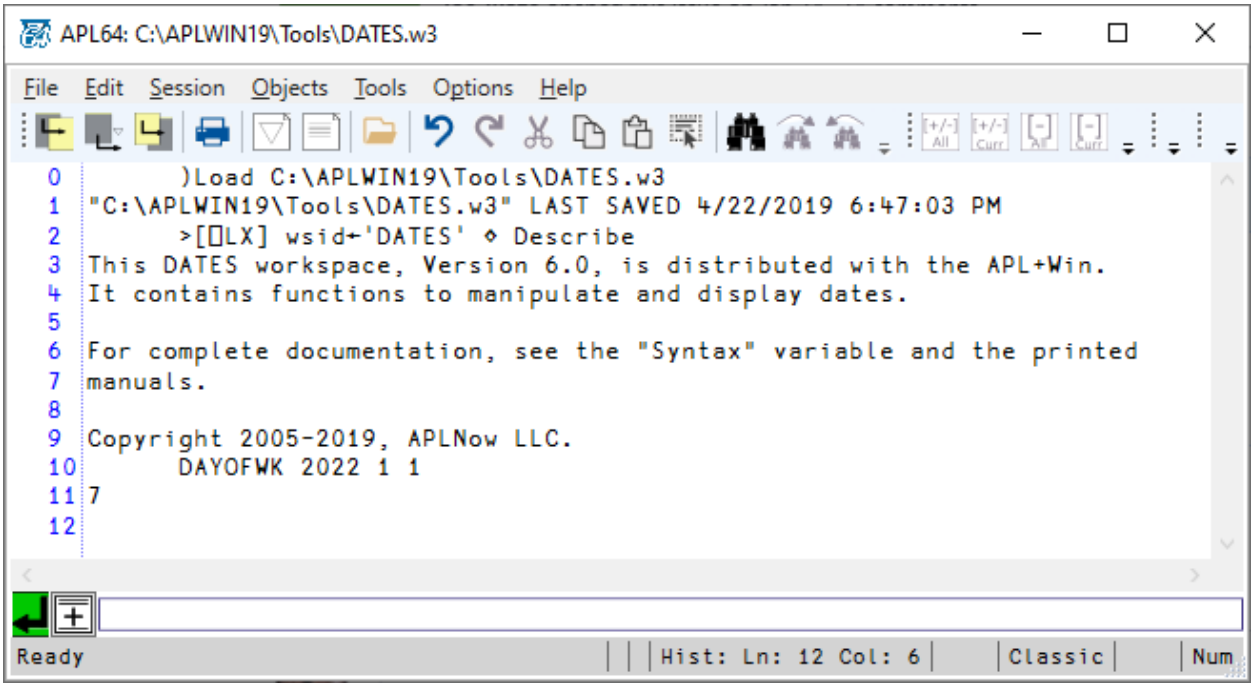
Advantages of APL64

Contents

Excellent Compatibility with APL+Win

Load and run an APL+Win workspace in APL64 in many cases with minimal or no modification required. The Help > APL+Win Compatibility menu item describes this feature.



Increased Workspace Size Up To The Available Workstation Memory

APL64 – Maximum workspace size up to available workstation memory permitting the creation and processing of multiple large APL variables.

APL+Win - Maximum workspace size less than 3.7 GB.

Increased Variable Size

APL64 - Maximum homogeneous variable size up to 2,146,435,071 elements in a variable. Nested arrays may recursively contain up to this number of elements at each nesting level.

Type	Max GB

Boolean	2
Integer	8
Double	16
Unicode Char	4
APL+Win Char	2

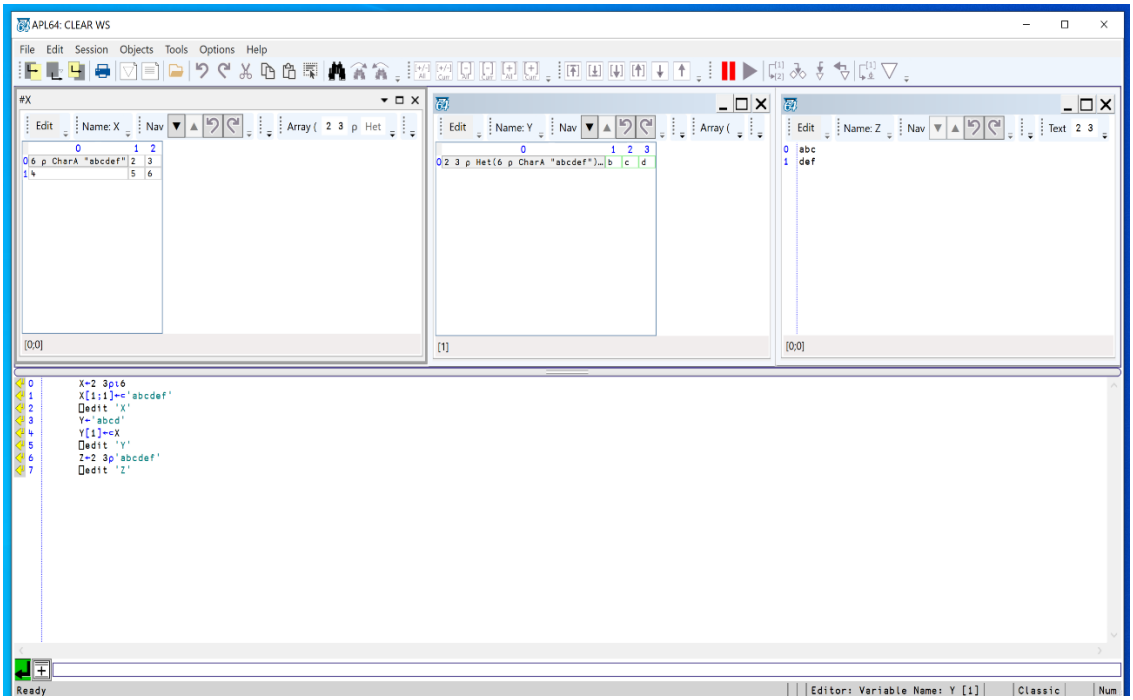
APLWin - Maximum homogeneous variable size less than 2 GB and in practice 500 MB or less.

Performance Optimizations of APL Primitives and System Functions

After an APL64 Project primitive or system function is implemented and tested, it is considered for performance optimization. The optimization process requires considerable effort and skill. These optimizations have yielded significant performance improvements. Click here to view graphs comparing APL64 performance to APL+Win in many areas. Performance optimizations are an ongoing process.

Unified Variable Editor To Review & Edit Simple, Homogeneous, Heterogeneous, and Nested Variables

The new unified variable editor in APL64 can review and edit an APL variable of any type, including heterogeneous and nested variables. The APL+Win graphical editor is limited to simple text or simple numeric variable editing.



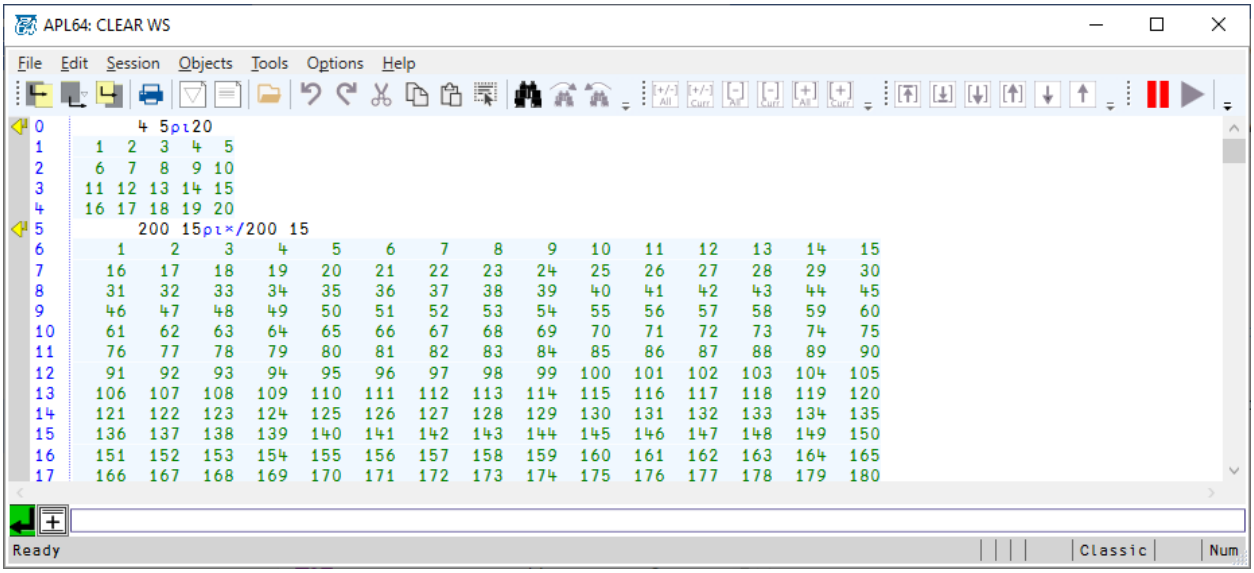
Full Unicode Support

Transparent support for Unicode characters in character scalars, vectors, matrices and arrays throughout the system. For compatibility ⎕AV has the same content as APL+Win.

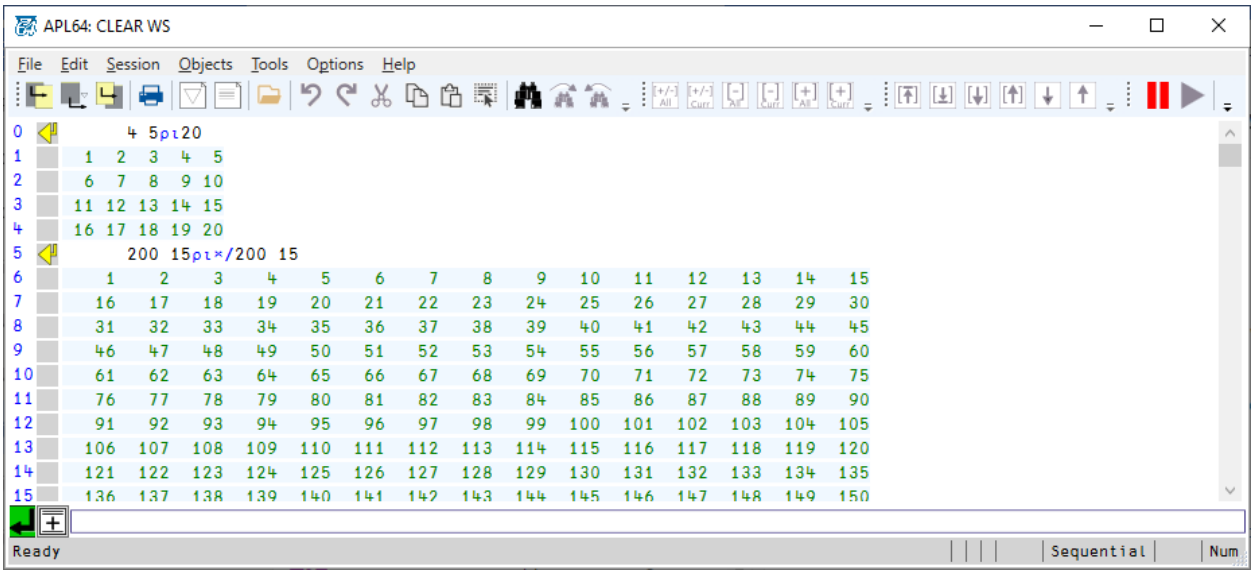
Multiple Formats to Display the History in the Developer Version

Four formats are available for the history pane in the APL64 developer version which may be selected at any time during an APL64 instance to illustrate previously-executed APL statements and resulting interpreter output.

Classic – This format is analogous to the APL+Win 'session' document-style format with the information in execution order. The APL64 classic history format has an 'Is Editable' option like in APL+Win.

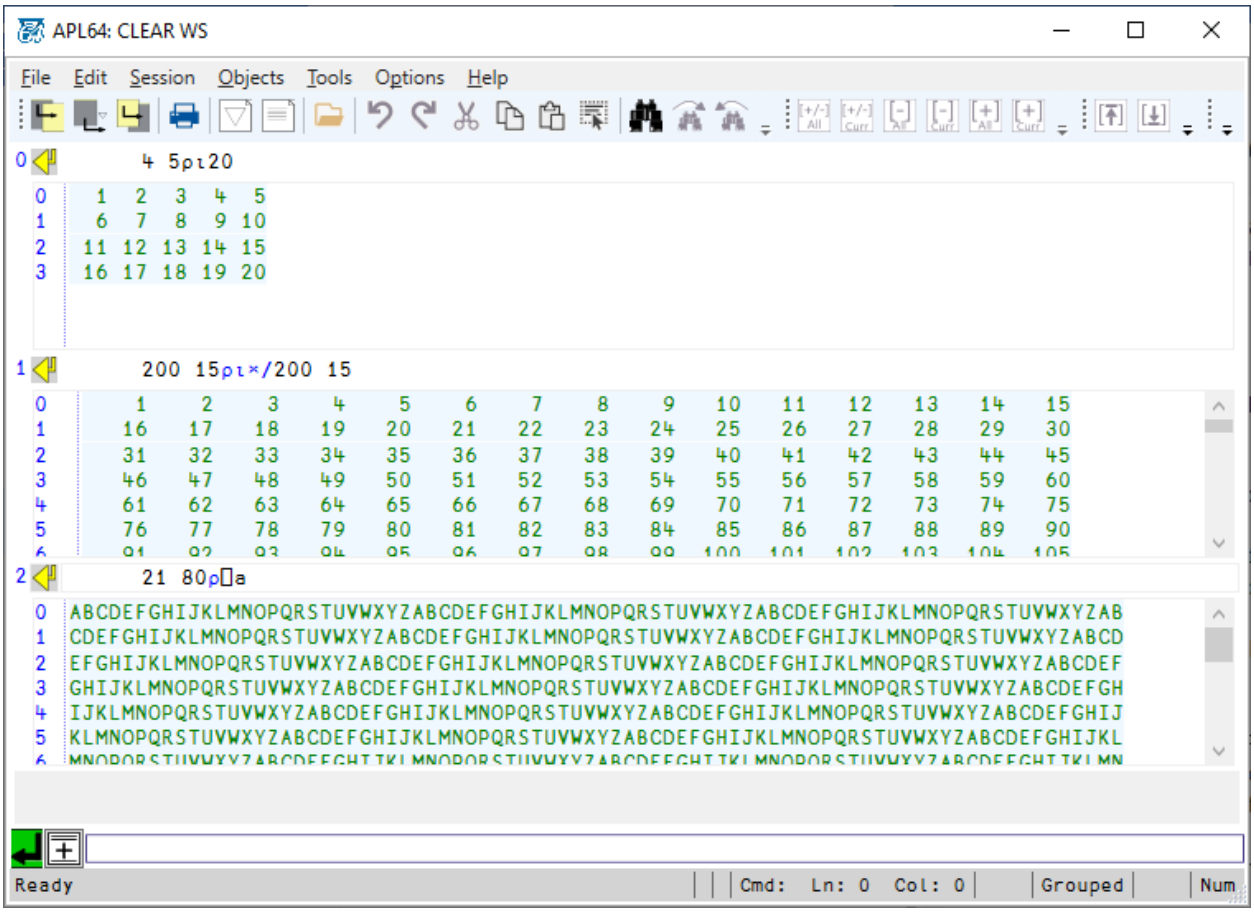


Sequential – Developer input and interpreter output is displayed as an array of rows.

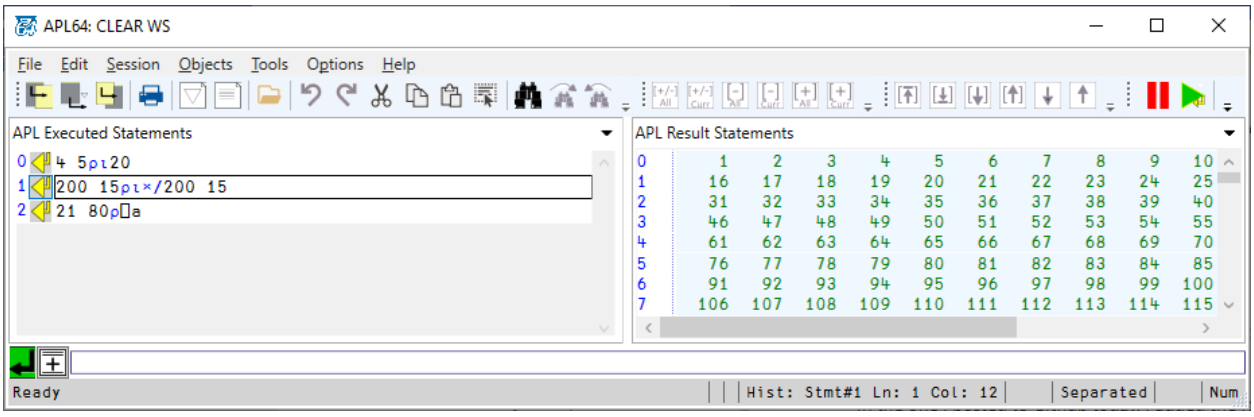


Grouped – The executed APL statements are displayed as an array of rows. The interpreter output can be displayed as an array of rows or in document-style. When the interpreter output

is extensive, it is presented as a scrollable, user-specified-height text block. The results text block will be scrolled to the first or last result line depending on user preference.

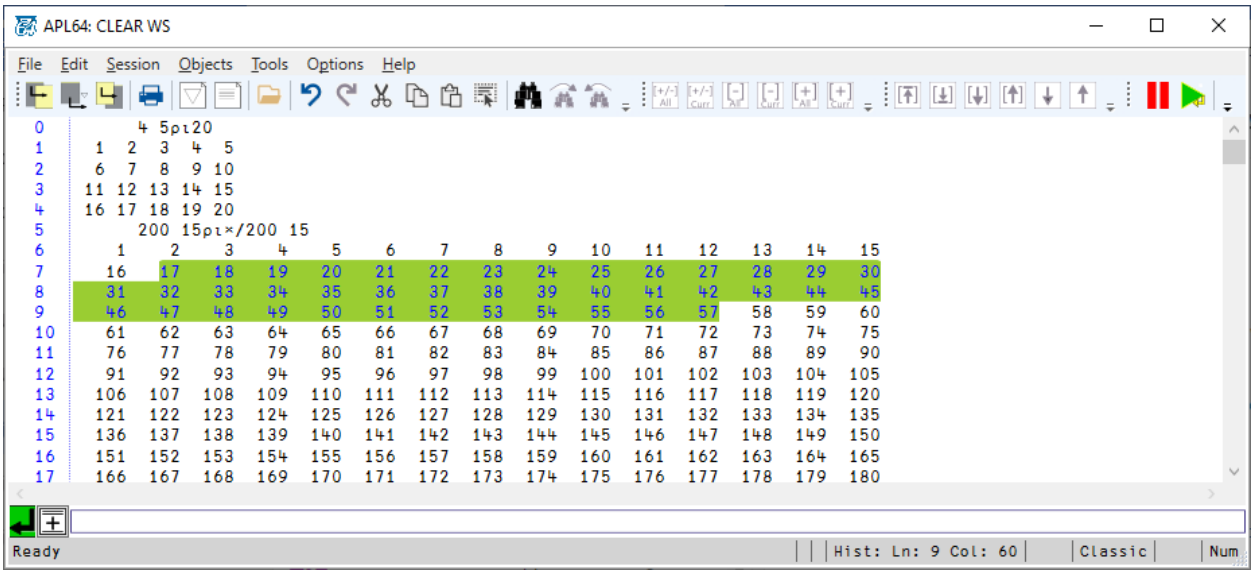


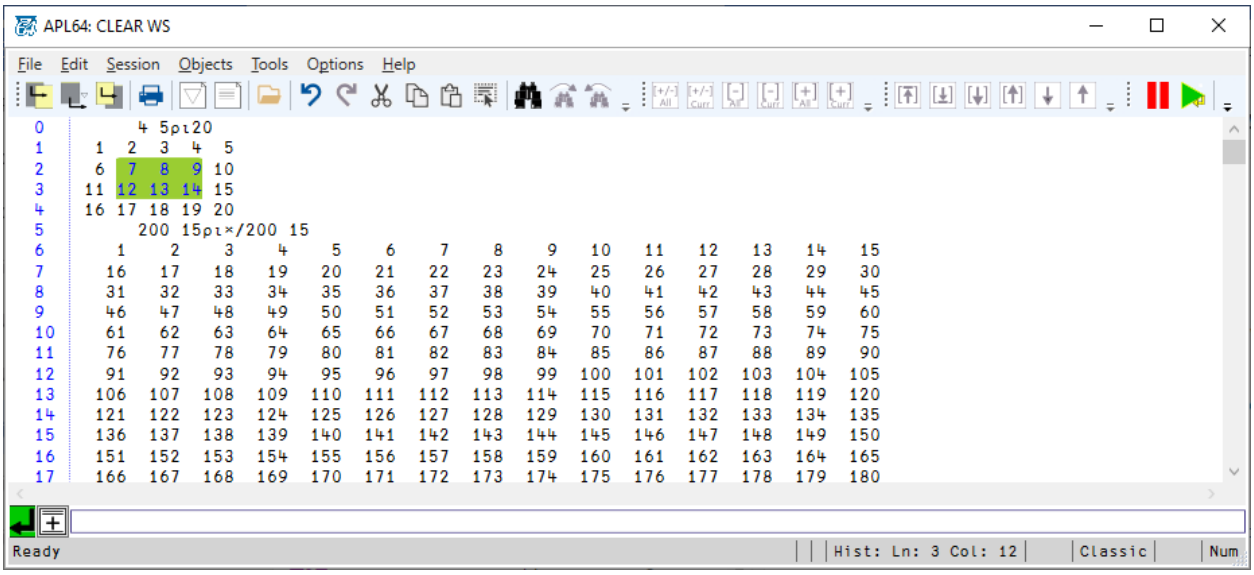
Separated – The executed APL statements are displayed as an array of rows. The interpreter output for a user-specified executed APL statement can be displayed as an array of rows or in document-style.



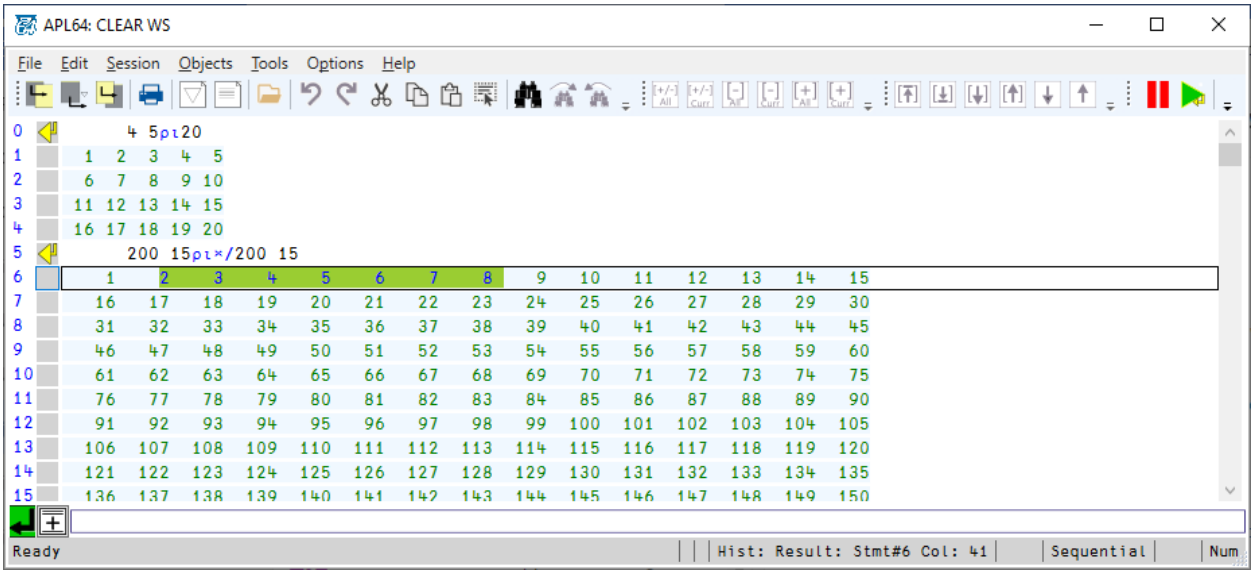
Additional Text Selection Options - Linear, Rectilinear and Row

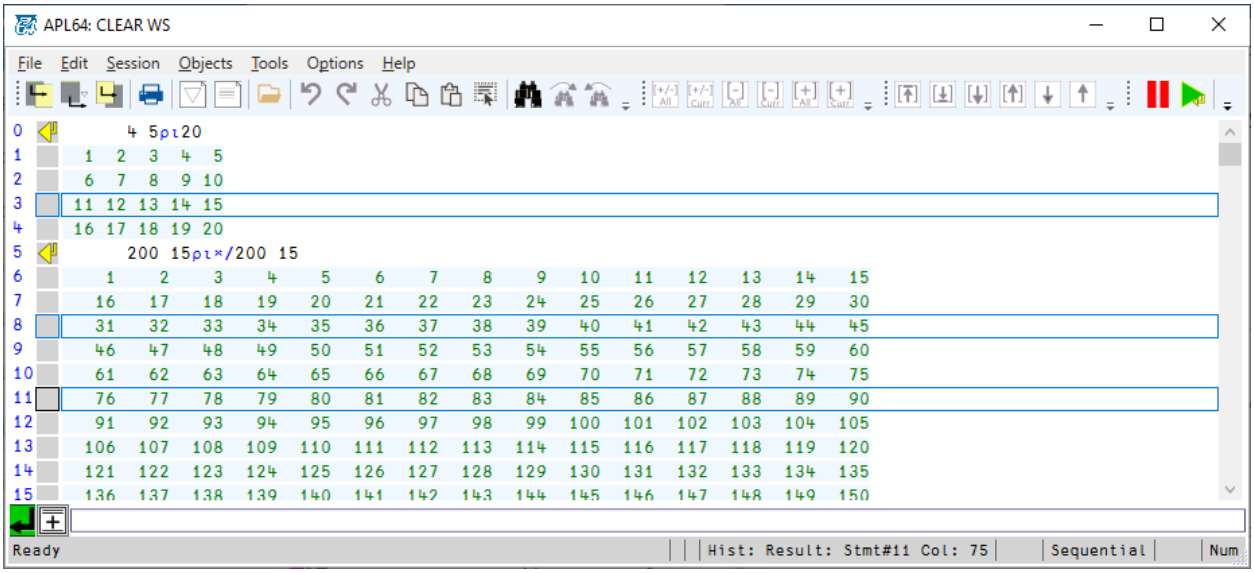
The information in the function and variable editors, the debugged function and some user-selected history formats is presented in document style which supports linear, contiguous selection and rectilinear selection.





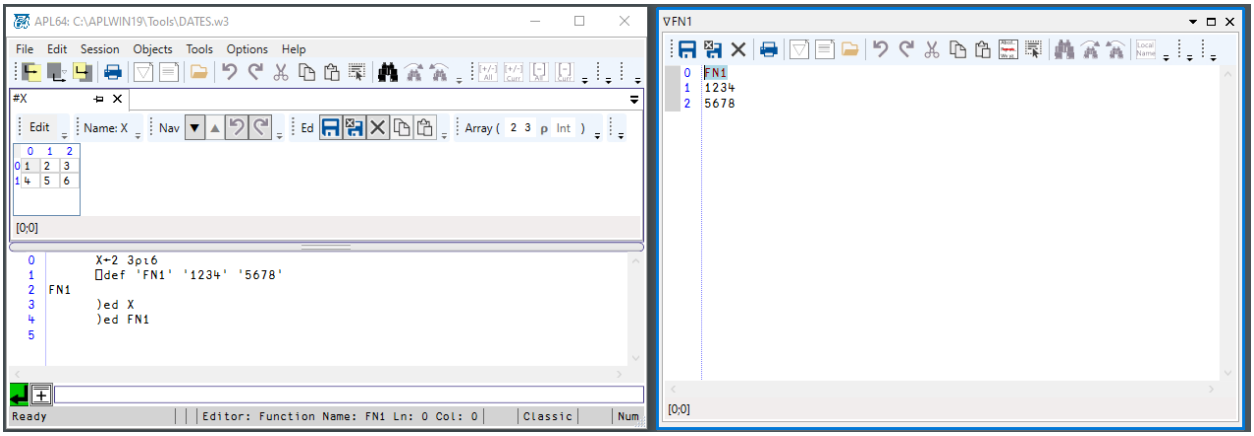
Some user-selected history formats present the information in row-style supporting linear, contiguous selection and rectilinear selection within a row and non-contiguous whole row selections.



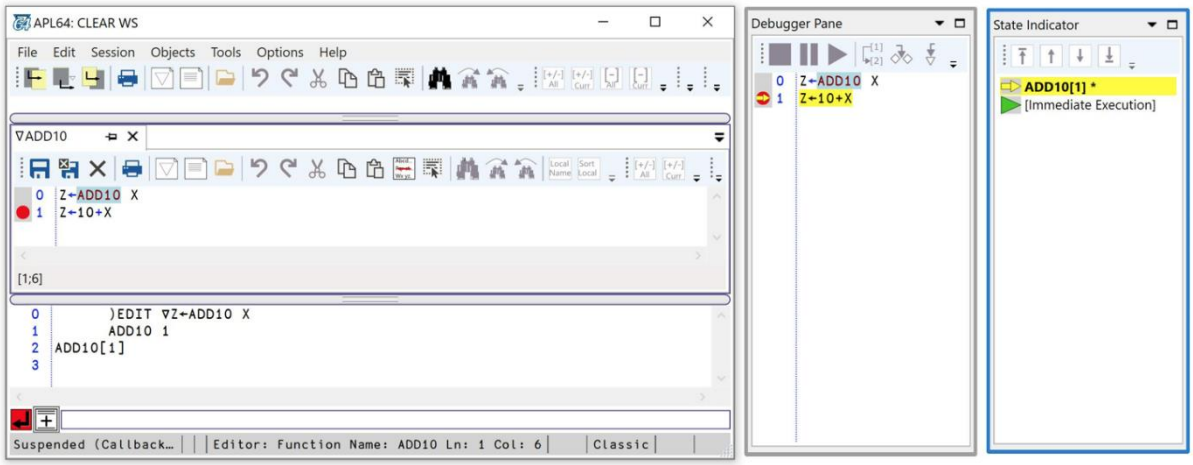


Floating or Docked Panes – History, Function &Variable Editor, Debugger

The variable editor and user-defined function editor panes may be floated separately from the main window.



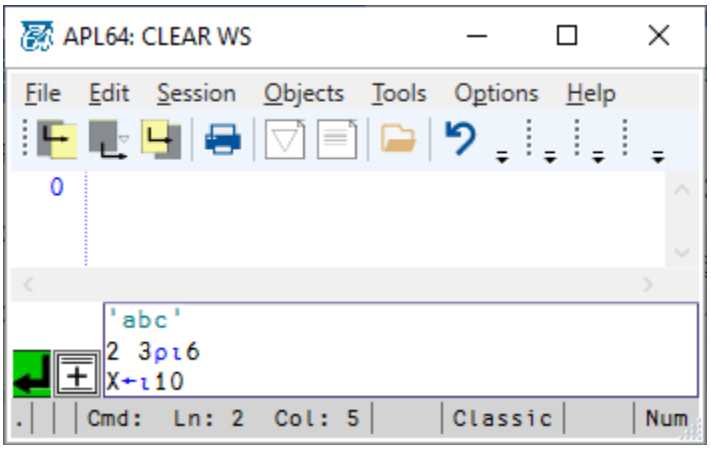
The debugged function pane and the state indicator panes may now be independently 'floated' separately from the main window:



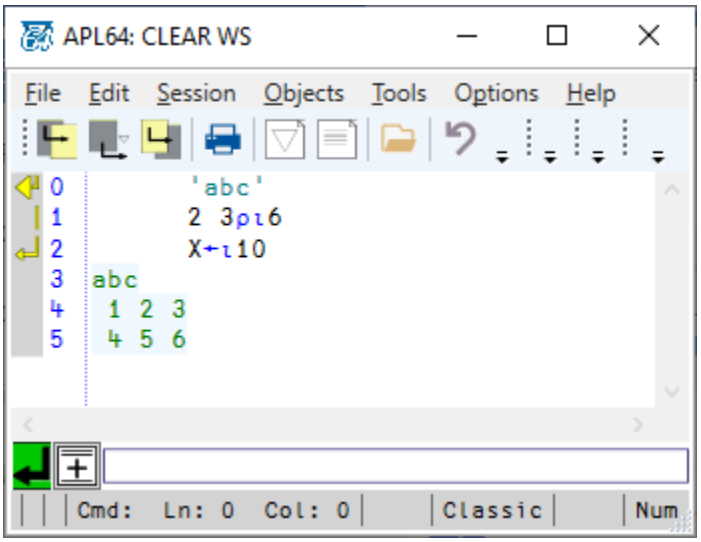
Multi-row Executable APL Statements

Often it is convenient to associate multiple APL executable statements into a 'script' which is executed *en masse*. A new line character is used to separate the rows of a multi-line executable APL statement to improve readability.

Multi-row APL executable statements may be manually entered or pasted into the APL64 Command Line.



Once executed, multi-row APL executable statements are identified in the history pane as a unit.



Runtime Deployment Simplified – APL64 Application in a Single File

There is no need for an APL64 runtime version or workspace in APL64. In APL64 the Options > Create Runtime .Net Assembly > Create Windows Runtime Executable utility will package application workspaces and other supporting application files into a single executable file which includes the APL64 engine.

All features of APL64 which are available in the Windows operating system environment may be used by the APL64 programmer in the development of an APL64 runtime application.

The APL64 programmer creates the main APL64 workspace containing an ⎕LX to start the application. This APL64 workspace and other application-specific files are combined with APL64 engine into a single, easy to deploy 'runtime' file containing:

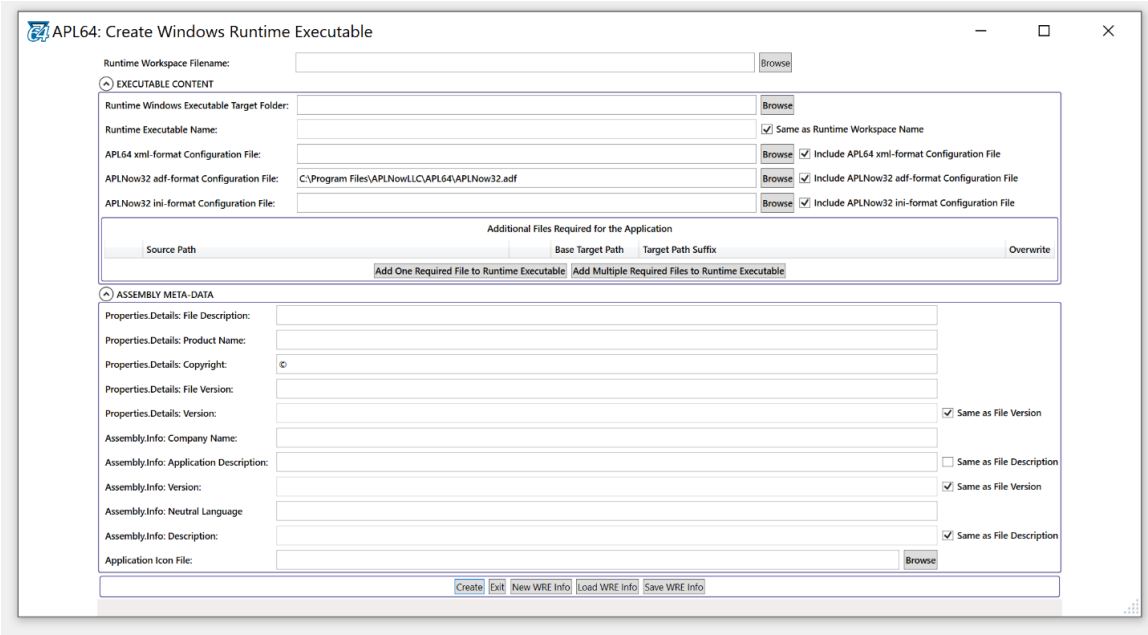
- APL64 runtime component
- APL64 xml-format configuration file
- APL programmer-created, runtime-ready APL64 workspace

• .Net components for the Windows operating system

• APLNow32 components to support the Win32 features of APL64 including ☐WI, ☐WCALL, ☐NI

• Other APL programmer-provided application-specific files

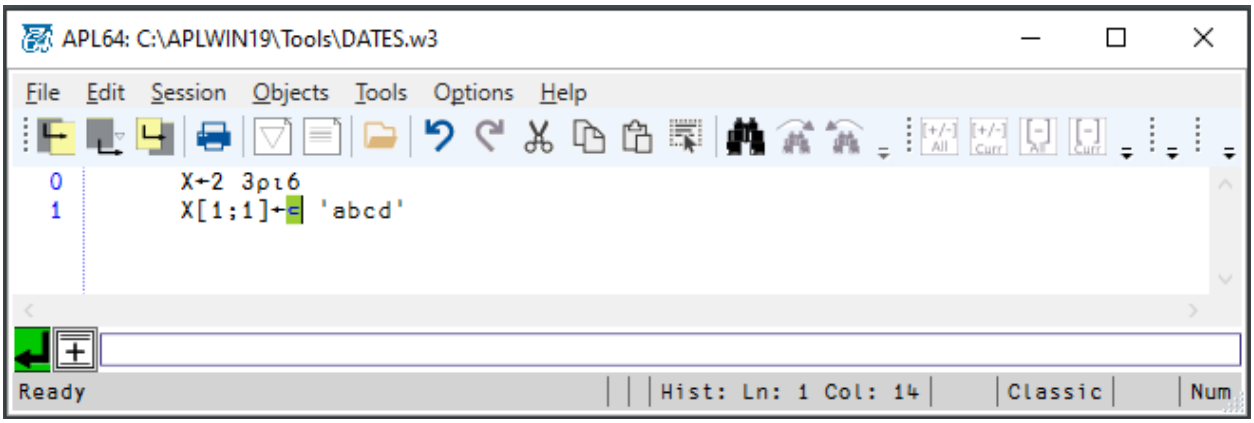
The published APL64 runtime executable is self-contained, including all of the necessary components of a complete APL64-based Windows application, so that it can be run via a double left mouse click or incorporated into a Windows program short-cut.

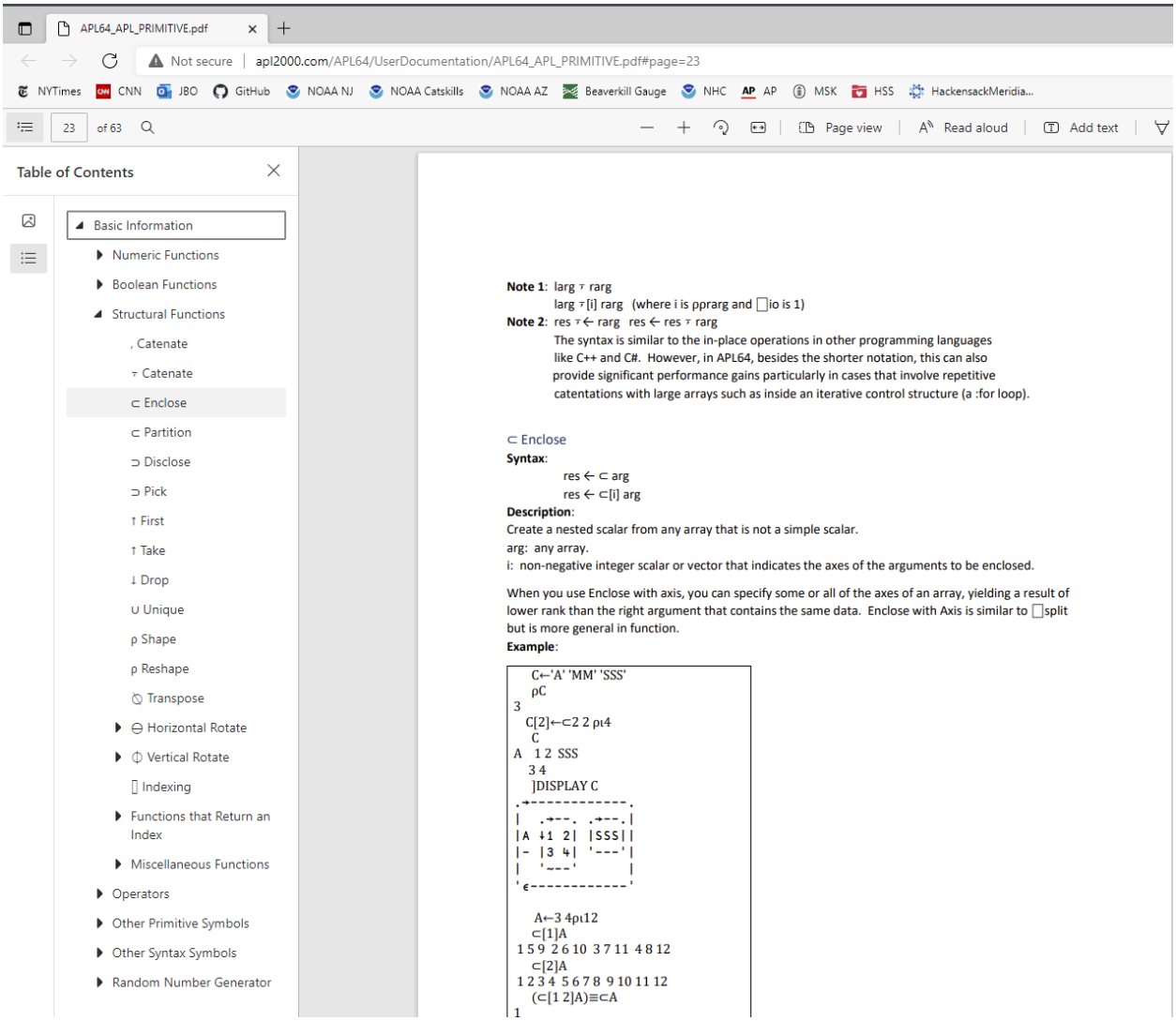


Context-Sensitive APL Developer Documentation

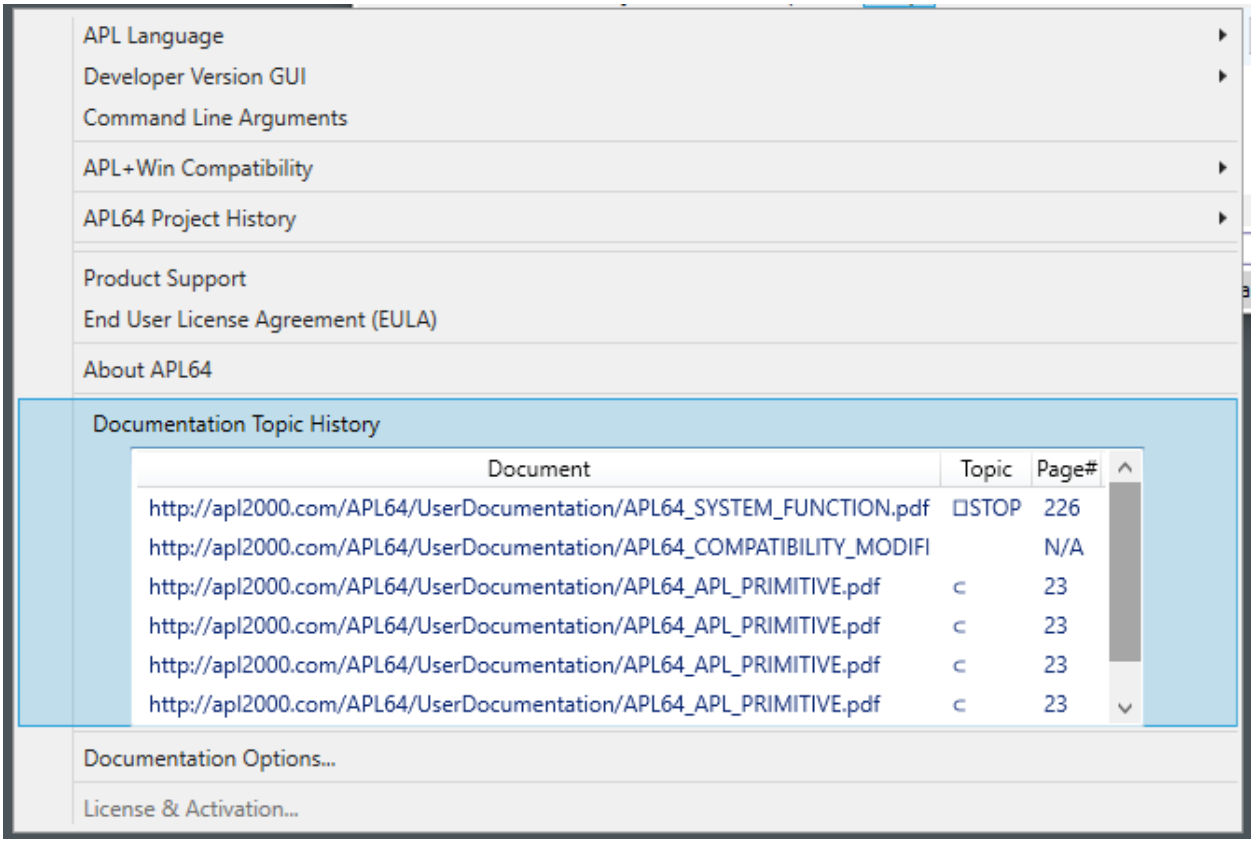
Context-sensitive documentation for APL primitive functions, system variables, functions and commands has been implemented in APL64.

When the cursor is placed on an element in the history pane, function editor or debugger and the F1 key is clicked, the applicable documentation for that element will be displayed in the user's default web browser. The pdf-format documentation supports bookmarks and table of contents containing APL glyphs and APL glyph names. This feature is new to APL64 and is not available in APL+Win.



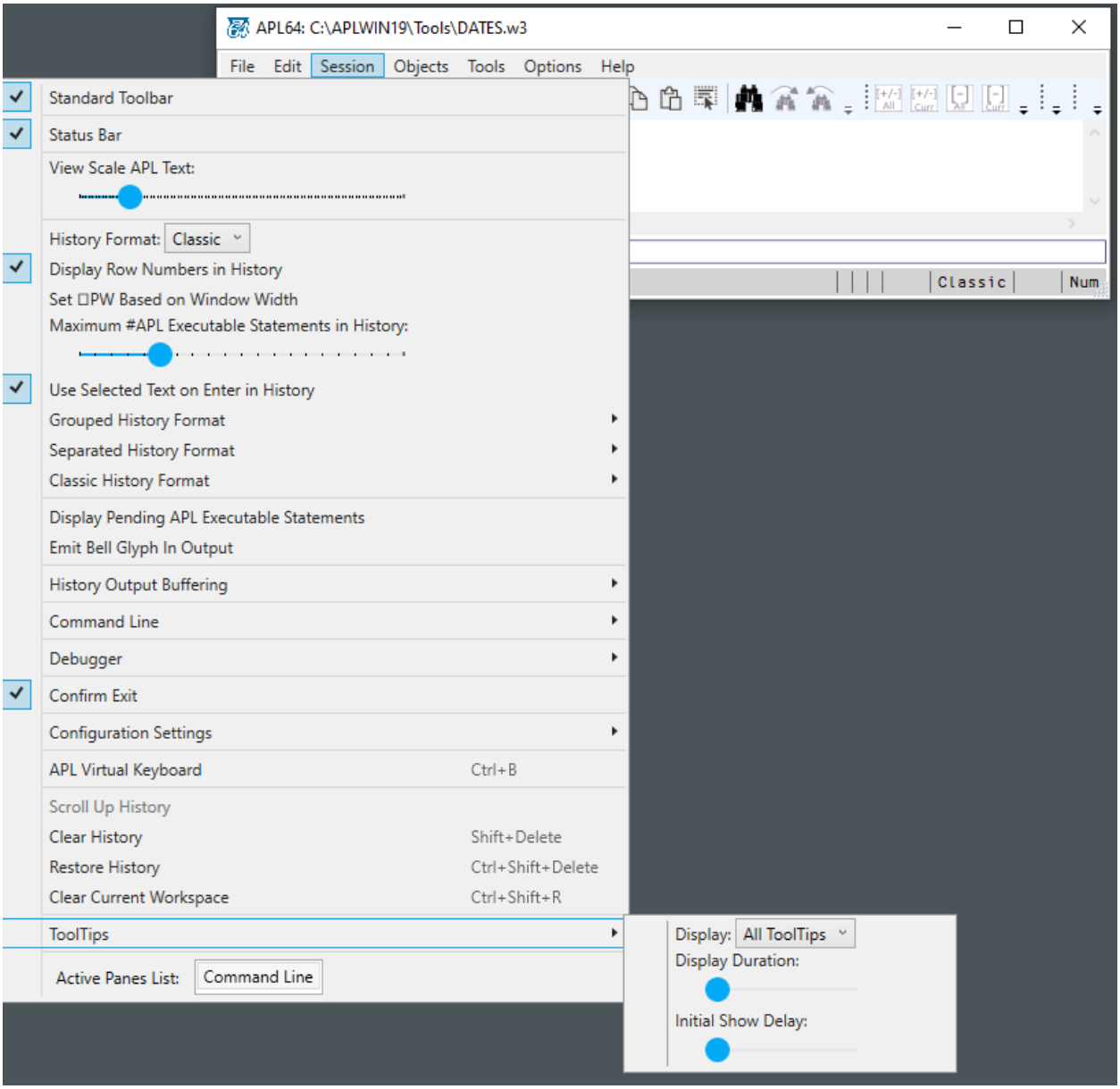


The Help > Documentation Topic History list is maintained during an APL64 instance. Double left click on a user-selected list item will open the associated document in the user's default web browser.

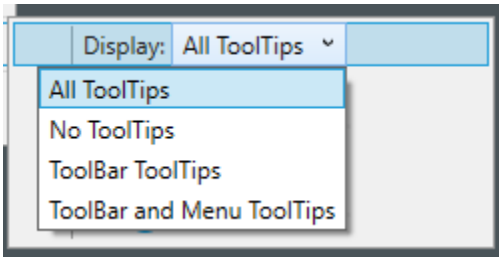


Optional Tooltips for Toolbars, Menus and Panes

Depending on developer preference tooltips are presented when the mouse hovers over a toolbar button, menu item, input field or history, editor or debugger panes in an APL64 developer version instance.



• Tooltip display options



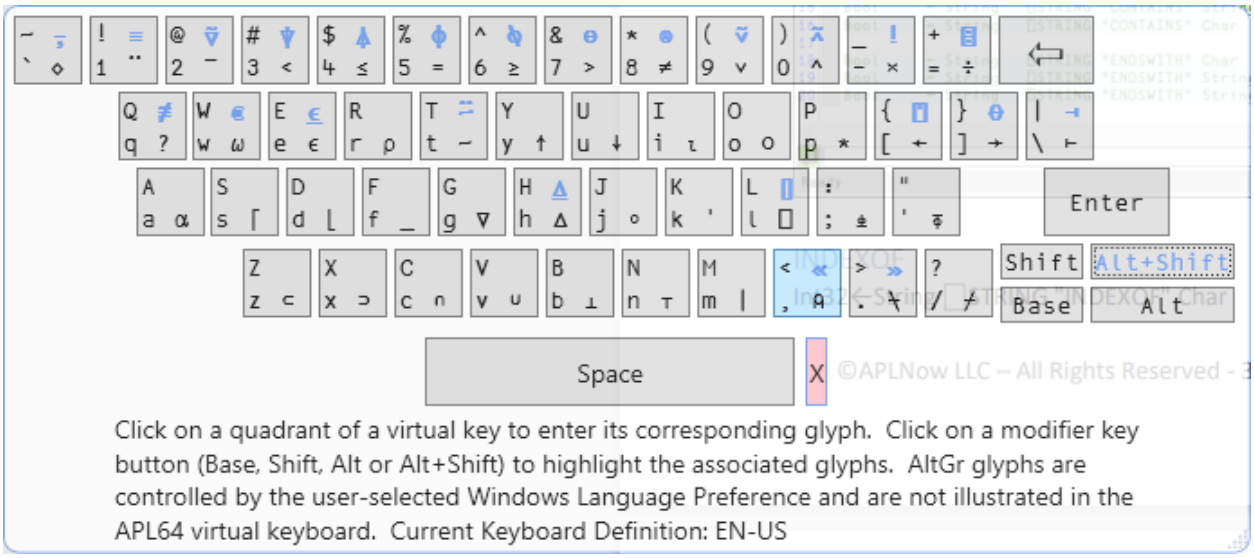
New APL String Data Type Supported

String Definition

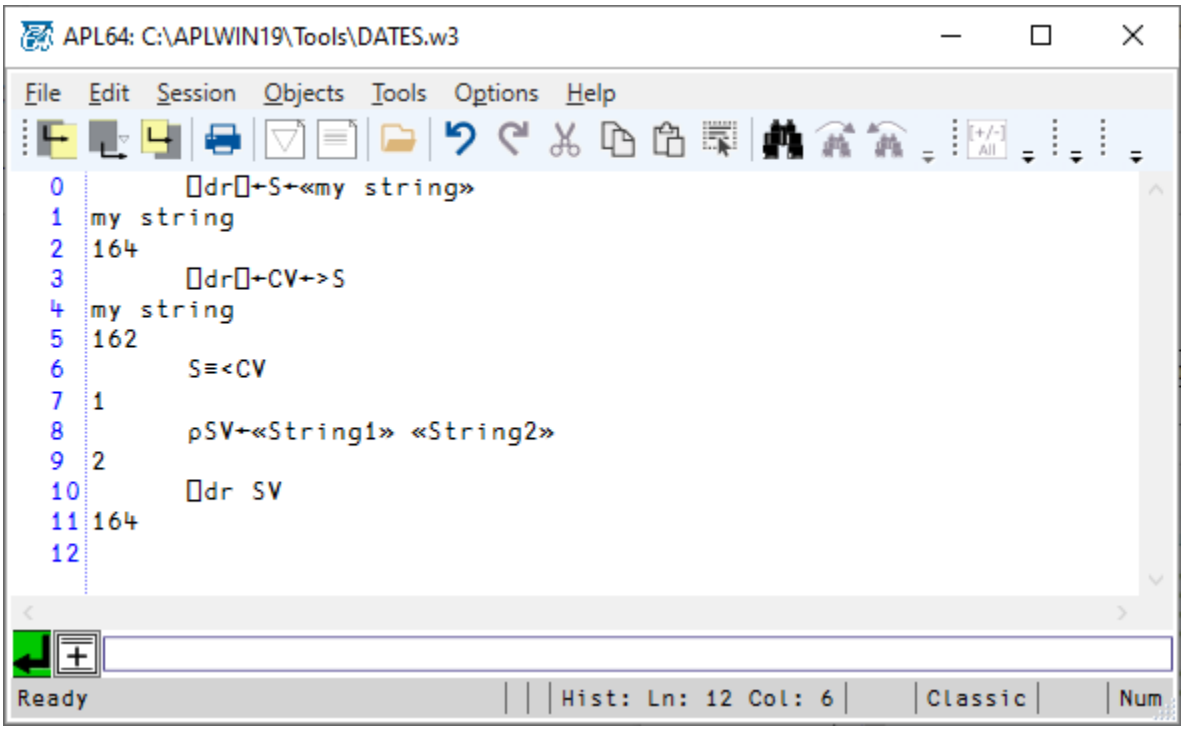
An APL string is an ordered group of characters treated as a scalar, with delimiters «…».

Defining a String Variable in APL64

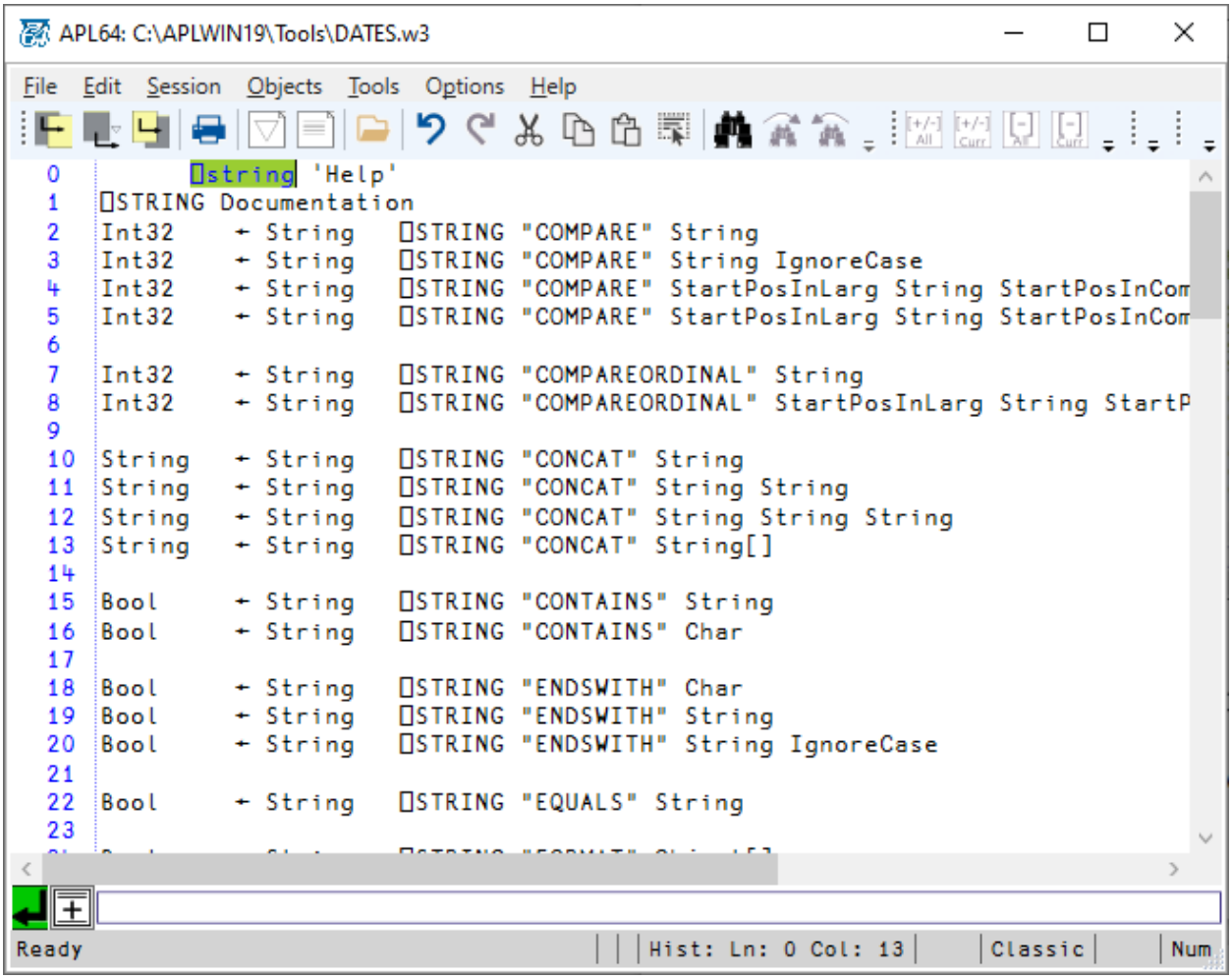
The ‘chevron’ string delimiters are available on the APL64 physical and virtual keyboards as Shift + Alt + < for « and Shift + Alt + > for ».



To define a string, enclose the text within the « and » glyphs. The APL64 enstring and destring functions are available for conversion between the APL character vector and string formats. APL64 supports string scalars and arrays.

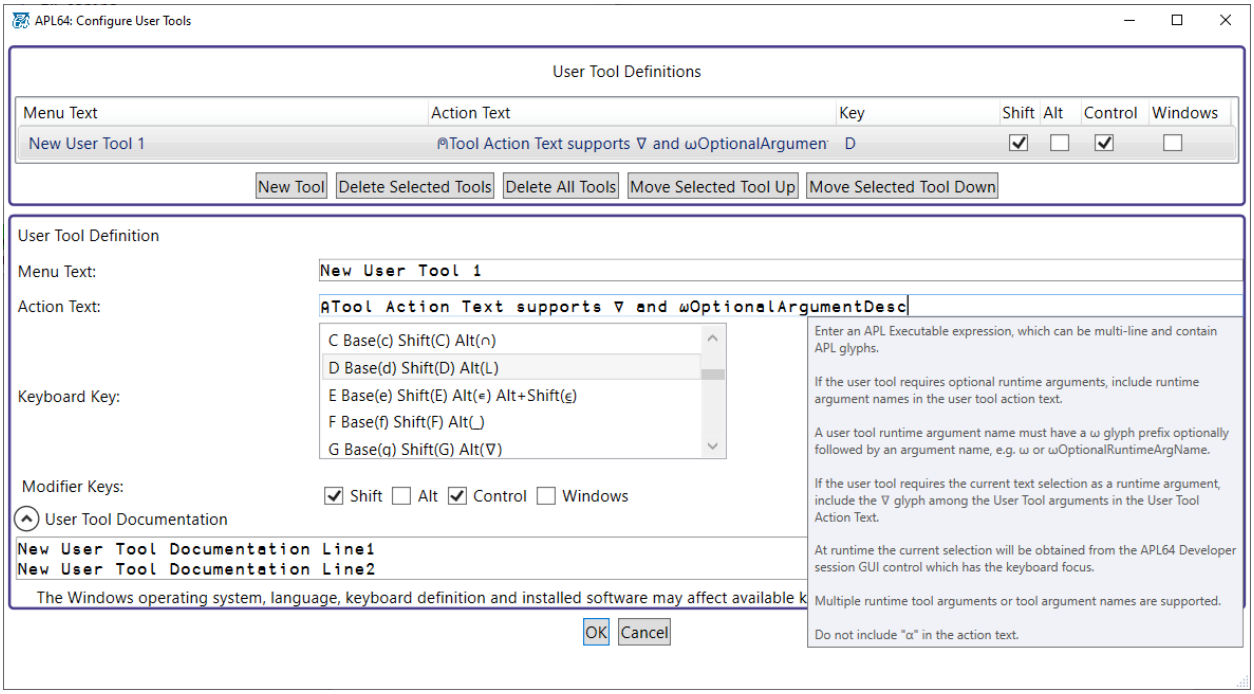


The ⎕STRING system function is an interface between the APL64 string datatype and the extensive features of the .Net String class.

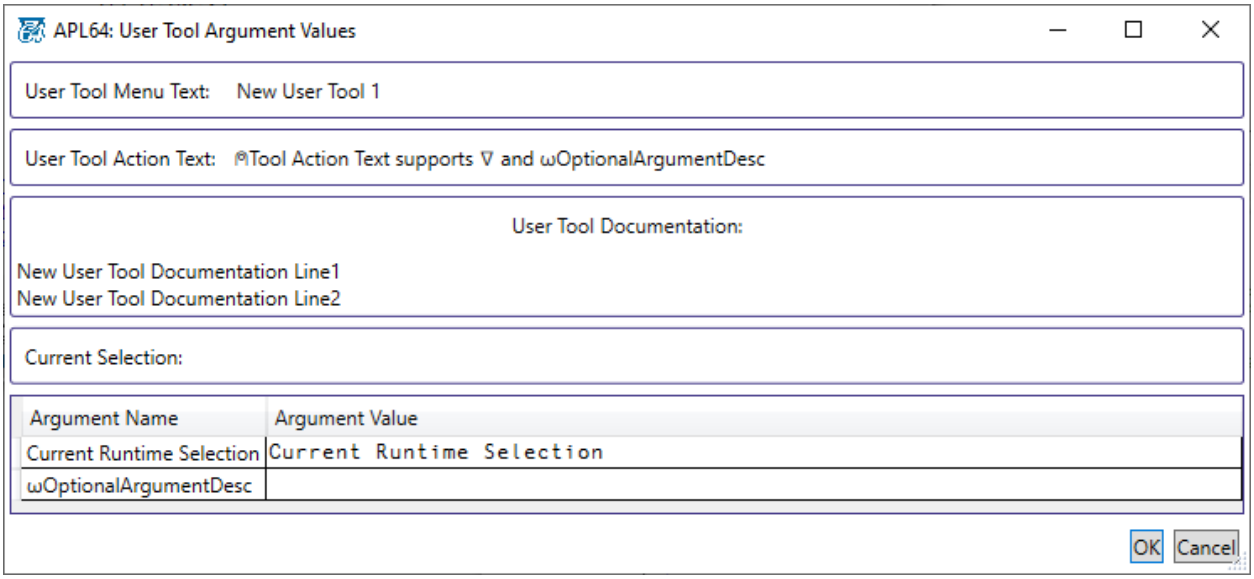


New User-defined Tools Features

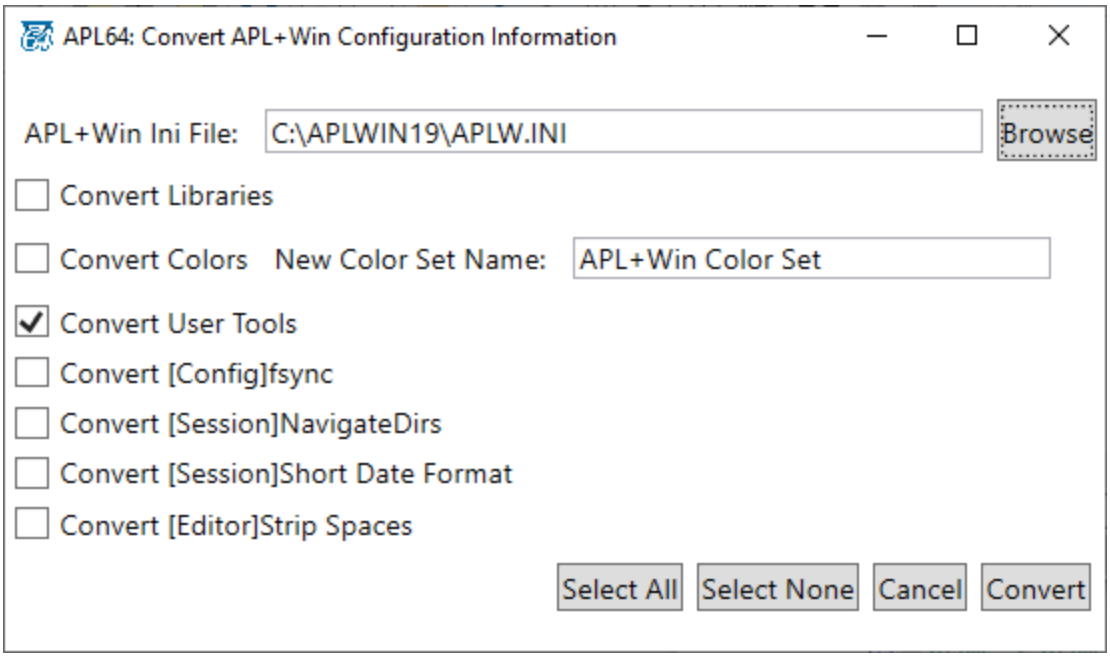
The Options > Configure User Tools dialog is enhanced including a user tool documentation multi-line field and optional user tool ‘ω’ arguments with suffixed user tool argument descriptions.



When a user-defined tool is used, the dialog to obtain the tool arguments is enhanced:

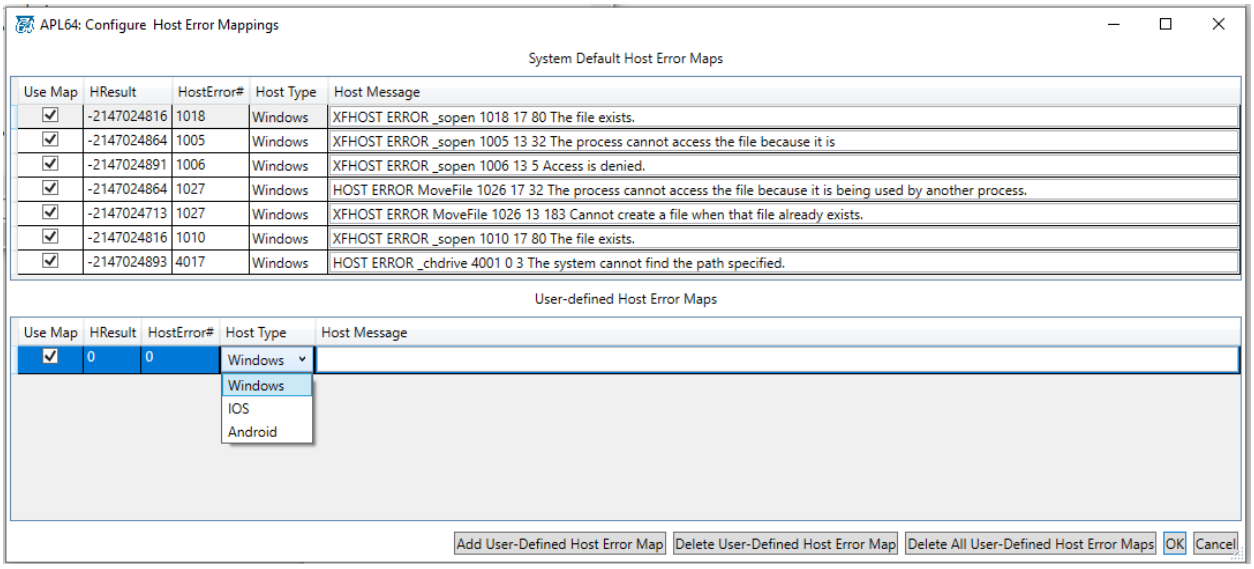


APL+Win user-defined tools may be converted to APL64 user-defined tools via the Options > APL+Win Configuration Conversion utility.



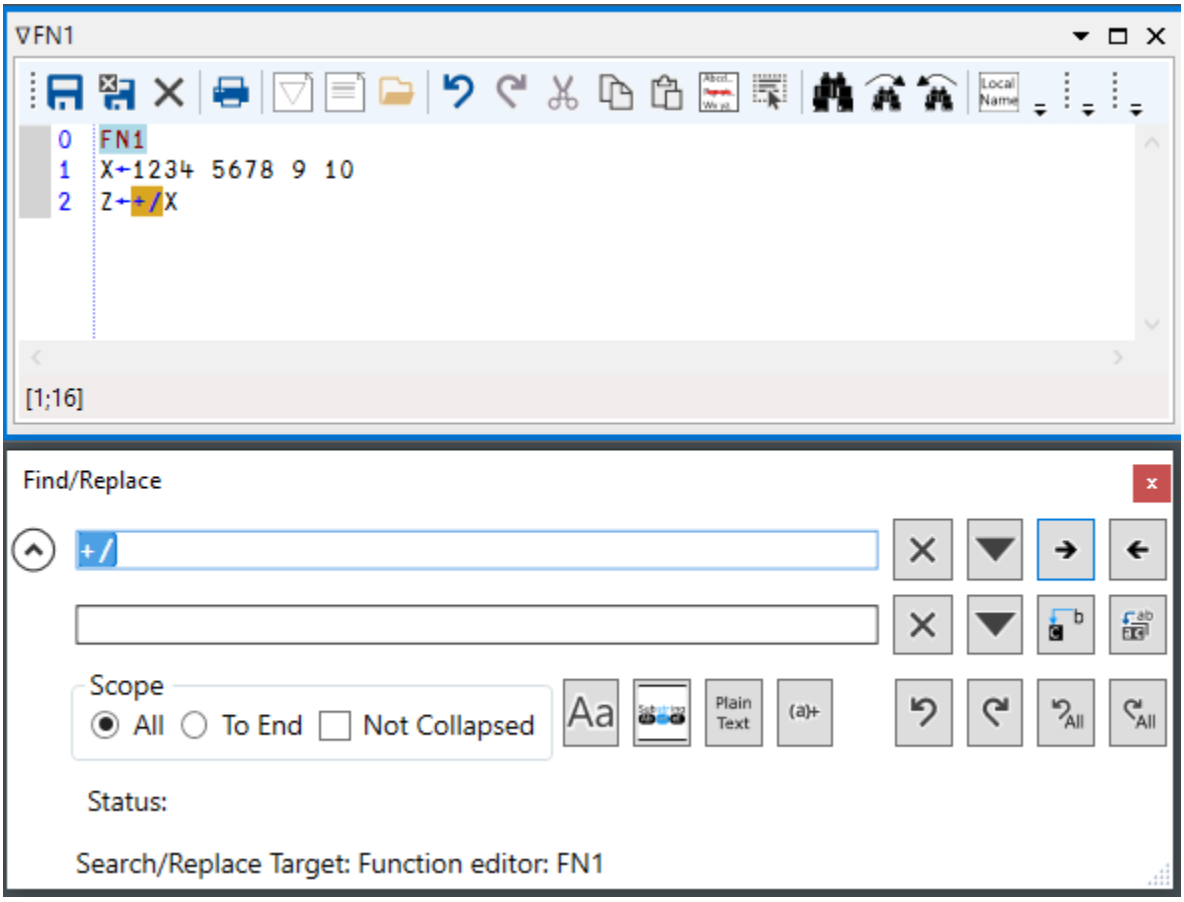
New Configure Host Error Mappings Dialog

The text of file system errors is not consistent between Win32 and .Net. The APL64 programmer may use the 'Host Error Mappings' dialog to coordinate exception handling in an APL+Win-based application with an APL64-based application system using the *Options > Configure Host Error Mappngs…* dialog.



Enhanced Find/Replace Dialog

The APL64 Find/Replace dialog supports Undo, Redo, UndoAll, RedoAll, Not Collapsed and specification of the text to find in WildCard or Regex format.



⎕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.

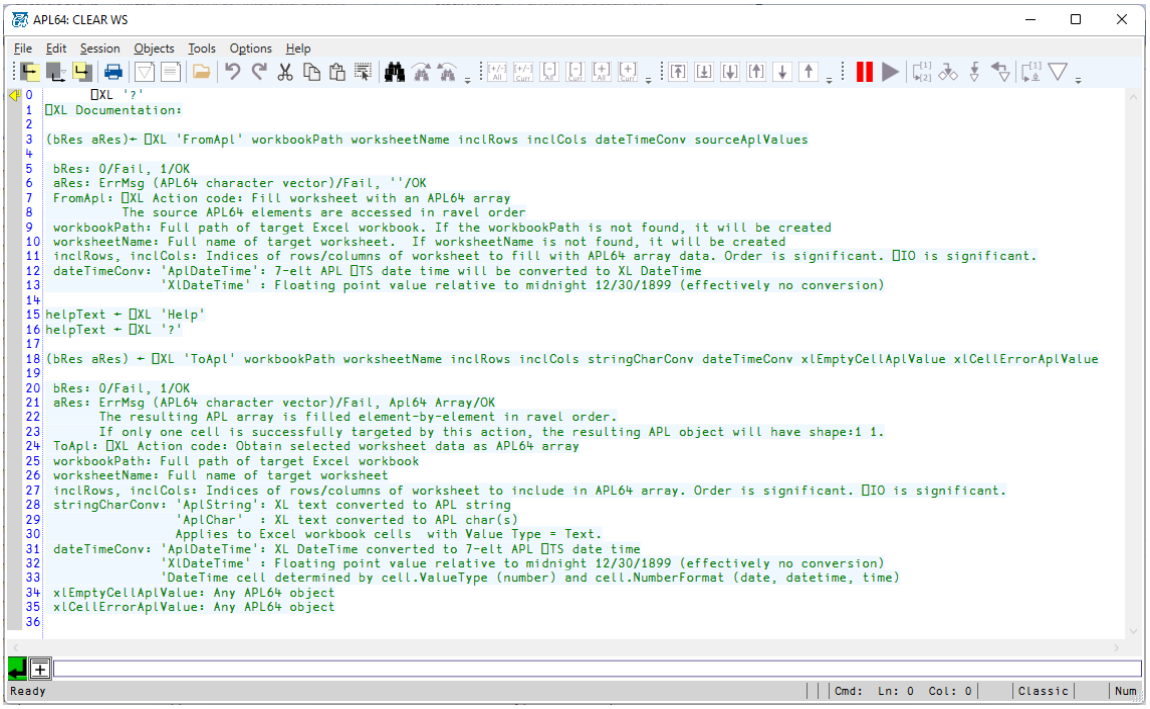
Syntax: Result ← ⎕XL Action [ActionArg1] …
Actions (not case-sensitive):

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'

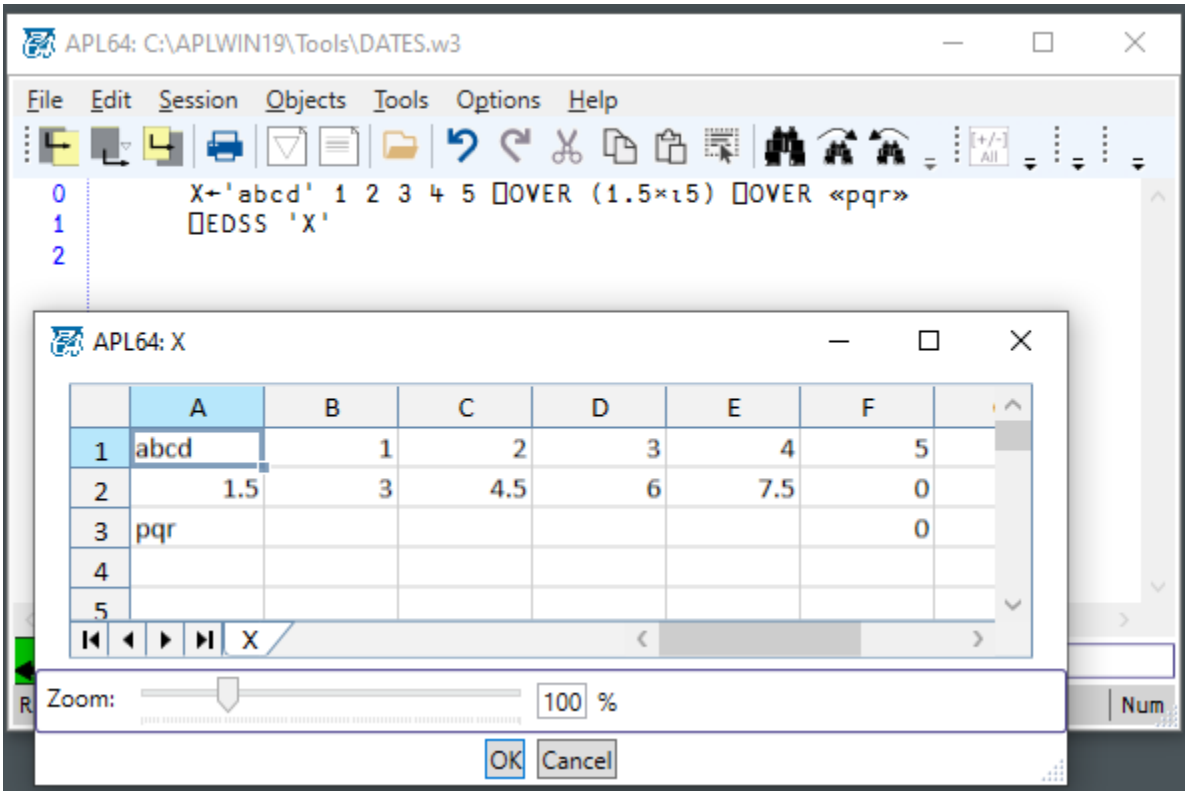
'?' or 'Help': Provides detailed syntax text in the form of an APL character vector:



⎕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.

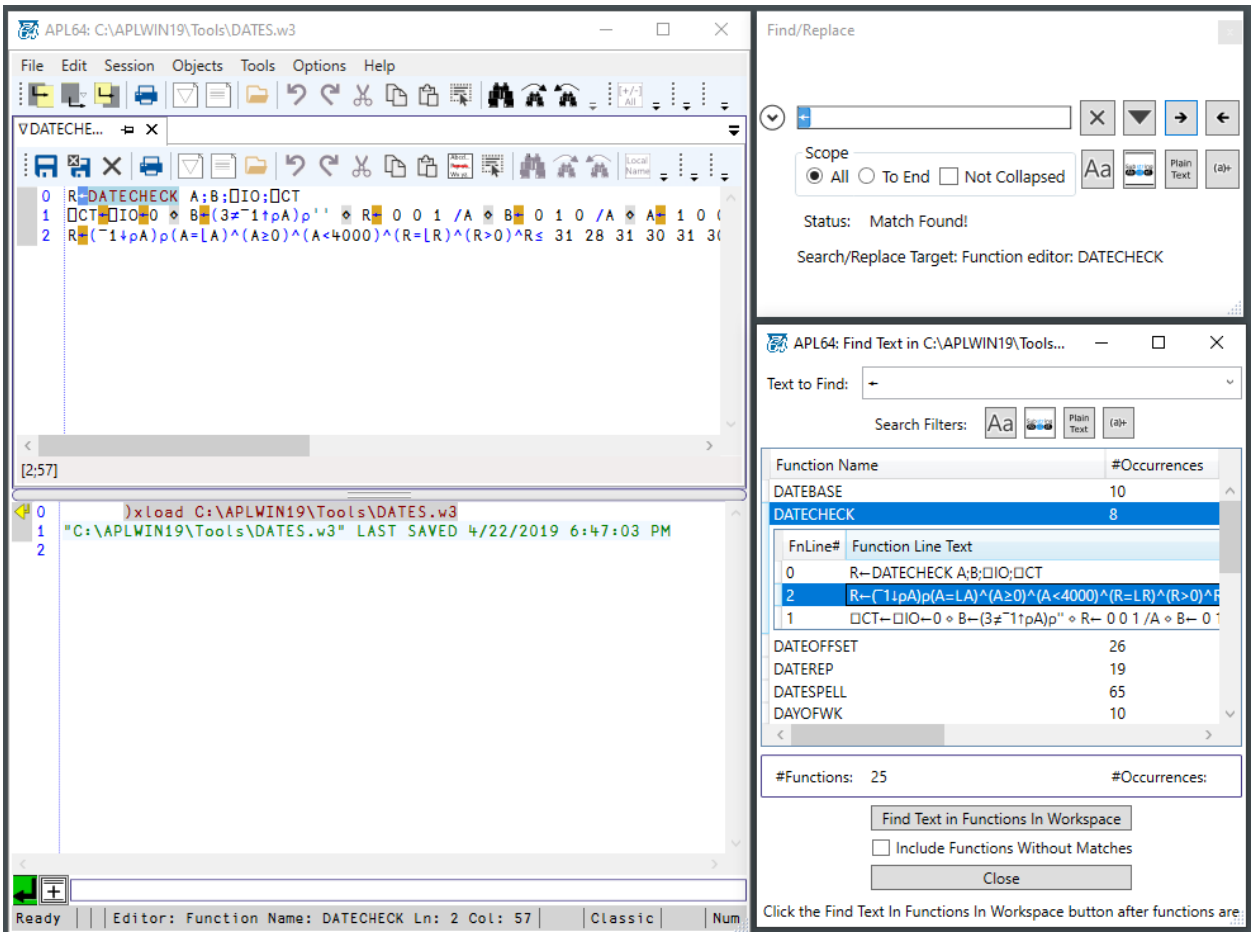


)EDSS Edit an APL Array in a Worksheet Editor

)EDSS is a new system command for editing an APL array of rank 2 or less in a graphical worksheet editor. This system command is analogous to the ⎕EDSS system function.

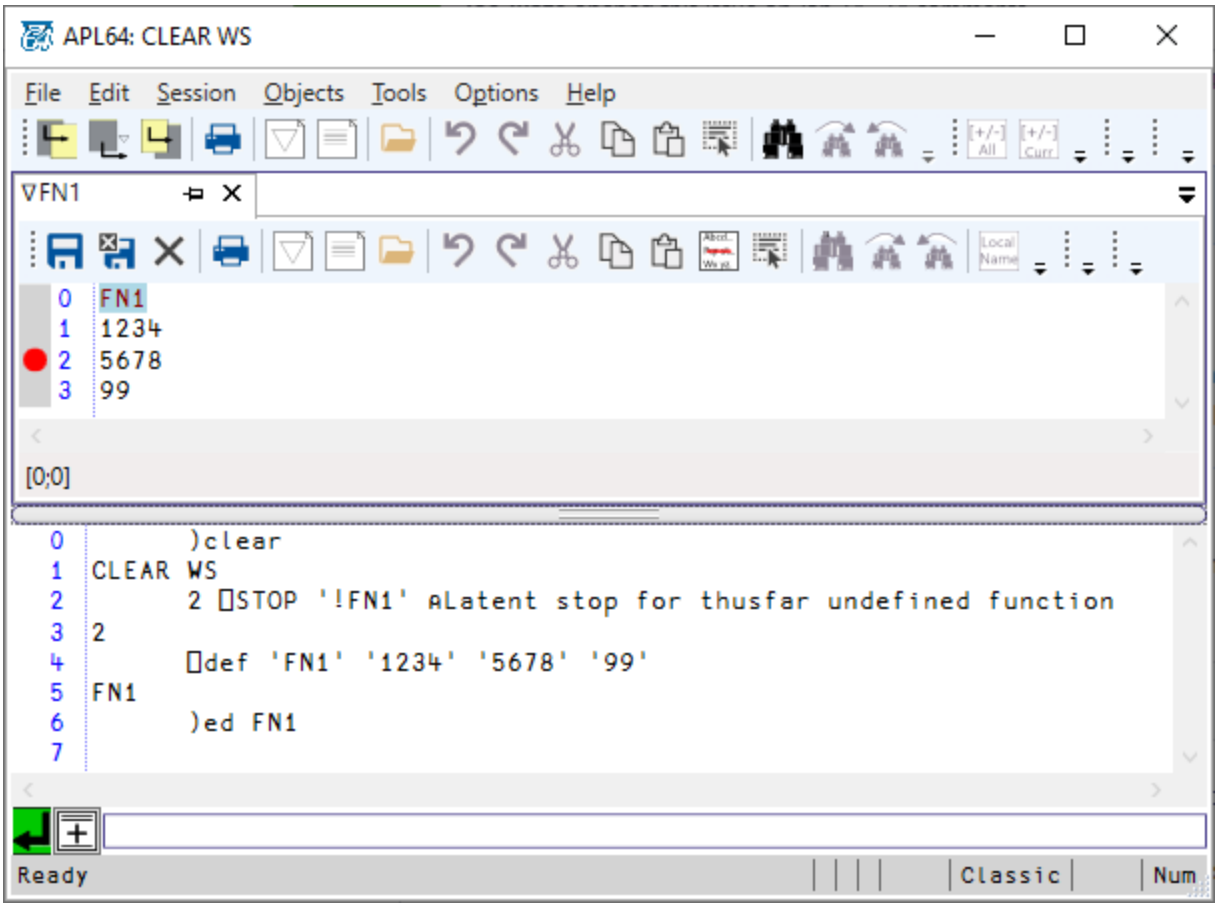
Find In Functions In Workspace Utility

The Edit > Find in Functions in Workspace utility will identify the function lines of the functions in the workspace which match the developer-selected text. A double left click on a user-specified row in the Find In Functions In Workspace dialog will present the selected function in a function editor.



Latent Function Stops

The ⎕STOP system function in APL64 supports latent stops which take effect when a function is defined in the current workspace. The enhanced syntax is: lines ⎕STOP '**!**fnName'.



Updates to Existing System Functions, Commands and Variables

Details of all APL64 System Functions can be found here.

)CMDA System Command and ☐CMD SystemFunction with Administrator Permissions

☐SYS[33] System Variable Enhanced Control of ☐INBUF Clearing Behavior

☐UCS Returns Unicode Points of Glyphs

☐UCASE, ☐LCASE

UCMD User Command Facility

☐LIBD System Function

Many New System Functions

Details of all APL64 System Functions can be found here.

☐A (char vector 'ABCDEFGHIJKLMNOPQRSTUVWZYZ')

☐ACBD (GET System.AppContext.BaseDirectory)

☐AL (char vector 'abcdefghijklmnopqrstuvwxyz')

☐**D (Char vector '0123456789')**

☐**DEVCAP**

☐**DIV**

☐**DRIVEINFO**

☐**DTB,** ☐**DLB,** ☐**DLTB,** ☐**DEB**

☐**DTBR**

☐**EDIT**

☐**EDITEVENTS**

☐**EXEPATH**

☐**FINDINFILES**

☐**LJUST,** ☐**RJUST**

☐**MATRIFY**

☐**NAN**

☐**NBLENGTH**

☐**NEXTO**

☐**OS**

☐**OVER,** ☐**OVERV**

☐**PATH**

☐**ROWFIND**

☐**SSCAT,** ☐**SSDEB,** ☐**SSDLB,** ☐**SSDLTB,** ☐**SSDTB**

☐**SCOM**

☐**SKD**

☐**SSFIND,** ☐**SSINDEX,** ☐**SSLEN,** ☐**SSSHAPE,** ☐**SSTAKE,** ☐**SSUNIQUE**

☐**SSTOMAT,** ☐**MATTOSS,** ☐**SSASSIGN,** ☐**SSDROP,** ☐**SSCOMPRESS**

☐**SPLASH**

☐**STRING**

☐**SYSCONFIGFILE**

☐**SYSVERN**

☐**TELPRINT**

☐TEXTREPL

☐TRANSLATE

☐UCASE, ☐LCASE

☐UCMDFILE

☐USERPATH

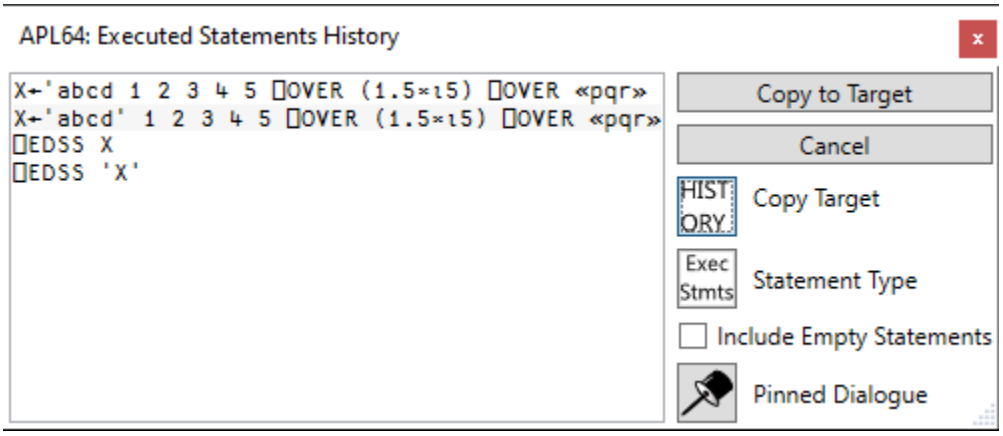
☐WHERE

☐WORDREPL

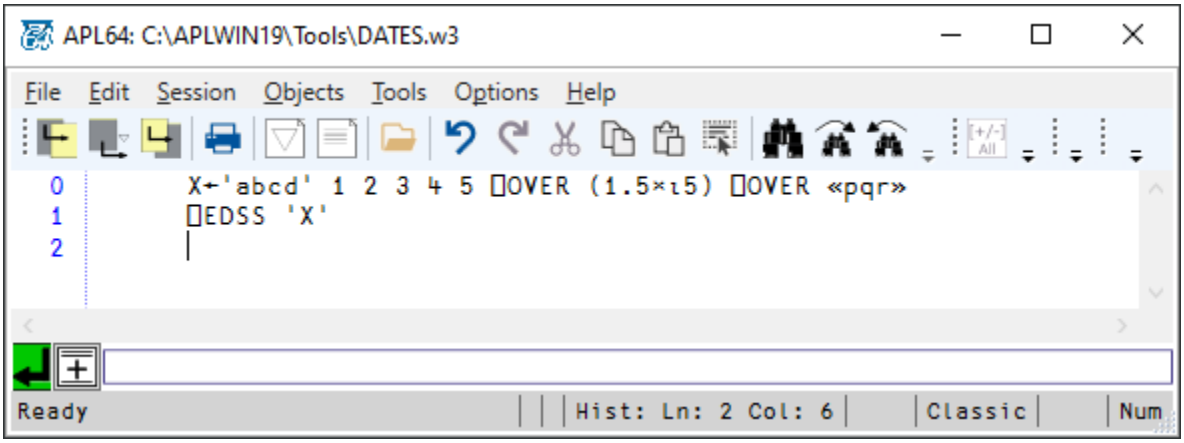
☐WRE

Enhanced Dialog

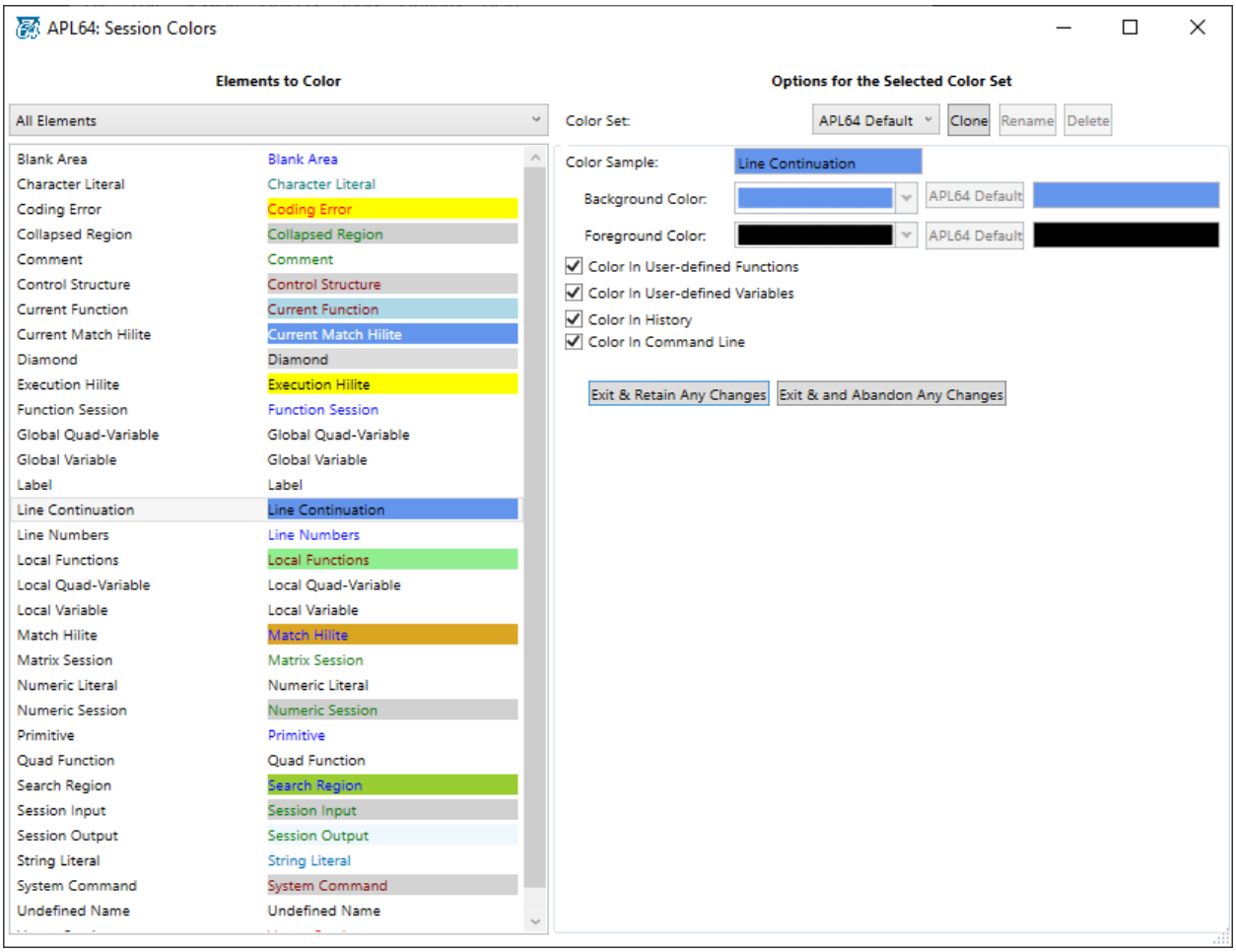
Gather Dialog



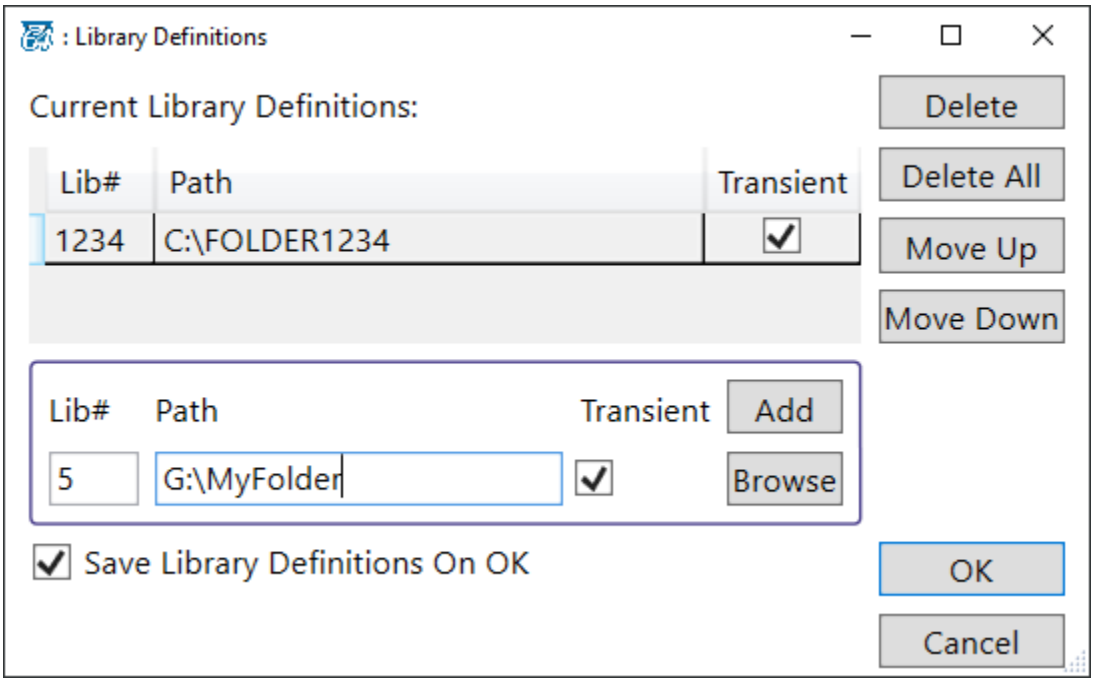
History Pane – Optional Row Numbers



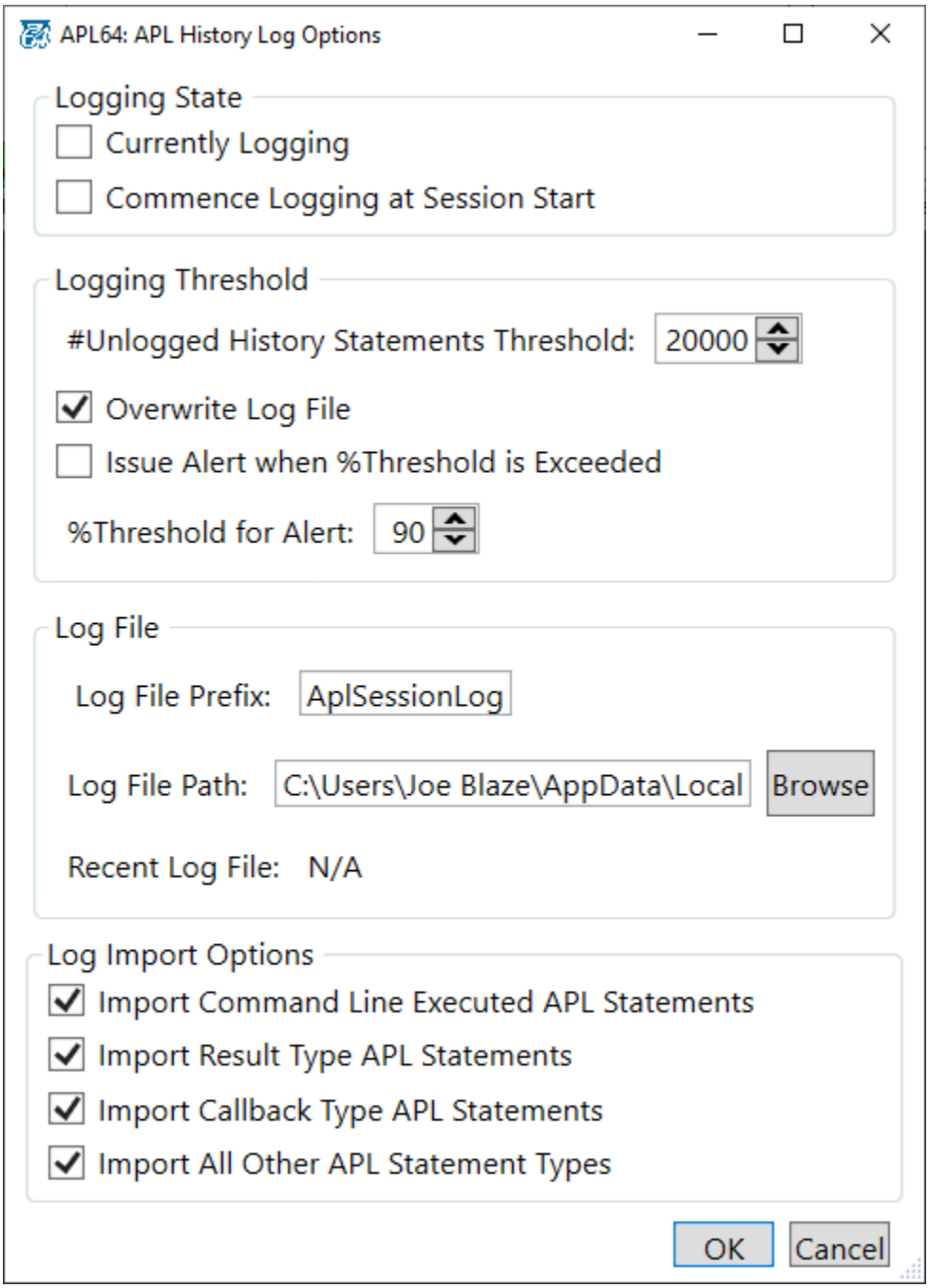
Colors Dialog



Library Definitions



History Log Enhanced



Keyboard Definition