APL64 2024.0.8 Update

Table of Contents

Overview

This document describes the enhancements and bug fixes that are planned for the APL64 2024.0.8 Update.

Interpreter Modifications

Arithmetic Functions Performance Improved for Small Arrays

The arithmetic functions (+ - × ÷) are improved for small arrays so that the performance of APL64 equals or may exceed that of APL+Win.

The APL64 interpreter identifies small arrays using the **⎕IT64 'TuneDS'** system function with a default threshold value of 100 elements. An APL64 programmer may query or modify this system function value to optimize system performance in an application system.

Since 2023 the analogous performance of APL64 for large arrays exceeds that of APL+Win. Since results may vary, application-specific testing is important.

Syntax:

```
existingSetting ← ⎕IT64 'TuneDS'   ⍝ query dyadic scalar primitive tuning parameter
⎕IT64 'TuneDS' newSetting          ⍝ set dyadic scalar primitive tuning parameter
```

New ⎕WSE system function: APL64 workspace engine object

The ⎕WSE system function exposes the APL64 workspace engine object. ⎕WSE instances may be created which can be used as independent APL64 engines from a controlling APL64 instance. A ⎕WSE workspace engine instance is an in-process server of the APL64 developer or runtime version which created it. All ⎕WSE instances are contained with the ⎕WSE object.

The ⎕WSE system function is designed to be used from an APL64 developer or runtime version instance. The ⎕WSE system function may be used in an APL64 developer version instance, an APL64 Windows Runtime Executable (WRE) instance and an APL64 Cross-platform Component (CPC) instance.

The ⎕WSE system function is designed to be a cross-platform feature of APL64. When an APL64 Cross-platform Component (CPC) is used in a non-Windows environment, the Windows-only features of APL64 cannot be used in a ⎕WSE instance.

Refer to **Help | APL Language | Using □WSE** for additional information.

New □AES system function: Encryption and Decryption

The □AES system function supports the encryption and decryption of text using the AES cryptographic algorithm, implemented in the Microsoft System.Security.Cryptography toolkit.

□AES supports the encryption of text which includes Unicode code points not in □AV.

Refer to □AES in **Help | APL Language | System Function** for additional information.

New □BROTLI system function: A better alternative to gzip inflate/deflate

Brotli is a lossless data compression algorithm developed by Google. It uses a combination of the general-purpose LZ77 lossless compression algorithm, Huffman coding and 2nd-order context modelling. Brotli is primarily used by web servers and content delivery networks to compress HTTP content, making internet websites load faster. A successor to gzip, it is supported by all major web browsers and has become increasingly popular, as it provides better compression than gzip.

□BROTLI supports the compression of text which includes Unicode code points not in □AV.

Refer to □BROTLI in **Help | APL Language | System Function** for additional information.

New □XML methods: JsonToXml and XmlToJson methods

These methods are helpful when accessing web services which use json format in APL64.

Refer to □XML in **Help | APL Language | Uisng □XML** for additional information.

Performance Improved for Colossal Component Files

By using explicit layout structures, rather than BitConverter logic, the performance of colossal component files is improved. Previously created colossal component files remain compatible.

APL64 Developer Version GUI Modifications

APL64 session responded to the Insert/Overwrite keyboard status changed in other applications

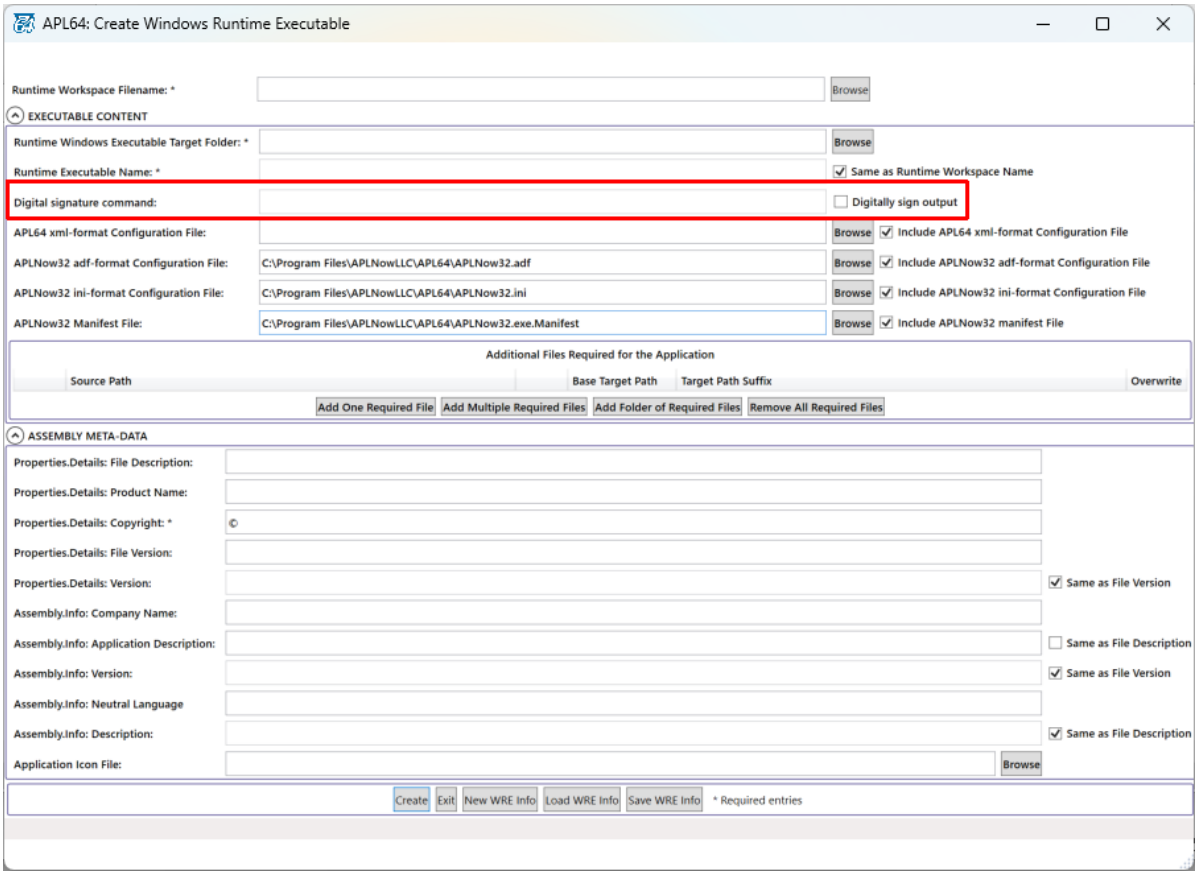
The CapsLock and NumLock keyboard status in the APL64 session could be out of sync with Windows

Bookmarks (ctrl+F2) did not work in a new editable classic history session

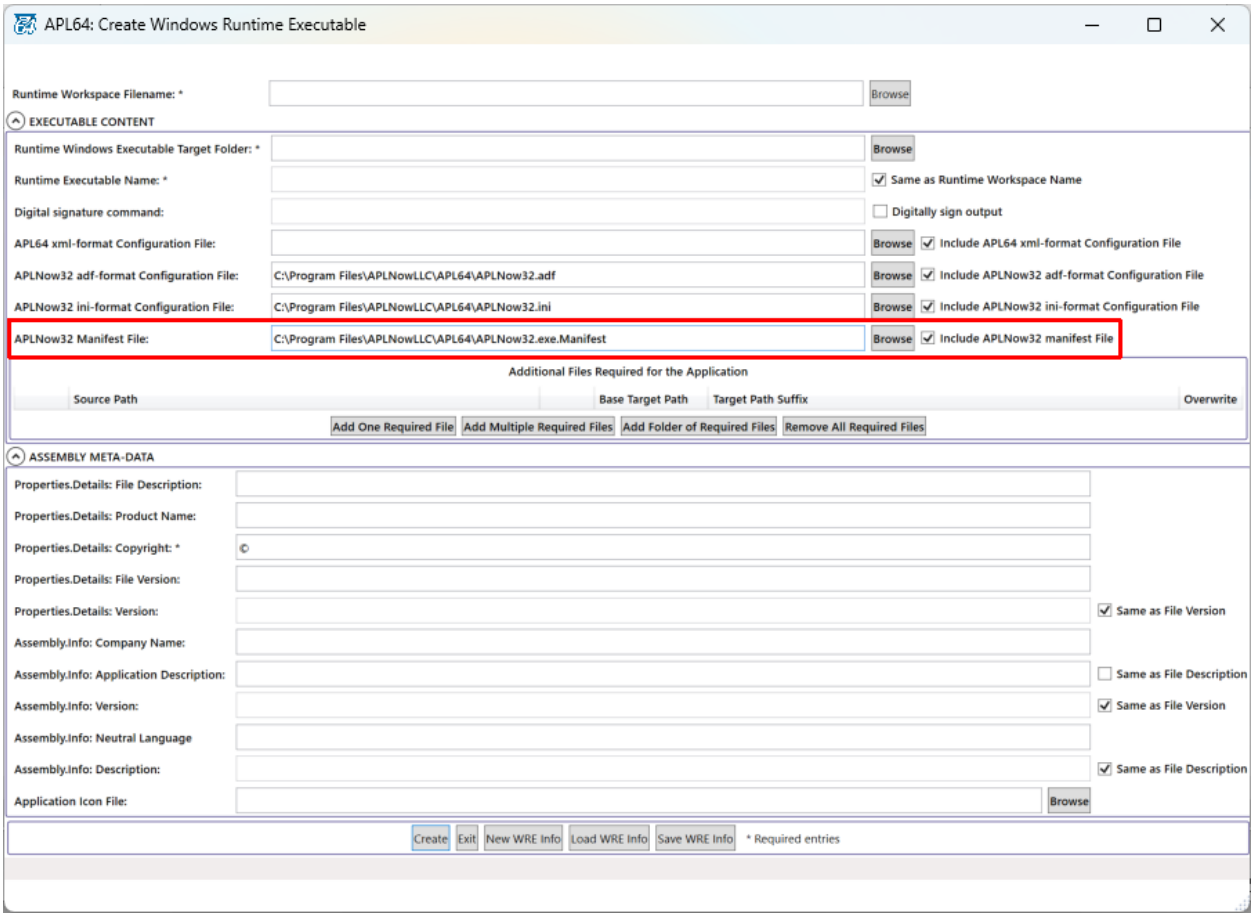
WRE and CPC Utility

Name validations for 'Runtime Executable Name', 'Company Name' and 'Copyright' fields

The Digital signature command and digital signature checkbox are on the same line in the utility



Entry added for the APLNow32.exe.manifest file*



*The APLNow32.exe.manifest file should be included in an APL64 WRE when some **wi** controls like CommandBars and CommandButtons are used.

Miscellaneous

Saving and Retrieving Unicode Data in APL64

Refer to new **Help | APL Language | NFE | Save Unicode Data** menu for information on handling application-specific text data that may contain Unicode code points which are not in AV.

Use an APL64 CPC in Excel

Refer to **Help | Developer Version GUI | Cross-platform Component | Cross-platform Component in Excel** for information on creating an APL64 cross-platform component (CPC) that can support application-specific Excel functions, which are transparently available for use in Excel worksheet cells or macros.

Bug Fixes

The outer product example, R←ΔB∘.*ΔP, caused APL64 to hang

Any ⎕WI commands did not execute in the history while a ⎕WI form was in a Wait method

Documentation corrections to the 'ToApl' action in the ⎕XL system function
Also described is that the ⎕XL 'ToApl' action returned Excel text data as APL64 string or Unicode character data instead of bytes (⎕DR type 82).

The ⎕EXEPATH system variable was relocated to the System Variable manual

CommandBars and CommandButtons ⎕wi controls did not display in WRE runtime

'inflate no_header wrapl=0' ⎕dr ... erroneously reported Zlib Error: Buffer Error

A FILE ACCESS ERROR may be reported when tying a classic component file with no File Access Matrix set
When APL64 reported a FILE ACCESS ERROR, the file wasn't closed, and other necessary cleanup steps weren't performed.

('inflate no_header wrapl=1') ⎕DR ⎕NREAD ... returned Zlib Error: More data than expected during inflate

⎕SA actions didn't work when set during ⎕LX execution

SCAN function (UTILITY.ws64) with left argument values 0 thru ¯3 didn't return function names

⎕CFRDCI sometimes failed with a floating point tie number