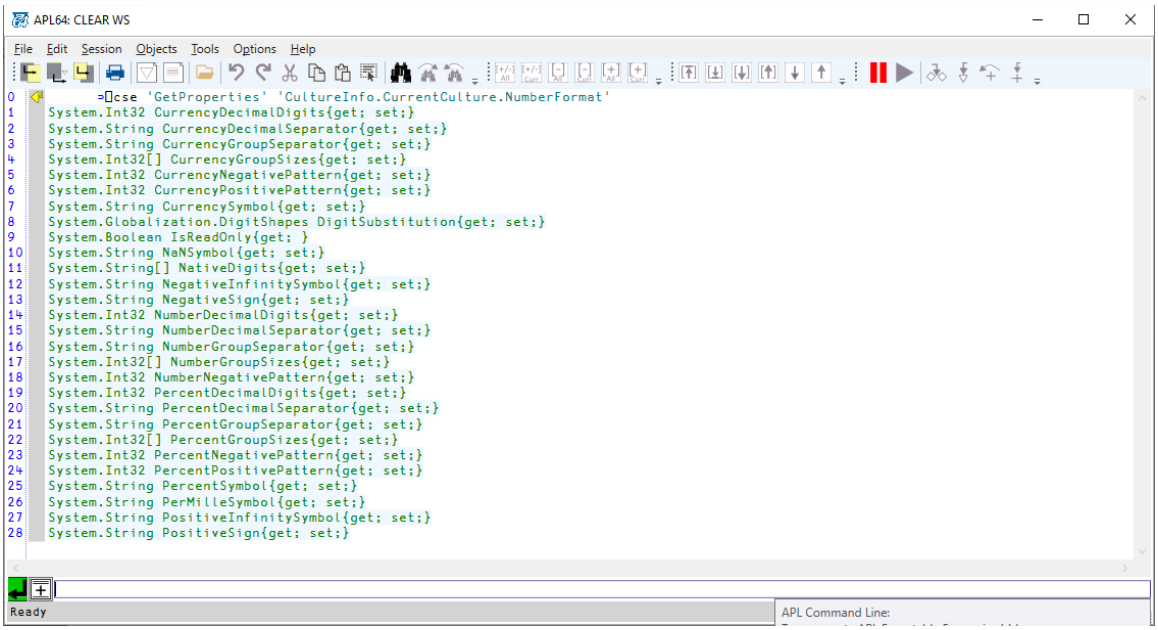


Globalization - return settings such as currency symbol, number/date format, etc.?

You can use the APL64 ⎕CSE system function to obtain this information easily, for example:



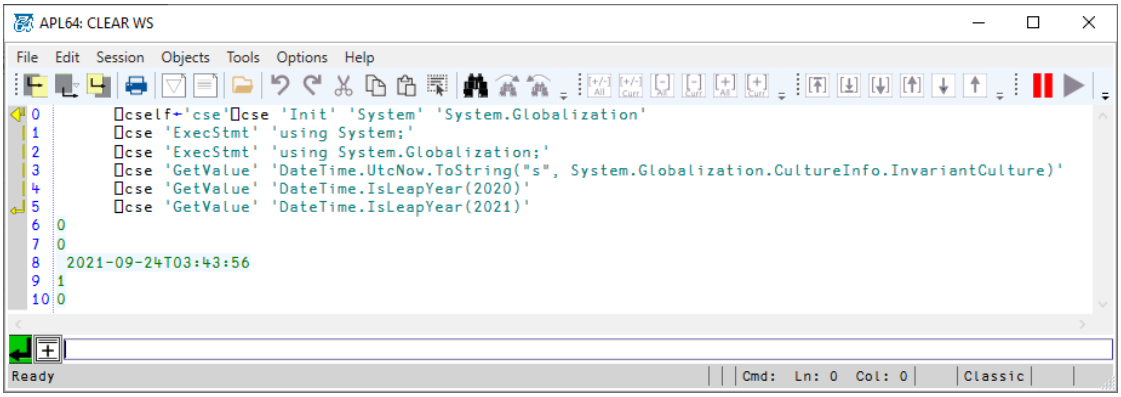




Add support for the date (in ISO–not regional—format) data type?

Just as in APL+Win it was not reasonable to duplicate all the Win32 API in APL+Win system functions, it is also not reasonable to duplicate all of the .Net Core API in APL64 system functions.

The APL64 ⍝CSE is incorporated into the APL64 interpreter (in-process), so you can easily and quickly use the APL64 ⍝CSE system function for this purpose. Some minor research in the Microsoft .Net documentation will provide all the information you are requesting. For example:



Can "Options | Libraries" facility verify that the path exists?

Restricting library definition to existing paths, would reduce the functionality of the ⍝LIBD system function. By design in APL+Win and APL64 any text may be incorporated into a library definition.

"File | Recent Workspace Ids" takes up real estate, in my opinion, unnecessarily

We don't favor any modifications to this menu currently. But are open to revisiting it if it's warranted.

The right-click (context) menu is NOT consistent throughout APL64

The order of the context menu items is currently consistent. Not all context menu items are available in all scenarios. Where applicable menu items are inserted into the applicable context menu in the same order.

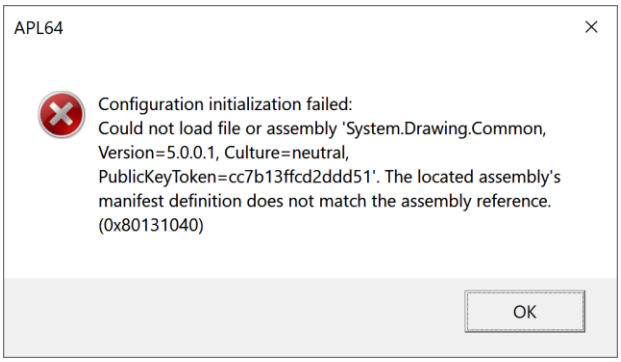
How is anyone supposed to remember the shape/number of the APL array written to native file?

⎕nsize {tie_number} returns this information.

The F9 key repeats the previous line twice in the Command window

This occurred for some users when the menu ***Session | Insert Text into Session Command Line*** was enabled. This has been resolved in this release.

Printing produced the "Configuration initialization failed;…" error message



The fix for this is in this release.

In APL64, only the first object name is preceded by a semicolon, the rest blank spaces

New in APL64 is for localized object names to be separated by a blank space. This option is configurable in APL64 with the menu option: *Objects | Function Editor | White Space Local Variables*. The option is disabled by default.

⎕DR value for Boolean data is reported using 8 bits instead of 1 bit

In prior releases of the pre-production version of APL64, extensive research was done to compare the implications of packed and unpacked APL Boolean values. Unpacking packed Boolean values to facilitate APL arithmetic calculations involving Boolean data incurs a heavy performance penalty. So, in the pre-production versions of APL64, the unpacked format for APL Boolean data was selected. This selection resulted in larger storage requirements and lesser performance for pure Boolean operations.

We are excited to report that the latest pre-production release has addressed this matter and is now storing Booleans 1 bit per element, resulting in a true compact Boolean data in APL64.

Where are User Tools stored?

User Tools are stored in the APL64 xml-format configuration file in the element labeled AplUserTools. The location of the configuration file can be queried with the APL64 menu: *Session | Configuration Settings | Settings File Path*.