Modifications Included in APL64 Pre-Production Version (2020.0.5)

Table of Contents

Overview 6

Enhancements 6

Inline Control Sequences (ICS) are supported 6

Editor Autosave Options Content feature 6

Find in Functions in Workspace Utility 6

Improvements to Keyboard Definition dialog 7

The Current Keyboard Definition state is displayed in the APL Virtual Keyboard 8

Widths for several menus were reduced in the APL64 window 8

⎕pw ← ¯1 enhancement 8

User Command Processor Available 8

Support for non-scalar primitives in in-place assignment 9

New Alt+Shift+End and Alt+Shift+Home keyboard shortcuts 9

History option added to Copy Target in Executed Statements History dialog 9

⎕TCBEL updated to produce a “beep” sound 9

Click a workspace in the Recent Workspace Ids and not run the ⎕LX in the workspace 9

Updates to ⎕SYSINIT 9

⎕SYS[20] bumped up to 100 10

New Regular Expression verification tool built into the Find/Replace dialog 10

New menu: File | Emit Timestamps in Long Date Format 10

New Command Line Arguments: NAVIGATEDIRS & EMITLONGDATEFORMAT 10

NAVIGATEDIRS 10

EMITLONGDATEFORMAT 11

Find/Replace Dialog: Text to Find and Text to Replace ComboBoxes improved 11

Classic History Format Improved 12

Classic History Format / Function Editor: Line Numbers Right Justified 12

Updates to menu Options | APL+Win Configuration Conversion 13

Bug Fixes 14

Incorrect results when Windows Regional settings for decimal separators set to comma 14

History display issues or crash when switching between Classic and Sequential formats 14

Canceling "Edit Variable Name" dialog returned DOMAIN ERROR in the message box 14

The left margin in the function editor for viewing and setting stops/traces is missing 14
⎕idloc did not return the correct result for a function Label 14
All assignment (←) and goto (→) glyphs may not be found in the Find/Replace dialog 14
Menu Objects | Variable Editor | Strip Trailing Blanks is functional 14
Improvements to the Configure User Tools dialog 15
New menu: Session | Tooltips | Initial Show Delay 15
Executing ⎕PENCLOSE with each operator on the right gave an erroneous error message 15
In-place commands returned NONCE ERROR: FIXME: Code.XFNASSIGN not yet implemented 16
The :goto control structure statement returned the error message: *:GOTO argument must be the name of a label* 16
Commenting/Uncommenting a large selection of lines may take longer to complete 16
Tabbing/Untabbing a large selection of lines may take longer to complete 16
Spacing/Unspacing a large selection of lines may take longer to complete 16
APL component files can be lost when tying a file with a tie number already in use 16
⎕**dr ↑0⍴3.14** incorrectly returned 645 instead of 323 16
Match Hilite Color Modifications 16
Down arrow key on Last Line in Classic History 16
The **File | Copy** and **File | PCopy** dialogs returned WS NOT FOUND error messages 17
Evaluators' Comments & APLNow's Responses 18
Pre-production 2020.0.3 failed to start for an evaluator 18
Pre-production 2020.0.3 stopped running on a Virtual Machine after changing the keyboard from US to French 18
The del-tilde () in APL64 has a different appearance from APL+Win (⍫) & does nothing 18
Initial feedback with running a demo workspace with ⎕WI and Assembler function 18
Displaying the tooltip on Mouseover at the command line and some other controls is disruptive. 18
I would recommend removing the tooltip for the session and for the command line. This information should be provided at another place. 18
With French keyboard in the Keyboard Definition, any APL calculation instruction involving a number with a period (.) yields a SYNTAX ERROR error message 18
In-place assignments return a NONCE ERROR 19
My Documentation Options dialog doesn't match the Jan 2022 update document 19
I am evaluator in Germany and 0.1 + 3 returned 4. Why? 19
How can I configure APL64 to have the same German keyboard settings as APL+Win? 19
The session command line and history have different keyboard layouts 21

How do you solve the problem of the multiple execution datagrid constantly popping up yielding a repetitive flicker in the Session? 21
There is an awful flicker when pressing F10 during a debugging session 22
Why did the Select Keyboard Definition combo box always display Custom? 22
Opening an existing function in the editor with double-click does not work 23
□idloc not returning the correct result for a function Label 23
All assignment glyphs were not found and highlighted in the Find/Search dialog 23
□sys[18] returns 0 for relative responsiveness 23
The Ctrl+= shortcut doesn't magnify the screen font 23
How do I enter split stile in the action text of a user defined tool? 24
APL64 help files open in the default browser instead of Adobe Acrobat set as default browser? 24
How can APL64 recognize F13-F24 keys? 24
Suggestion to display more information in error messages containing FIXME 24
Switch Focus Between Session Command Line/History & Most Recent Object Editor…(Ctrl+T)" doesn't work 25
I am really puzzled by □PF 25
Odd APL64 crashes in Windows 8.1 25
The Delete Selected User Tools and Delete all User Tools buttons in the Existing User Tool Definition dialog do NOT achieve their purpose UNLESS there depression is followed by the depression of the OK button 26
Consider removing 'User' from the Delete Selected User Tools and Delete all User Tools buttons in the Existing User Tool Definition dialog for a neater appearance 26
Menu Text in the Existing User Tool Definition dialog is NOT showing a shortcut 26
Why did I receive the message Configuration Conversion Failed A duplicate key 'Config:InstallationPath' was found when converting an APL+Win configuration file? 26
Would []XL work with Google Sheets? 26
Does []XL manipiulate the underlying Excel XML documents or is it using ODBC? 26
)EDSS could simply be)SE (Sheet Editor) 27
How do you abandon the)EDSS editor session? 27
Is it possible to save the)EDSS editor sessions in progress? 27
There is no shortcut or quick way to empty the Command Line window 27
Pressing Ctrl+V pastes the clipboard content in the Command window by default 27
Could you at least arrange the tooltips not to appear until I actually MOVE my mouse, instead of it just being reactivated after hitting Enter? 27
Why does the report result of □MF have 5 columns instead of 3? 28

What is the value in the cells that are not populated in worksheet editor? 28
Rank of the array changes in Worksheet Editor when entering values non-contiguously 28
Error messages displayed when selecting the SpreadsheetGear option from the context menu in Worksheet Editor 28
Can visible rows/columns can be restricted in worksheet editor? 28
'#' []WI 'caption' 'XYZ' doesn't seem to work - should it? 28
'#' []WI 'hwndmain' reports only a 0. Has this not been implemented? 28
Are APL64 files the same as .sf files? 29
□CF-fns with floating point tie numbers (e.g., 1.5-.5) returned DOMAIN ERROR 29
Option for a traditional APL workspace environment in APL64 29
It's no longer possible to set/remove stops in the margin in the function editor 29
Ability to debug (with stop and trace) functions on the go 29
In APL+Win, I cannot place a Stop on a comment line of a function, except for line [1], which is different from APL64 30
[]PENCLOSE returned SYSTEM ERROR:Aval.Categorize encountered an uncategorized child element at index=0 30
What is the latest info on support for Inline Control Structures (ICS) 30
Is there an alternative to □wi based forms in APL64? 30
Does APL64 come with supplied workspaces? 30
FTIMEFMT looks like a user-defined function. How do I get the FTIMEFMT function? 31
The :goto control structure statement does not recognize labels 31
There is an awful flicker (debugger panes) when pressing F10 during a debugging session 31
Are you sure APL64 is supporting custom classes, e.g., zObjects? 31
Is the pre-compiled version of the APLNow32.adf file provided in APL64? 31
Would it be possible to make the Alt+Shift+End key work like it does in APL+Win, i.e. erase from cursor to end of APL Session? 31
Commenting 200 lines in the function editor with Ctrl+/ takes longer in APL64 31
Moving a bunch of lines to the right or left with Tab & Shift+Tab takes longer in APL64 31
Moving a bunch of lines one space with Ctrl+Spacebar & Ctrl+Shift+Spacebar takes longer in APL64 31
APL component files can be lost when tying a file with a tie number already in use 31
User Defined Control works in APL64 31
□dr ↑0ρ3.14 returns 645 instead of 323 like in APL+Win 31
Why am I getting all-0's out of this formatting example: 1 0 ⊤⍤0 1 0 1 0 1 31
Issue with the take primitive function for packed Booleans 32

Tooltips remained visible after Alt+Tabbing to another app 32
Incorrect highlighting in the target for regular expressions in the Find dialog 32
Are there limitations to the Find dialog Regular Expressions? 33
Regular expressions in the Find/Replace dialog can easily crash APL64 33
Ctrl+ mouse click a Recent Workspace Id does not)XLOAD the workspace like in APL+Win 33
Results for ⎕AT, ⎕CFHIST, ⎕CFRDCI are wrong by two hours (relative to ⎕TS) 33
Is there a way to run a function in the background i.e. asynchronously? 33
⎕vr has the wrong indentation (i.e., missing leading spaces) starting at line 1000 33
How Unicode aware is APL64? Which APL fonts, if any, has the boxing characters? 34
The ~ (not) symbol and the – (minus) symbol look a bit too much the same in the APL Session and in APL64 programs 34
I can no longer display the ⍨ (Commute) symbol on my keyboard 34
Will APL64 will be compatible with EVLEVEL 1? 35
Error Message: "SYSTEM ERROR:Aval.Categorize encountered an uncategorized child element at index=1" 35
Error Message: "SYSTEM ERROR:Exception: CS0023: Operator '!' cannot be applied to operand of type 'int'" 35
Could you improve the look and feel of horizontal splitters [in APL64]? 35
Can the closing message box and dialogs be centered over the APL64 application? 35
Outlining shortcut keys are missing in APL64 35
Ctrl+Z and Ctrl+Y shortcuts work in the Find/Replace form to be synonyms as clicking the Undo and the Redo buttons 36
Is ⎕os in Windows 11 returning the correct result? 36
Interfacing APL64 with C# 37

Overview

This document describes the enhancements and bug fixes incorporated into the fourth pre-production version of APL64 released in 2020.0.5. This document also describes the evaluator comments and APLNow responses based on the prior pre-production versions of APL64.

Enhancements

Inline Control Sequences (ICS) are supported

We had not planned to release APL64 with support for ICS in the first production release of APL64. But after considerable demand for it by our users, the decision was made to accelerate the implementation of ICS for this pre-production release to ensure that it would be available for the first production release of APL64.

Editor Autosave Options Content feature

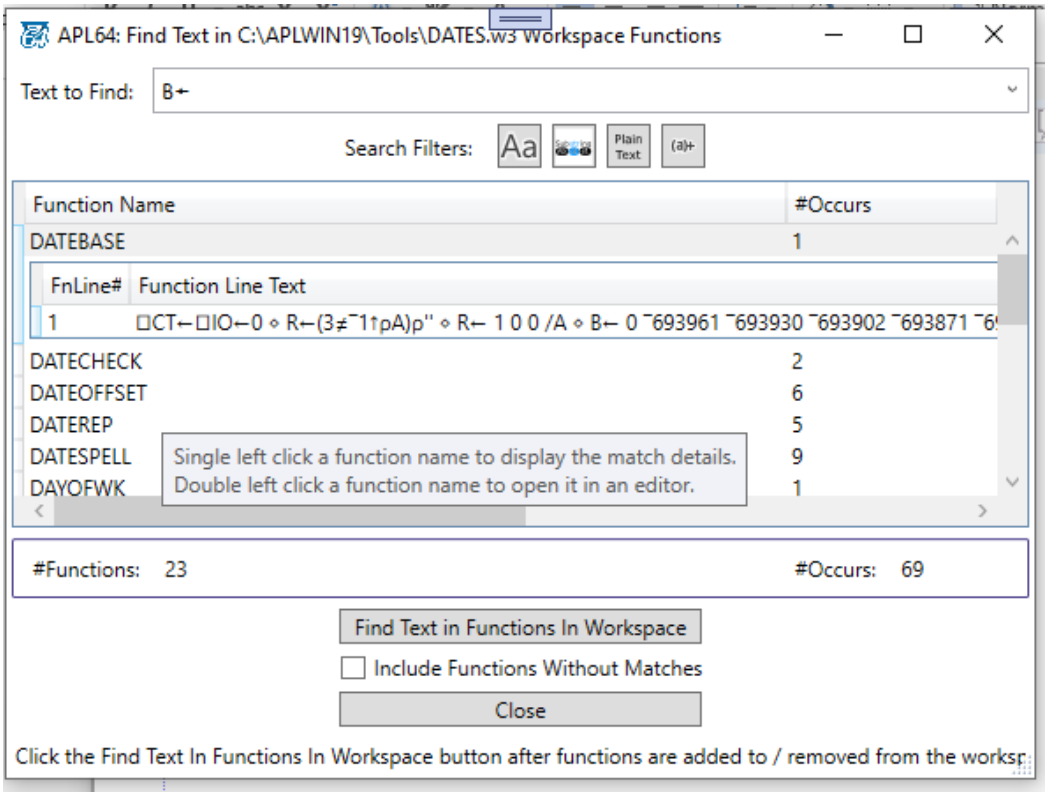
The Editor Autosave Options dialog provides user-selected options to automatically save object editor content during an APL64 Developer version instance. This feature may be useful if there is a system crash due to an unhandled software exception or a hardware or environment problem. Rather than 'waiting' for such a problem to occur and attempting to save object editor content using what may be an unstable system environment, periodic autosaving of editor content is more reliable.

The Editor Autosave Options dialog is accessed from the menu **Objects | Editor Autosave Option**.

Documentation is available from the menu **Help | Developer Version GUI | Using Autosave Object Editor Content**.

Find in Functions in Workspace Utility

The Find in Functions in Workspace utility may be used to search for user-selected text within all currently defined functions in the workspace.



The Find in Functions utility is accessed from the menu **Edit | Find in Functions in Workspace**.

Documentation is available from the menu **Help | Developer Version GUI | Using Find in Functions in Workspace**.

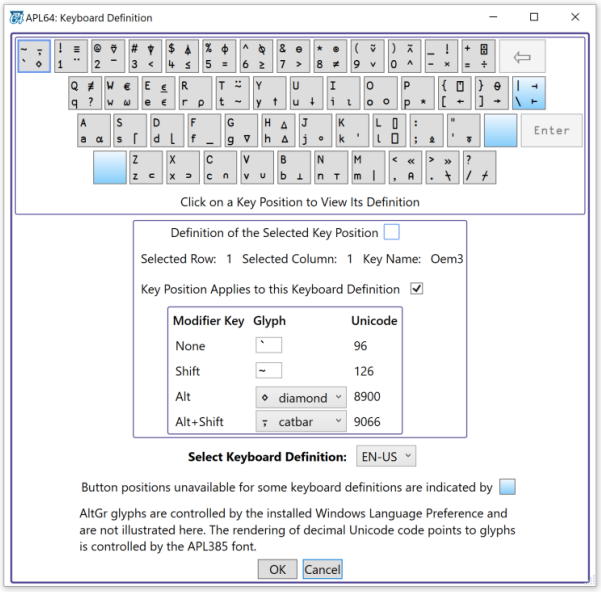
Improvements to Keyboard Definition dialog

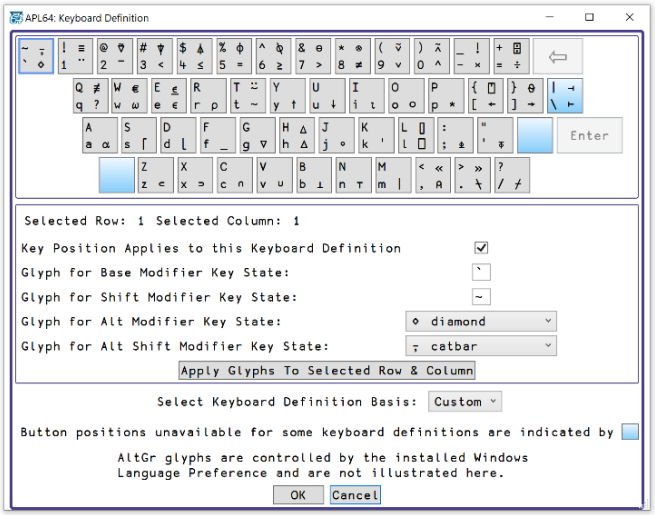
Key updates:

- General redesign of the layout
- The Custom keyboard definition has been removed
- The Select Keyboard Definition combo box now displays the user selected keyboard definition

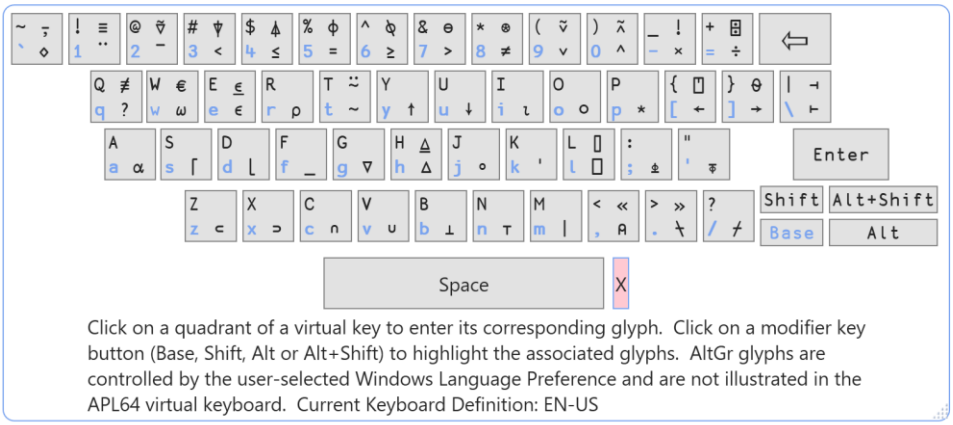
Release 2020.0.4 and prior

Release 2020.0.5





The Current Keyboard Definition state is displayed in the APL Virtual Keyboard



Widths for several menus were reduced in the APL64 window

Menu items in the File, Edit, Session and Objects menus were reorganized to reduce the overall width of the menus.

□pw ← ¯1 enhancement

□pw is assignable to ¯1 under program control to match the line width to the current history window.

User Command Processor Available

The user command processor facility is automatically installed and configured in this release. After starting APL64, execute **]?** to display the names of the user command functions available.

Support for non-scalar primitives in in-place assignment

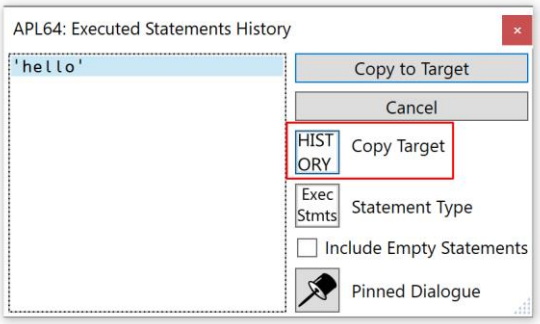
The previous enhancement to in-place assignment has been expanded to support non-scalar primitive functions like rotate, transpose, match, mismatch, member of and find.

New Alt+Shift+End and Alt+Shift+Home keyboard shortcuts

Shortcut	Action
Alt+Shift+End	Erases all characters from the current caret position to the end in the history and editor.
Alt+Shift+Home	Erases all characters from the current caret position to the beginning in the history and editor.

History option added to Copy Target in Executed Statements History dialog

When the menu **Session | Classic History Format | Is Editable** is checked, History is an option in the Copy Target button in the Executed Statements History and **Edit | Gather** dialogs to allow the copy to appear in the history. E.g.,



⎕TCBEL updated to produce a "beep" sound

⎕TCBEL now produces a "beep" or bell sound like in APL+Win.

Click a workspace in the Recent Workspace Ids and not run the ⎕LX in the workspace

A selection of a workspace in the Recent Workspace Ids list in the File menu while holding the Ctrl key will XLoad the workspace and not run the ⎕LX (LATENT EXPRESSION) when set in the workspace.

Updates to ⎕SYSINIT

The two yellow highlighted indices below are new in this release:

Index	Index ⊃⎕SYSINIT
1	Int32: Process.GetCurrentProcess().Id
2	Reserved
3	CharVec: Environment.CommandLine
4	Reserved
5	CharVec: Process.GetCurrentProcess().ProcessName
6	Reserved
7	Reserved
8	CharVec: Window title for the main APL64 session window
9	Int32: Windows handle for the main APL64 session window; zero in runtime. Note 1

10	Vector of character vectors: Environment.GetCommandLineArgs(): One row for each command line argument used to start the APL64 instance

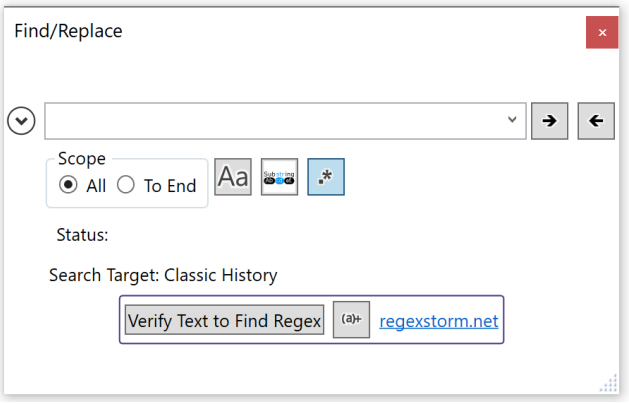
Note 1: This is the recommended replacement for the ⎕wi command **'#' ⎕wi 'hwndmain'**.

⎕SYS[20] bumped up to 100

⎕SYS[20] stores the saved workspace version number. The APL64 workspace version number was increased to 100 from 10 to avoid any conflict with APL+Win workspace version numbers. **Note**: This will result in the display of the standard workspace version upgrade warning message when loading a workspace saved in the prior version.

New Regular Expression verification tool built into the Find/Replace dialog

When in **Regular Expression [Regex] Text to Find** mode, click the button labeled *Verify Text to Find Regex* to quickly validate the regular expression courtesy of the Regex Storm .NET regex tester. Click the adjacent button to open the browser to the Regular Expression Language – Quick Reference guide from Microsoft.



New menu: File | Emit Timestamps in Long Date Format

You can choose between the short date format and the long date format with the **File | Emit Timestamps in Long Date Format** menu. When checked, dates displayed in the history will be displayed in the long date format, e.g., **Friday, March 4, 2022 4:00:29 PM**, instead of the short date format, e.g., **3/4/2022 4:00:20 PM**. This applies to the timestamps associated with loading, saving, copying, and dropping workspaces.

New Command Line Arguments: NAVIGATEDIRS & EMITLONGDATEFORMAT

NAVIGATEDIRS

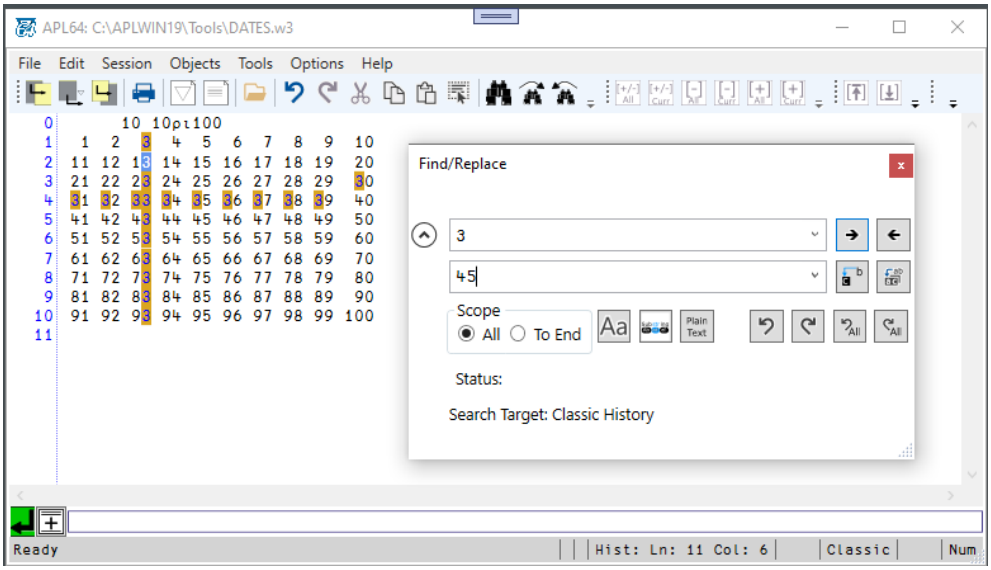
- This command line argument determines if the current directory of the APL64 Developer version will change when specified file actions are performed via menu items, shortcuts or system functions/commands:
  - File | Load Workspace
  - File | XLoad Workspace
  - File | Copy Workspace
  - File | PCopy Workspace
  - File | SaveAs
  - File | Drop

    - Session | Configuration Settings | Export Settings
    - Options | History Log | Import History Log
- Supported argvalue: 0, 1
- Default value: 0
- Value 0: The current directory is not modified
- Value 1: The current directory may be modified
- This information is saved in the APL64 xml-format configuration file.
- This command line argument does not affect file options where the current directory is not modified:
    - Session | Configuration Settings | Import Settings
    - Options | Create Runtime .Net Assembly
    - Options | APL+Win Configuration Conversion
    - Help | License & Activation | APL64 License Activations/Deactivations
- Examples:
    - NavigateDirs=0
    - NAVD=1

EMITLONGDATEFORMAT

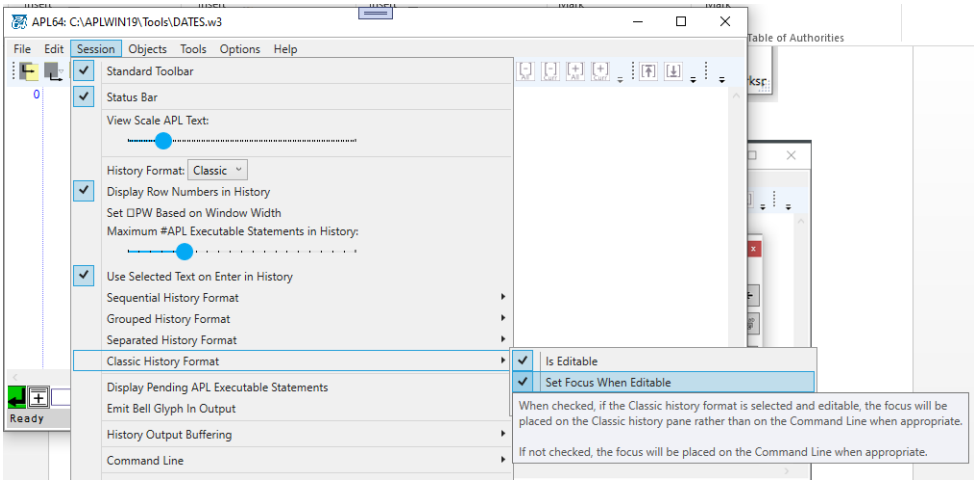
- This command line argument determines if the APL64 interpreter will emit timestamps in long or short format associated with loading, saving, copying, dropping workspaces and session log.
- Supported argvalue: 0, 1
- Default value: 0
- This information is saved in the APL64 xml-format configuration file.
- Value 0: The short date format will be emitted, e.g. …Friday, March 4, 2022 4:00:29 PM
- Value 1: The long date format will be emitted, e.g. … 3/4/2022 4:00:29 PM
- Examples:
    - EMITLONGDATEFORMAT=0
    - ELDF=1

Find/Replace Dialog: Text to Find and Text to Replace ComboBoxes improved

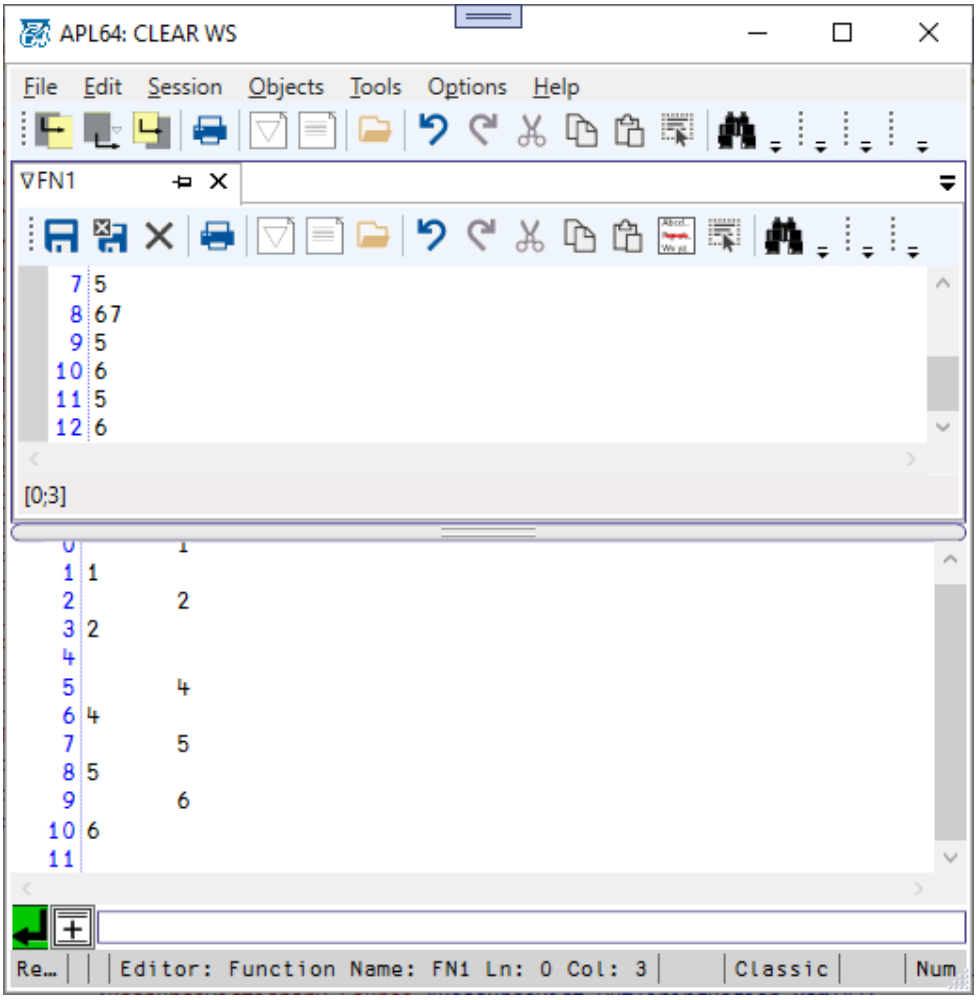


Classic History Format Improved

When the APL64 programmer selects **Session | History Format: Classic**, **Session | Classic History Format | Is Editable** is checked and **Session | Classic History Format | Focus When Editable** is checked, the Classic history format behavior will be very similar to the APL+Win history.

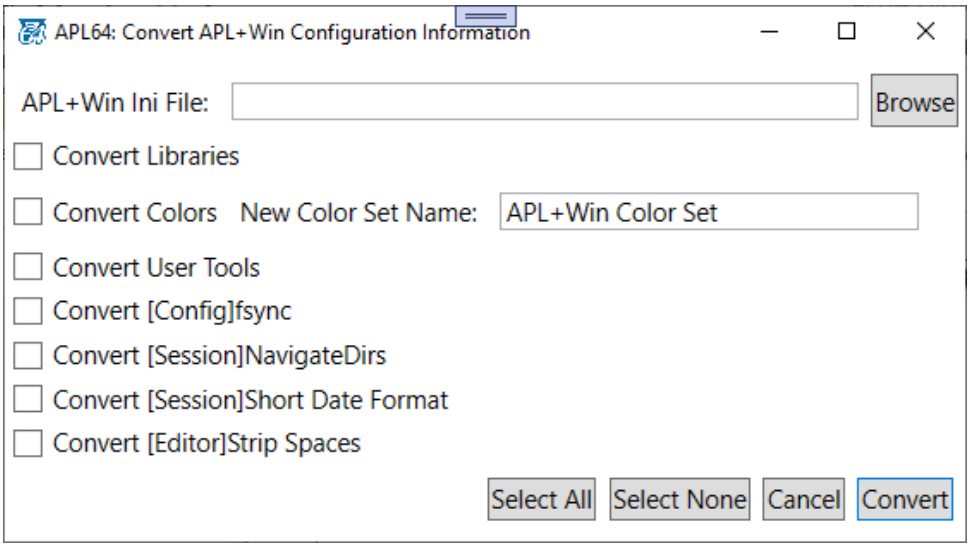


Classic History Format / Function Editor: Line Numbers Right Justified



Updates to menu Options | APL+Win Configuration Conversion

Additional elements of the APL+Win Configuration may be converted to the analogous APL64 configuration.



Bug Fixes

Incorrect results when Windows Regional settings for decimal separators set to comma

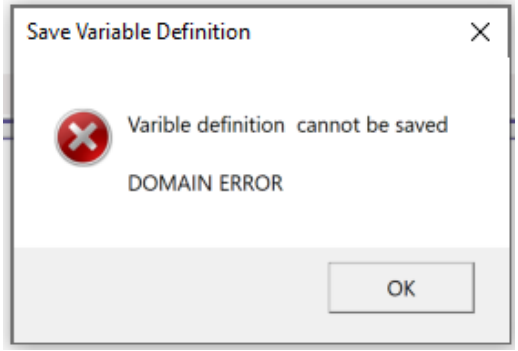
When the Windows regional settings for the decimal symbol was set to a comma instead of a period, APL64 was interpreting the comma as the APL catenate, not the decimal separator. This resulted in the APL statement of $0.1 + 3$ to return a 4.

History display issues or crash when switching between Classic and Sequential formats

Somewhat related to the preceding bug, when the Windows regional settings for the decimal symbol was set to a comma instead of a period and the digit grouping symbol was set to a period instead of a comma, the row numbers in the history could become distorted (enlarged) or possibly crash APL64 when switching to the Sequential from the Classic history format.

Canceling "Edit Variable Name" dialog returned DOMAIN ERROR in the message box

When pressing Ctrl+E (Save and Exit) in the edit session and prompted with the Edit Variable Name dialog, when selecting the Cancel button, instead of closing the dialog and returning to current edit session, a second dialog displayed with the message: Variable definition cannot be saved
DOMAIN ERROR. E.g.,



The left margin in the function editor for viewing and setting stops/traces is missing

This margin to the left of the row numbers in the function editor for viewing and setting stops and traces was not present in the pre-production 2020.0.4 release. The left margin has been restored in this release.

□idloc did not return the correct result for a function Label

The value 2 (variable) was returned instead of 8 (label).

All assignment (←) and goto (→) glyphs may not be found in the Find/Replace dialog

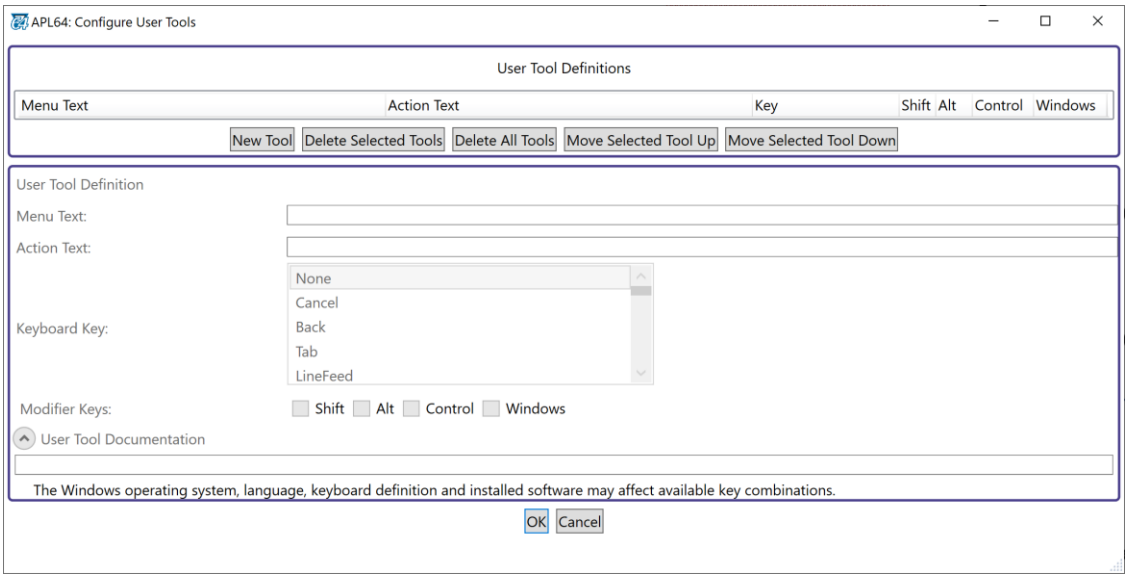
This occurred when the search type was set to APL Tokens and/or Whole words matching.

Menu Objects | Variable Editor | Strip Trailing Blanks is functional

The menu was present but inoperable in the previous release.

Improvements to the Configure User Tools dialog

1. The MoveUp, MoveDown and User Tool Definition section of the dialog are enabled only if exactly one user tool definition is selected
2. 'Existing' removed from 'Existing User Tool Definitions' label
3. Button captions for tool definitions were made consistent
4. Tooltips are provided for New Tool and Delete Selected Tools buttons
5. The Modifier Keys settings were added to the User Tools Definitions area
6. A new prompt for confirmation will display when the X or Cancel buttons are clicked when there are unsaved changes in the dialog



New menu: Session | Tooltips | Initial Show Delay

This menu option allows you to specify, in milliseconds, the frequency in showing tooltips while the mouse navigates in the window.

Executing ⍁PENCLOSE with each operator on the right gave an erroneous error message

E.g.,

```
      1 ⍁PENCLOSE ¨'APL64'
SYSTEM ERROR:Aval.Categorize encountered an uncategorized child element at index=0
  at System.Environment.get_StackTrace()
  at APLNow.Data.SystemError.Message(Exception exception, String more, Object[] args)
  at APLNow.Data.Aval.Categorize()
  at APLNow.Data.Aval.IpDemote()
  at APLNow.Core.Code.EachBase.Exec(Apl apl)
  at APLNow.Core.Apl.Run()
  at APLNow.Core.Apl.Interpret(Who caller)
  at APLNow.Core.Apl.DoExecCall()
  at APLNow.Core.Apl.OnExec(Chars expr, Who who, ExStepFlags step)
  at APLNow.Core.AplExecTask.<>c__DisplayClass3_0.<.ctor>b__0()
  at APLNow.Core.Executor.ExecPending(Boolean wasWaiting)
```

```
   at APLNow.Core.Executor.Run()
[imm] 1 ⎕PENCLOSE ¨'APL64'
```

In-place commands returned NONCE ERROR: FIXME: Code.XFNASSIGN not yet implemented

In-place commands are supported in this pre-production release of APl64.

The :goto control structure statement returned the error message: *:GOTO argument must be the name of a label*

This situation occurred when the :goto statement appeared in a diamond delimited statement.

Commenting/Uncommenting a large selection of lines may take longer to complete

There may be a noticeable delay when using the Ctrl+/ and Ctrl+Shift+/ keyboard shortcuts to add a comment and remove a comment, respectively, a large block of lines (>100) in the function editor when the function contains several hundred lines of code.

Tabbing/Untabbing a large selection of lines may take longer to complete

There may be a noticeable delay when using the Tab and Shift+Tab keyboard shortcuts to add an indent and remove an indent, respectively, a large block of lines (>100) in the function editor when the function contains several hundred lines of code.

Spacing/Unspacing a large selection of lines may take longer to complete

There may be a noticeable delay when using the Ctrl+Spacebar and Ctrl+Shift+Spacebar keyboard shortcuts to add a space and remove a space, respectively, a large block of lines (>100) in the function editor when the function contains several hundred lines of code.

APL component files can be lost when tying a file with a tie number already in use

Tying a component file with a tie number that was already in use reported the FILE TIED error message and deleted the file that was being tied.

⎕dr ↑0ρ3.14 incorrectly returned 645 instead of 323

Match Hilite Color Modifications

When the Match Hilite color is modified, the Find/Replace results will be updated if the Find/Replace dialog is open.

Down arrow key on Last Line in Classic History

When the down arrow key was clicked, without any modifier keys, on the last line in the Classic History, the focus did not move to the Command Line, as expected.

The File | Copy and File | PCopy dialogs returned WS NOT FOUND error messages

This occurred when the folder pathname contained one or more blank spaces.

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 (2020.0.4.3062). We would like to thank the evaluators for their contribution.

Pre-production 2020.0.3 failed to start for an evaluator

The root of the problem is still unknow. However, the issue didn't surface with pre-production 2020.0.4.

Pre-production 2020.0.3 stopped running on a Virtual Machine after changing the keyboard from US to French

The root of the problem is still unknow. However, the issue didn't surface with pre-production 2020.0.4.

The del-tilde () in APL64 has a different appearance from APL+Win (⍛) & does nothing

Incorrect. The lock primitive function still has purpose in APL64 (and APL+Win). The lock primitive function can lock functions when created with a system function like ⎕DEF, in both APL systems.
E.g. ⎕def '⍛foo',⎕tcnl,'[1]1+1',⎕tcnl,'[2]2+2',⎕tcnl,'⍛'

The difference in the appearance of the lock function in APL+Win and APL64 is purely cosmetic and the result of the different fonts in both systems; TrueType in APL+Win and Unicode in APL64.

Initial feedback with running a demo workspace with ⎕WI and Assembler function

Regarding the issue of migrating applications utilizing assembler code, substitute the current APL+Win functions with the APL64 replacement functions available in ASMFNS.ws64 workspace that was installed with APL64. The workspace will be in the Tools folder in the APL64 User's path. You can query the User's path with the system variable ⎕userpath in APL64.

It will also be beneficial to review the document *APL64 Compatibility Modifications for APL+Win Applications* accessible in the menu **Help | APL+Win Compatibility**.

Displaying the tooltip on Mouseover at the command line and some other controls is disruptive.

The display of tooltips in the system is easily configurable in the menu setting **Session | ToolTips | Display[]**.

I would recommend removing the tooltip for the session and for the command line. This information should be provided at another place.

You can already do this with setting the menu to either **Session | ToolTips | Display: [ToolBar ToolTips]** or **Session | ToolTips | Display: [ToolBar and Menu ToolTips]**.

With French keyboard in the Keyboard Definition, any APL calculation instruction involving a number with a period (.) yields a SYNTAX ERROR error message

This is not a keyboard definition issue. In the next APL64 version, the APL64 interpreter will parse numeric values, including those with a decimal point, using the .Net Culture Invariant option to eliminate the disparity between the user's regional language settings for decimal and digit separators

and APL64. Because the comma is reserved for the APL catenate and ravel functions, in APL the regional language settings for decimal and digit separators must be ignored.

In-place assignments return a NONCE ERROR

The next APL64 preproduction version will resolve this issue.

My Documentation Options dialog doesn't match the Jan 2022 update document

The description and image on page 21 identified the check box as having been eliminated in 2020.0.4.

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:

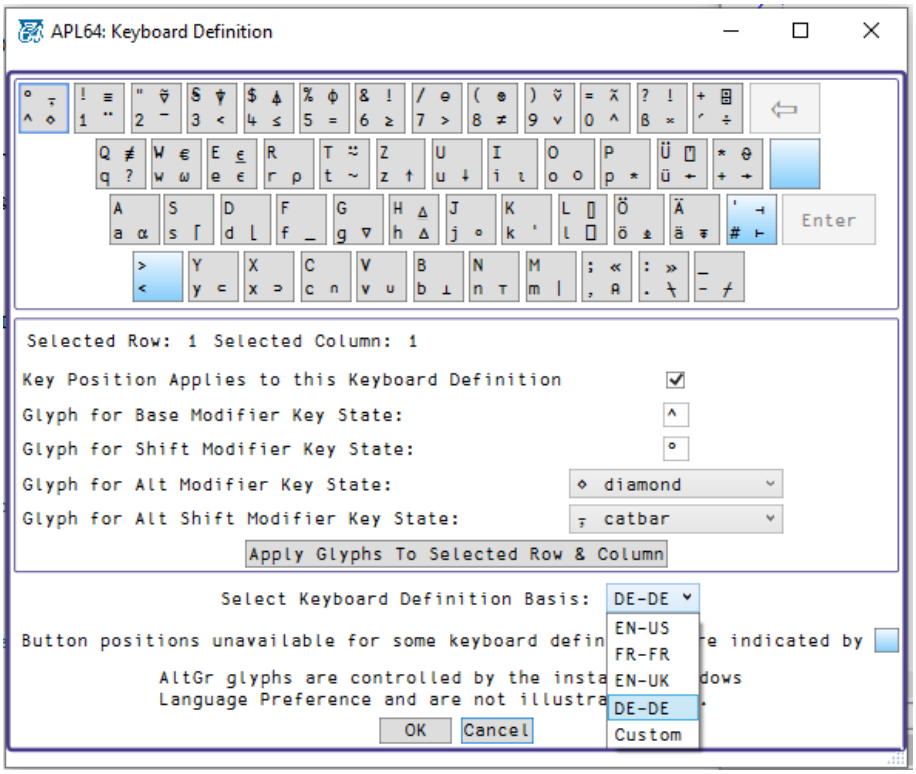
I am evaluator in Germany and 0.1 + 3 returned 4. Why?

When regional settings for the decimal separator was set to a comma instead of a period, APL64 was interpreting the comma as the APL catenate, not the decimal separator. This resulted in the APL statement of $0.1 + 3$ to return 4. A fix will be available in a future release.

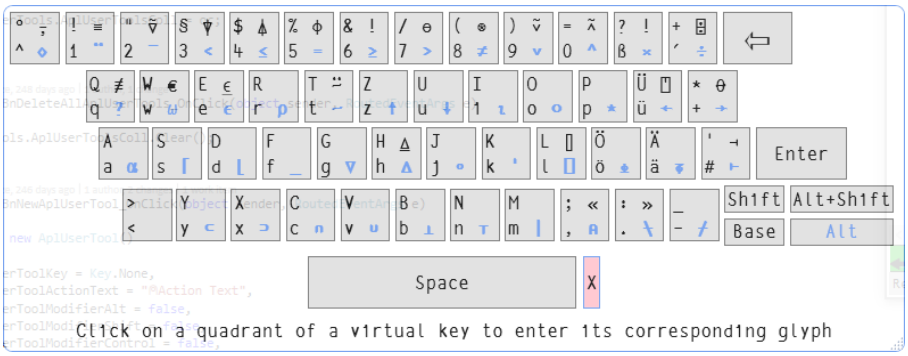
How can I configure APL64 to have the same German keyboard settings as APL+Win?

Here is how to check that the DE-DE keyboard is properly setup on the workstation:

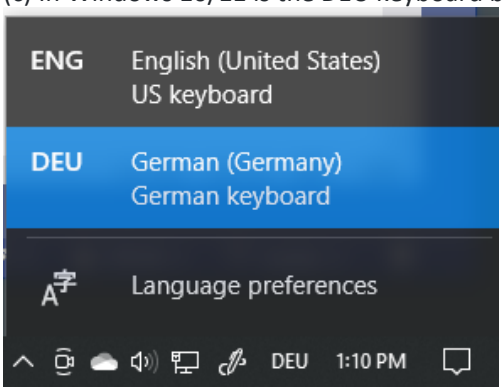
(a) Has the menu **Options | Keyboard Definition** in APL64 been used to select the DE-DE keyboard definition? This operation requires a restart of APL64.



(b) When the virtual keyboard is presented (menu **Session | APL Virtual Keyboard**) is presented, is the DE-DE keyboard definition shown?



(c) In Windows 10/11 is the DEU keyboard been selected using the list in the Windows taskbar?



If not the **All Settings | Time & Language | Language** options must be used in Windows 10/11.

The session command line and history have different keyboard layouts

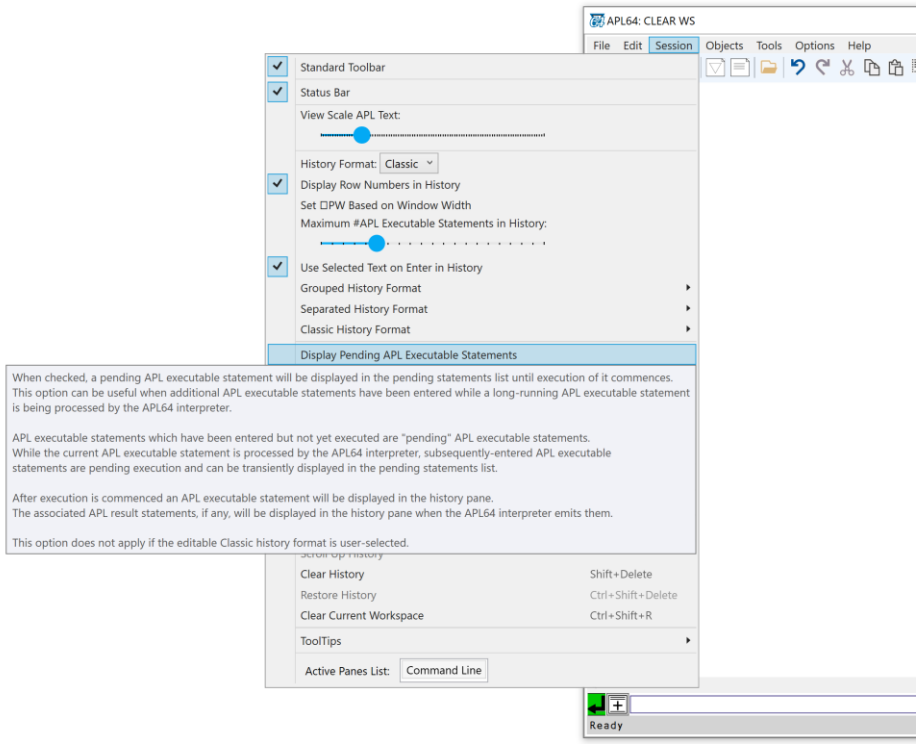
This can occur when the keyboard definition is not currently set for your region (country). The keyboard definition is configured in the menu **Options | Keyboard Definition**.

How do you solve the problem of the multiple execution datagrid constantly popping up yielding a repetitive flicker in the Session?

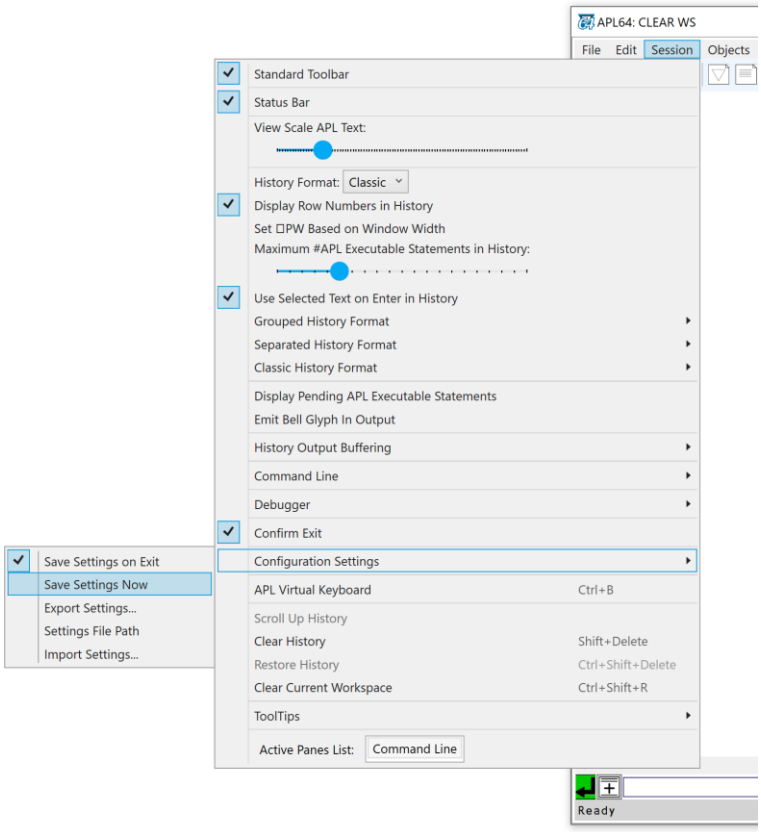
Multiple execution in APL64 is the same as multiple execution in APL+Win, i.e. the ability to include more than one independent APL executable statement into one APL executable statement. The only difference is that is APL64 the separator between the included APL executable statements is a user choice of the APL diamond glyph or the new line character. APL+Win is limited to the diamond glyph separator.

Follow the steps below to prevent the display of the pending APL executable statements datagrid for the current and future APL64 developer version instances on this workstation.:

- Start an APL64 instance on your workstation
- Uncheck the menu **Session | Display Pending APL64 Executable Statements** checkbox



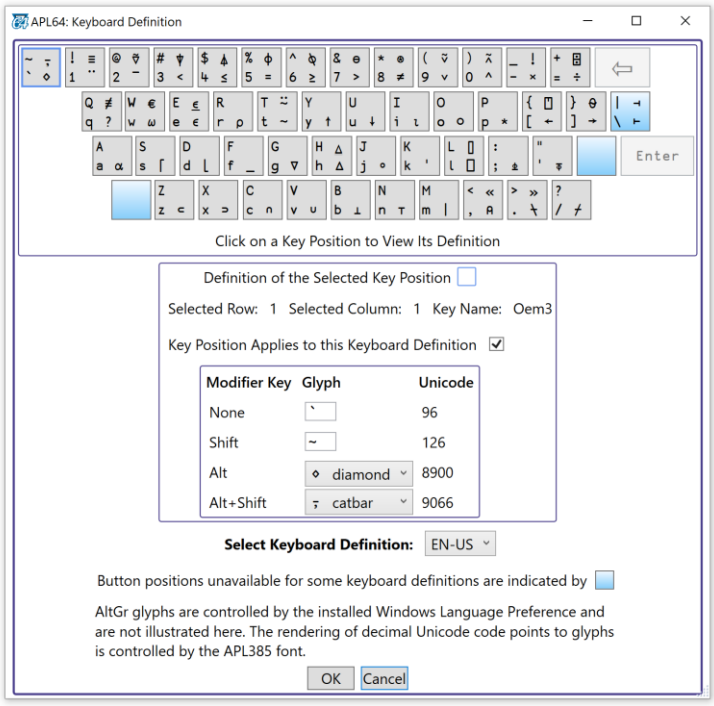
- Use menu **Session | Configuration Settings | Save Settings Now**



There is an awful flicker when pressing F10 during a debugging session

Why did the Select Keyboard Definition combo box always display Custom?

This was the behavior in pre-production versions prior to 2020.0.5. In 2020.0.5, the user selected keyboard definition now displays. E.g.,



Opening an existing function in the editor with double-click does not work

This option is not enabled by default in APL64. This is configured in the menu settings below:

- **Objects | Left Mouse Double Click: Open Object Named at Caret**
- **Objects | Right Mouse Double Click: Open Object Named at Caret**

☐idloc not returning the correct result for a function Label

This was identified as a bug and corrected in the next pre-production release.

All assignment glyphs were not found and highlighted in the Find/Search dialog

There was an issue with searching some APL glyphs in the Find/Replace dialog with the search type set to whole word and APL Token matching. This will be investigated and addressed in a future pre-production release.

☐sys[18] returns 0 for relative responsiveness

In APL+Win, the value corresponds roughly to the number of primitive functions the system executes between tests to detect when the Ctrl+Break shortcut key is pressed to signal an interrupt in the interpreter. This value is irrelevant in APL64 because of the asynchronous separation of the GUI window from the interpreter. Therefore, the value returned is always a zero in APL64. Also, the responsiveness value in the command line or the configuration file ([Config]Responsiveness = n) will have no impact in APL64.

The Ctrl+= shortcut doesn't magnify the screen font

We are investigating the ease with which to provide this functionality in APL64.

How do I enter split stile in the action text of a user defined tool?

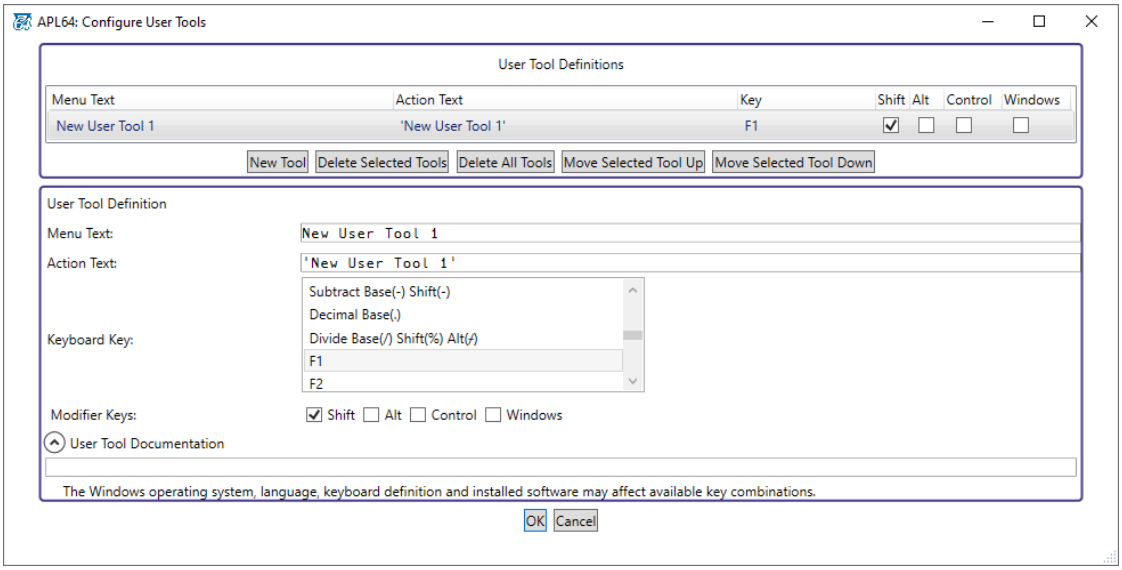
It will not be possible to enter the split stile glyph in APL64 because the APL385 Unicode font in APL64 does not support the glyph.

APL64 help files open in the default browser instead of Adobe Acrobat set as default browser?

The user's default browser and not the user's default pdf-format document viewer will be used. The documentation of this behavior in the January, 2022 APL64 evaluation version was not correct.

How can APL64 recognize F13-F24 keys?

Try defining a user tool in APL64 with the F1 key and Shift modifier key selected. This user tool will be executed when Shift + F1 is click by the user:



Evidently your keyboard does not have physical F13-F24 keys. If it did, or if your keyboard keys could be programmed to generate genuine F13-F24 keystrokes, APL64 would recognize them as such. APL64 recognizes Shift + F1-F12 emitted by your keyboard as Shift + F1-F12 and not F13-F24. Unfortunately, some popular keyboards come pre-programmed with the Shift + F1-F12 targeted to hardware settings such as monitor brightness and contrast, sound intensity, sleep/awake etc. Therefore, it would not be appropriate for APL64 to assume that Shift + F1-F12 should be handled by APL64 as F13-F24 and thereby block the pre-programmed hardware actions.

Suggestion to display more information in error messages containing FIXME

Any mention of the word FIXME in the error message isn't appropriate. You're seeing them in the pre-production release because they are acting as placeholders for the developers to revisit for implementation.

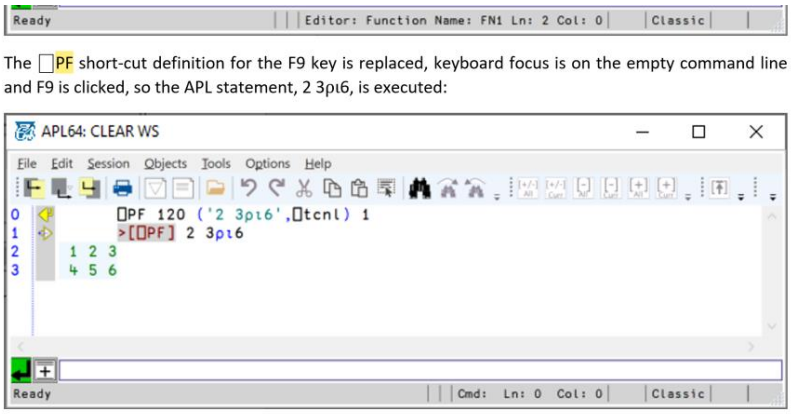
Switch Focus Between Session Command Line/History & Most Recent Object Editor…(Ctrl+T)" doesn't work

It has been investigated and the fix will be in the next pre-production release. Note that the placement of the focus will depend on the setting of the menu **Session | Classic History Format | Is Editable**; the target of the focus will be the history when **Is Editable** is checked and the command line when **Is Editable** is not checked.

I am really puzzled by ⎕PF

Thank you for your comments on the ⎕PF system function. In case you were not aware, the design and action of ⎕PF in APL64 is the same as that in APL+Win, where it Is an undocumented feature. Some of the observations you listed are a rehash of what's already documented in the Effect: section. So, users should be aware of what the implications are with using ⎕PF. For example, one shouldn't be surprised that ⎕PF can "override APL64 predefined function e.g. F9" because the same thing can happen with a user tool. Right? Basically, the last one set will be the one in effect.

As to "ANY example of a second argument that might be an APL expression", please refer to the example on bottom half of the page on page 166.



Odd APL64 crashes in Windows 8.1

What we can share with you is that our limited testing was on a desktop machine with 16 GB of RAM running Windows 8.1 Pro with access to a LAN. Fortunately for us, we were unable to observe any of your issues.

For what it's worth, the minimum operating system requirement will likely be Windows 10. So, my recommendation is to upgrade your virtual machine to Windows 10 or newer.

The Delete Selected User Tools and Delete all User Tools buttons in the Existing User Tool Definition dialog do NOT achieve their purpose UNLESS there depression is followed by the depression of the OK button

Basically, any changes will not be preserved/saved until the OK button is clicked as is the case in most applications.

Consider removing 'User' from the Delete Selected User Tools and Delete all User Tools buttons in the Existing User Tool Definition dialog for a neater appearance

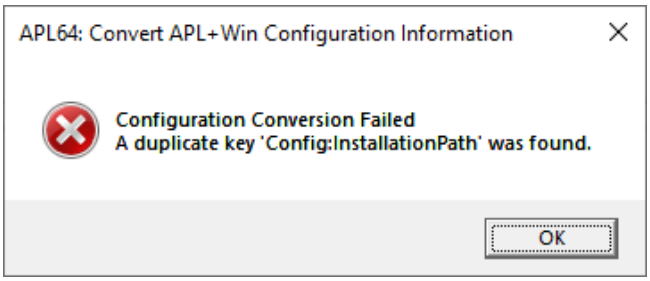
These changes will appear in the next pre-production release. A new prompt for confirmation will also be displayed when the X or Cancel buttons are clicked when there are unsaved changes in the dialog.

Menu Text in the Existing User Tool Definition dialog is NOT showing a shortcut

This is by design. The definition of a user tool in APL64 provides for an explicit specification of the short cut, if any, using the 'Keyboard Key' list and the 'Modifier Keys' checkboxes. In APL64 the user tool short cut is specified separately from the user tool menu text. When a tool definition is made using the menu **APL64 Options | Configure User Tools** dialog and the definition is saved, the APL64 Tools menu will reflect the specified shortcut, if any.

Why did I receive the message Configuration Conversion Failed A duplicate key 'Config:InstallationPath' was found when converting an APL+Win configuration file?

The 'Options: APL+Win Configuration Conversion' operation correctly identified that the user-provided APLW.ini file was defective with the message:



caused by duplicate entries in the user-provided APLW.ini file. After removing the duplicate entries in the user-provided APLW.ini file, the conversion of the APL+Win user tools to APL64 user tools was successful.

Would []XL work with Google Sheets?

Why not try it yourself? Create a Google Sheet file on the local workstation and point the []XL to that file. Please report back on your experiment.

Does []XL manipiulate the underlying Excel XML documents or is it using ODBC?

No. []XL does not modify the underlying Excel proprietary format (.xls) or the Open XML format (.xlxs, .xlxm) documents other than putting the APL64 data into the targeted worksheet. The APL64 programmer may create a customized Excel workbook and use []XL to fill or retrieve APL64 data without modifying the structure or formatting of the Excel workbook.

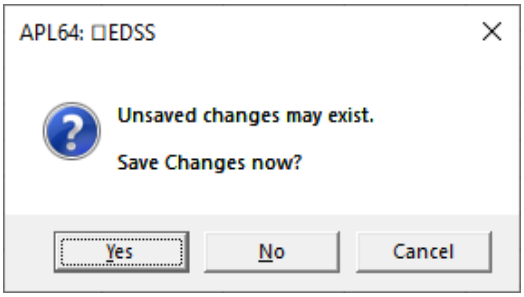
The interaction between APL64 and the Excel workbook and worksheet targeted by □XL is proprietary and does not use ODBC, ActiveX or Excel.exe.

)EDSS could simply be)SE (Sheet Editor)

)EDSS was selected to indicate that it is an editor, which is analogous to)ED for)EDIT. Its not obvious that)SE indicates a worksheet editor.

How do you abandon the)EDSS editor session?

Click the Cancel button or the Window's [X] button on the upper right to display the message box:



Then click the No button of the message box.

Is it possible to save the)EDSS editor sessions in progress?

No.

There is no shortcut or quick way to empty the Command Line window

With focus on the Command Line, use Ctrl+A (select all) and then click the Spacebar or the Delete keyboard key or type in some other text.

Pressing Ctrl+V pastes the clipboard content in the Command window by default

The action of Ctrl+V on the APL64 Developer GUI is to attempt to paste the Windows Clipboard content into the GUI control which has the focus. If the Command Line has the focus, the Windows Clipboard content will be pasted into the Command Line. If an editor has the focus, the Windows Clipboard content will be pasted into the focused editor. If the Classic History pane has the focus, the Windows Clipboard content will be pasted into the Classic History pane, assuming that:

1. menu **Session | History Format | Classic** is user-selected
2. menu **Session | Classic History Format | Is Editable** is checked

By design, pasting into the Classic History pane when it is not editable or pasting into the Sequential, Grouped or Separated History panes is not supported.

Could you at least arrange the tooltips not to appear until I actually MOVE my mouse, instead of it just being reactivated after hitting Enter?

In the next preproduction release, there will be a new menu option, **Session | Tooltips | Initial Show Delay**, to allow you to specify in milliseconds, the frequency in which to show the tooltips while navigating the mouse in the window.

Why does the report result of ⎕MF have 5 columns instead of 3?

In APL64 is a new and default ⎕mf mode 4 that adds two new columns to the ⎕mf output matrix. These columns document the execution time and count for compiling each line of execution. Refer to the System Functions Manual for additional information.

What is the value in the cells that are not populated in worksheet editor?

When the content of a worksheet cell is empty or is in the error state, the value extracted to APL64 is an empty string value (⊂"").

Rank of the array changes in Worksheet Editor when entering values non-contiguously

This is by design due to the limitations of the worksheet which cannot contain/preserve metadata like APL shape or rank. It is up to the APL64 programmer to handle the shape and rank of the result of)EDSS Ctrl+E.

Error messages displayed when selecting the SpreadsheetGear option from the context menu in Worksheet Editor

Those features have not yet been implemented by SpreadsheetGear in the .Net Core environment. A future version of Spreadsheetgear may implement them. If so, the will be incorporated into ⎕EDSS.

Can visible rows/columns can be restricted in worksheet editor?

The features you request for ⎕EDSS and)EDSS are currently not available. Remember that ⎕EDSS and)EDSS are available in the APL64 Developer version only and not in APL64 runtime. ⎕EDSS and)EDSS are not intended for use by end users of applications developed by APL64 programmers. Think of ⎕EDSS and)EDSS as an enhanced replacement for the APL+Win)EDIT of rank 2 or less pure numeric or pure text arrays.

'#' []WI 'caption' 'XYZ' doesn't seem to work - should it?

Per the help document **Compatibility Modifications for Existing APL+Win Applications**, the command '#' ⎕WI 'caption' was replaced with the new ⎕devcap system variable in APL64.
E.g., ⎕devcap←'my caption'

'#' []WI 'hwndmain' reports only a 0. Has this not been implemented?

The '#' ⎕wi 'hwndmain' returns 0 in APL64 because ⎕wi executes in the APLNow32.exe component that is a runtime system that doesn't have a main window. So, to retrieve the main APL64 window handle, you can use an alternate solution like the updated APLFrame function below that is compatible with APL64:

```
    ∇ H←APLFrame;N;I;J
[1]   ⍝ Return handle of the APL64 window.
[2]   ⍝ Assumes the main window is the only one with
[3]   ⍝ a caption that begins with "APL64:" and that its
[4]   ⍝ process id matches ↑⎕SYSINIT.
[5]   I←↑⎕sysinit
[6]   H←⎕wcall 'FindWindow' 0 0
[7]   →(×H←⎕wcall 'GetWindow' H 'GW_HWNDFIRST')↓0
[8]   :Repeat
[9]       (N J)←⎕wcall 'GetWindowThreadProcessId' H (,0)
[10]      :If I=J
```

```
[11]          N←↑↑/⎕wcall 'GetWindowText' H (64ρ⎕TCNUL) 64
[12]          →(1↑N ⎕SS 'APL64:')/0
[13]      :EndIf
[14]  :Until 0=H←⎕wcall 'GetWindow' H 'GW_HWNDNEXT'
[15]
    ∇
```

The alternative method is to use **9⊃⎕sysinit**. See the comments to Updates to ⎕SYSINIT above.

Are APL64 files the same as .sf files?

Yes. The same applies to the colossal (.cf) files.

⎕CF-fns with floating point tie numbers (e.g., 1.5-.5) returned DOMAIN ERROR

This issue will be fixed in the next pre-production release of APL64.

Option for a traditional APL workspace environment in APL64

The history pane is the region of the main window where the executable APL statements and any associated results are illustrated during an APL64 or APL+Win instance. In APL+Win there is only one format for the history pane and the history pane is editable. Being editable, it is a programmer scratch pad that may not reflect what APL statements were actually executed during the APL+Win instance. In APL64 there are several formats available for the history, including a format which behaves similarly to the APL+Win editable history pane.

To enable the editable history pane format in APL64, make these selections from the APL64 menu:

- Session | History Format: Classic: selected
- Session | Classic History Format | Is Editable: checked
- Session | Classic History Format | Focus When Editable: checked

When these selections are made, APL64 Command Line will remain visible, but its use will be optional, because the focus will remain in the editable classic history pane in most circumstances. A screen cast would be good too.

It's no longer possible to set/remove stops in the margin in the function editor

This was a bug in the pre-production 2020.0.4 release.

Ability to debug (with stop and trace) functions on the go

In APL64 setting stops and tracing function execution is the same as in APL+Win. The debugger is used to display APL functions or expressions while they are executed step-by-step (tracing execution) and when a function stop is in effect (execution suspended). In APL64 the debugger is the analogue of the APL+Win Code Walker. In APL+Win the code walker is manually shown or hidden based on the APL programmer's discretion. In APL64 the APL programmer can make a choice to manually show/hide the debugger or automatically show/hide the debugger.

In APL64 to automatically show/hide the debugger, check the **Session | Debugger | Automatically Show/Hide Debugger** checkbox. If the APL64 programmer wants to manually show/hide the debugger, like in APL+Win, uncheck this menu item and check/uncheck the **Session | Debugger | Show Debugger** checkbox as desired.

In APL+Win, I cannot place a Stop on a comment line of a function, except for line [1], which is different from APL64

Unlike APL+Win, there is no restriction in APL64 with setting stops on any empty and comment lines in user-defined functions.

[]PENCLOSE returned SYSTEM ERROR:Aval.Categorize encountered an uncategorized child element at index=0

This issue will be investigated and fixed in a future release of APL64.

What is the latest info on support for Inline Control Structures (ICS)

Inline control structures (ICS) are not yet supported in the APL Pre-Production release. We are not yet sure if it will be supported in time for the production release.

We would like to get feedback on how important ICS features are to our users. If we can eliminate them from the system, we can consider alternative design possibilities that might use embedded colons in a different way such as for declaring named arguments in argument lists such as:

```
Foo(Name: "John"; Address: "Somewhere")
```

Note: This response has been superseded by the first announcement under the **Enhancements** heading above.

Is there an alternative to □wi based forms in APL64?

At this time it is not anticipated that there will be another GUI development tool within APL64. Such a tool would rapidly become out-of-date, just like []wi-based forms.

It is planned, but not yet available, in APL64 that an APL64 programmer may create a workspace containing at least one 'public' function and use an APL64 Developer version utility to create a .Net assembly so that the public APL64 function(s), can be called as static methods of that assembly by a containing .Net solution. What this feature will accomplish is make it possible for a programmer to implement an application GUI in based on any .Net GUI development technology and use APL64 to support the algorithms and business rules of that application.

'APL64 as a .Net assembly' is analogous to the existing menu **Options | Create .Net Runtime Assembly | Create Windows Runtime Executable**, which is currently available.

Does APL64 come with supplied workspaces?

The ASMFNS.ws64 workspace is currently distributed with this pre-production release of APL64. It is installed in a folder named Tools in the path to the OS-specific User's folder identified by the system variable □userpath in APL64, e.g., C:\Users\John\AppData\Roaming\APLNowLLC\APL64\Tools

We will investigate which workspaces from APL+Win will be suitable for inclusion with APL64 and explore other new workspaces that are specifically designed for APL64.

FTIMEFMT looks like a user-defined function. How do I get the FTIMEFMT function?

You can obtain the FTIMEFMT function from the TOOLS\DATES.W3 workspace included with APL+Win.

The :goto control structure statement does not recognize labels

The next APL64 preproduction release will resolve this issue.

There is an awful flicker (debugger panes) when pressing F10 during a debugging session

The excessive flicker in the debugger panes will be eliminated in the next pre-production release of APL64.

Are you sure APL64 is supporting custom classes, e.g., zObjects?

Custom classes will be supported in the next pre-production release of APL64.

Is the pre-compiled version of the APLNow32.adf file provided in APL64?

Yes, with the production release of APL64. But being that these files are the same in APL64 and APL+Win, in the interim you can use the APLWADF.INI included with your APL+Win release.

Would it be possible to make the Alt+Shift+End key work like it does in APL+Win, i.e. erase from cursor to end of APL Session?

Your suggestion will be available in the next pre-production release of APL64. There will also be support for the Alt+Shift+Home shortcut for erasing all characters from the current caret position to the beginning in the history and editor.

Commenting 200 lines in the function editor with Ctrl+/ takes longer in APL64

A correction for this issue will be available in the next pre-production release.

Moving a bunch of lines to the right or left with Tab & Shift+Tab takes longer in APL64

A correction for this issue will be available in the next pre-production release.

Moving a bunch of lines one space with Ctrl+Spacebar & Ctrl+Shift+Spacebar takes longer in APL64

A correction for this issue will be available in the next pre-production release.

APL component files can be lost when tying a file with a tie number already in use

Tying a component file with a tie number that was already in use normally resulted in a FILE TIED error message. The problem was that in addition to the error message, the file being tied was also deleted. This will be corrected in the next pre-production release.

User Defined Control works in APL64

I tried what was posted at http://forum.apl2000.com/viewtopic.php?f=15&t=416 (Building C# User Controls for APL+WIN) with APL64. And pleased to see that it works.

⎕dr ↑0⍴3.14 returns 645 instead of 323 like in APL+Win

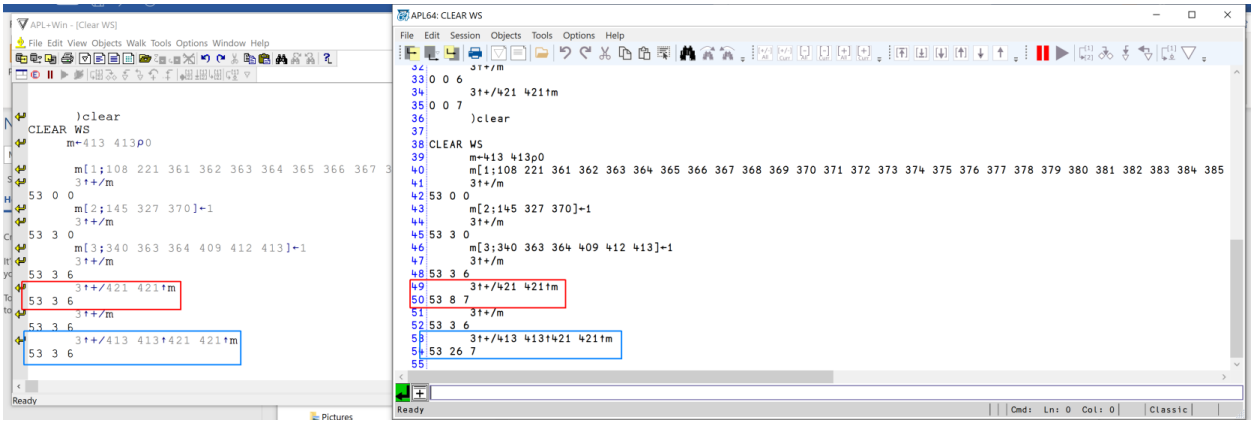
This bug will be fixed in the next pre-production release.

Why am I getting all-0's out of this formatting example: 1 0 ⍕⍤0 1 0 1 0 1

It will be investigated and corrected in a future APL64 release.

Issue with the take primitive function for packed Booleans

The following demonstrates the differences between APL+Win and APL64:



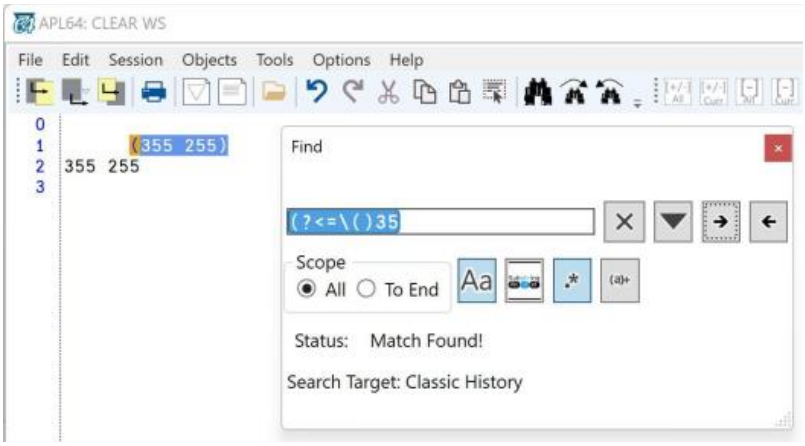
This bug will be fixed in the next pre-production release.

Tooltips remained visible after Alt+Tabbing to another app

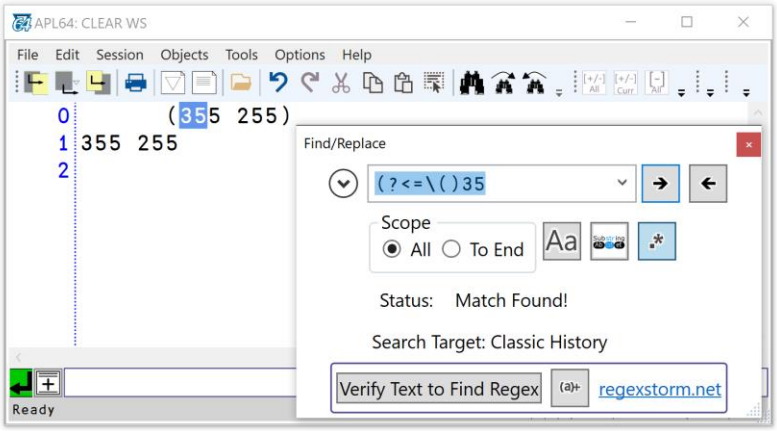
We haven't been successful in duplicating it here, even on a slow machine. If you come up with a reproducible scenario, please share it with us.

Incorrect highlighting in the target for regular expressions in the Find dialog

E.g., The find result highlight is wrong below:



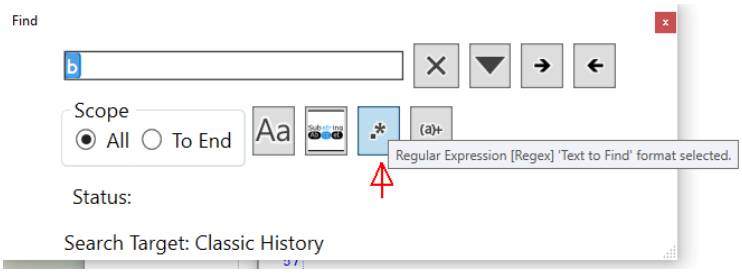
The following demonstrates the correct behavior, which will appear in the next pre-production release:



The fix for this will be in the next pre-production release.

Are there limitations to the Find dialog Regular Expressions?

Your example illustrates that you have not selected the Regex filter because the 'Wild' toggle button is visible. Click the toggle button repeatedly until the Regex icon is shown. If you have all tooltips on, the tooltip for this toggle button will also be updated. E.g.,



Regular expressions in the Find/Replace dialog can easily crash APL64

Thank you for reporting this issue. Coincidentally, the same error was discovered by us when investigating your previous issue with regular expressions in the Find dialog. It's now been corrected for implementation in the next pre-production release.

Ctrl+ mouse click a Recent Workspace Id does not)XLOAD the workspace like in APL+Win

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

Results for □AT, □CFHIST, □CFRDCI are wrong by two hours (relative to □TS)

Thank you for reporting this issue. This will be investigated, and any modification will be considered, if necessary, in a future release of APL64.

Is there a way to run a function in the background i.e. asynchronously?

Not currently. This may be possible in APL64 when there is a suitable replacement for the APLNext Supervisor in APL+Win; i.e., a APL64 Supervisor.

□vr has the wrong indentation (i.e., missing leading spaces) starting at line 1000

This issue will be fixed in the next pre-production release.

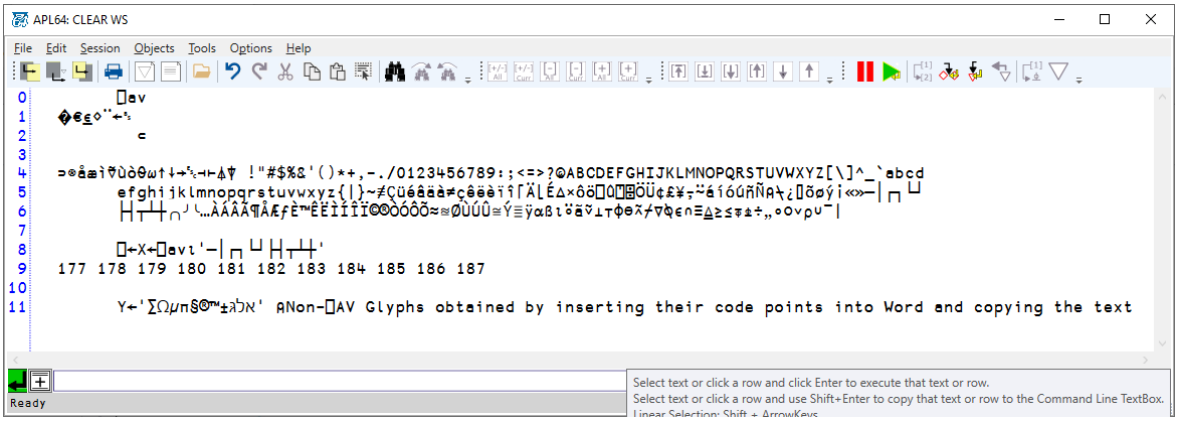
How Unicode aware is APL64? Which APL fonts, if any, has the boxing characters?

In APL64, in contrast to APL+Win, any Unicode code point may be displayed by APL64Developer version GUI. This Unicode 'aware' enhancement in APL64 refers to the ability to render Unicode codepoints to the display, printer, etc. by including such Unicode code points in displayable APL64 variables. In APL64 the rendering of Unicode code points relies on the APL385 Unicode monospaced font or for any user-selected font's ability to properly render Unicode code points.

The Unicode 'aware' enhancement in APL64 does not imply that □AI contains all Unicode code points. □AI is a well-defined list of Unicode code points which does not contain all Unicode code points.

For Unicode code points that are in □AI, they may be used by indexing into □AI. This is a 'convenience' property of □AI. For those Unicode code points which are not in □AI, the code point must be provided by the APL programmer in the form of an APL variable which can then be rendered by APL64.

For example:



Note that □WI, a Win32 technology, in APL64 has the same features and limitations as in APL+Win with respect to Unicode code points.

The ~ (not) symbol and the – (minus) symbol look a bit too much the same in the APL Session and in APL64 programs

APL64 uses the APL385 font. The specifications of the APL385 font are not under the control of APLNow, LLC. The APL385 font is the only available font which is non-proportional and maps all the APL glyphs. The legibility of the font on a particular workstation may be influenced by:

1. The resolution of the monitor(s) used with APL64
2. The user-selected value of menu **Session | View Scale APL Text**

We do not anticipate any modification to APL64 which will change the behavior of this issue.

I can no longer display the ⍨ (Commute) symbol on my keyboard

We have not been able to duplicate this issue here. If 'another app has registered this hotkey', it is up to the Windows user to modify the properties of such an app to avoid the conflict with APL64.

Questions:

1. Do you observe this behavior in APL+Win?
2. What APL64 keyboard definition has been selected?
3. What Windows language preference has been selected?
4. If you identify the offending application, please provide the app's name to use so we can note it in the APL64 documentation.

Will APL64 will be compatible with EVLEVEL 1?

Yes.

Error Message: "SYSTEM ERROR:Aval.Categorize encountered an uncategorized child element at index=1"

This bug that caused this error will be fixed in the next pre-production release.

Error Message: "SYSTEM ERROR:Exception: CS0023: Operator '!' cannot be applied to operand of type 'int'"

This bug that caused this error this bug will be fixed in the next pre-production release.

Could you improve the look and feel of horizontal splitters [in APL64]?

The definition of 'nice' is a personal preference. Splitters in WPF can be custom designed, so it might be possible to revert to the older Window Forms splitter style, which may be more familiar to you, however it would be best to see if any other APL64 users comment on this issue. It would not be good to revert to the APL+Win style splitters which are difficult to use on modern high-resolution displays.

Note also that the vertical 'splitter' between the debugged code pane and the state indicator pane are integral to the pane control GUI control and cannot be used elsewhere.

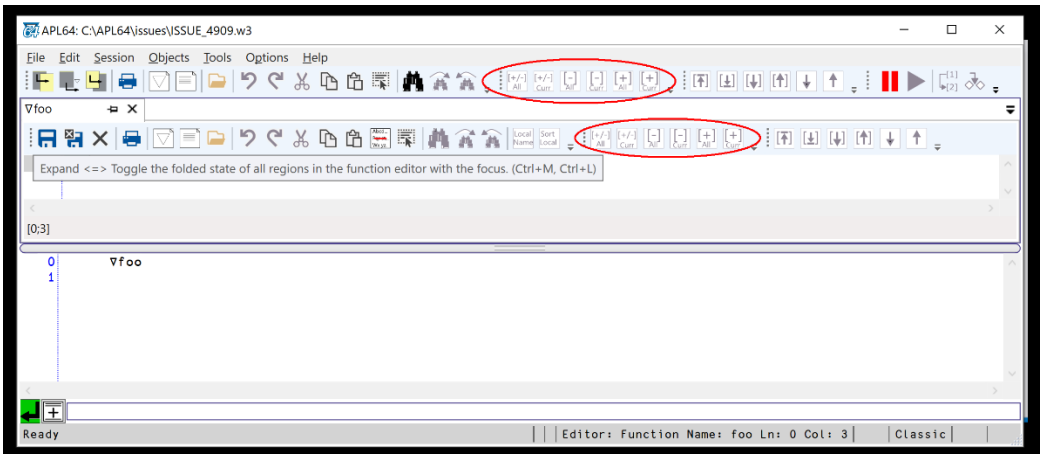
Can the closing message box and dialogs be centered over the APL64 application?

The closing message box is centered in the display, not the window. The reason being the .Net managed code option for 'on top' centers the message box it in the 'primary' display.

It's also not recommended to use a C# class to center MessageBoxes and Dialogs over their parent window because this would implement unmanaged code (DllImport).

Outlining shortcut keys are missing in APL64

The Outlining options are not on the Edit menu but in the toolbars of the main APL64 session and the function editor as demonstrated below:



The practical reason for this is that when a function editor is undocked and floated, it would not have direct access to the Edit menu to invoke the Outlining options. So, for convenience sake, the decision was to make them available only on the toolbar.

We welcome any comments you have about this.

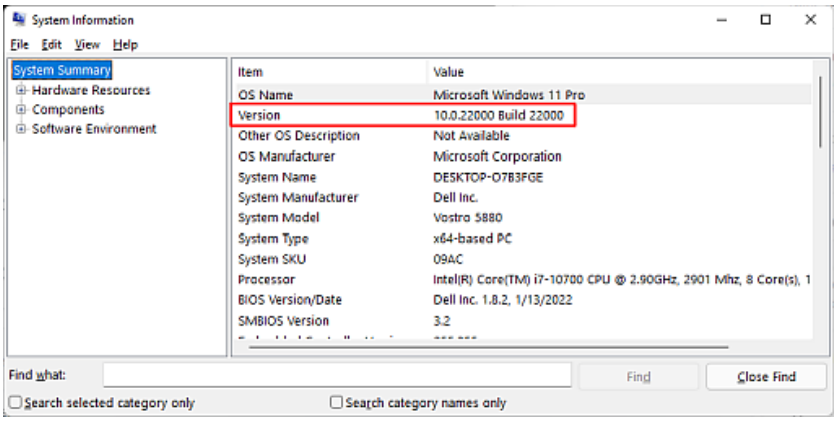
Ctrl+Z and Ctrl+Y shortcuts work in the Find/Replace form to be synonyms as clicking the Undo and the Redo buttons

If you're suggesting that these shortcuts should operate when the focus is in the Find/Replace dialog, then I believe you're mistaken because this isn't something that is currently supported in APL+Win or other typical Windows applications we tried with a similar Find/Replace dialog. In any case, these shortcuts do function when the focus is moved to the editor with recent changes.

The only application tested that did what you suggested was Visual Studio 2019. But Visual Studio doesn't use the typical Find/Replace dialog but some specialized window that is built into the editor and not floatable. We could explore whether a similar Find/Replace window could be adapted to work in some future release of APL64.

Is □os in Windows 11 returning the correct result?

Yes. The version of Windows 11 is 10.0.22000 as demonstrated in the image below captured on a Windows 11 machine:



Also, the digits after the second period does match the build number.

Interfacing APL64 with C#

You can still use ⎕cse to call C# methods/properties from APL64. ⎕CSE is integrated into APL64 and has excellent performance. Refer to the C# Script Engine documents in the menu **Help | APL Language**.

As for calling APL64 functions from C#, that information is currently under development. When this feature is available in APL64, the menu **Options | Create .Net Runtime Assembly | Create Platform-independent Runtime Library** utility will be the mechanism to call APL64 from C#.