Dyalog Version 17.0

Dyalog Version 17.0 was released in July 2018 and is supported under Microsoft Windows, IBM AIX, Linux (including the Raspberry Pi) and Apple macOS. Version 17.0 is faster than any previous release of Dyalog and contains many new additions aimed at increasing developer productivity in a modern computing environment. This document introduces the highlights of the release.

Performance

As usual, we have done significant performance-related work in this release. Many primitive functions and operators have been enhanced to take advantage of vector instructions on x64, POWER and ARM hardware. 128-bit Decimal Floating-Point operations are a factor of 2 or more faster for most operations. Many algorithms involving Booleans and small range integers, or where one argument is a scalar, have been revisited – often with spectacular results.

Multi-Platform/Cloud Computing

Virtually all new features are designed to work on all supported platforms. Our goal is to make it practical to distribute the development, testing and deployment of an application across multiple platforms. It is becoming common to develop an application under Microsoft Windows or Apple macOS and deploy it on a cloud-based Linux platform – or run a front-end using Dyalog for Windows, with services running under Linux on a compute cluster or in the cloud.

Version 17.0 includes sample code for launching APL processes securely on remote machines using secure shells (APLSSH), and examples of how to do parallel computing in the cloud using futures and isolates. The set of cross-platform system functions for working on files and folders has been enhanced in version 17.0, with new functions `⎕NCOPY` and `⎕NMOVE`, and the extension of many existing functions to work on multiple files in a single operation.

Language Enhancements

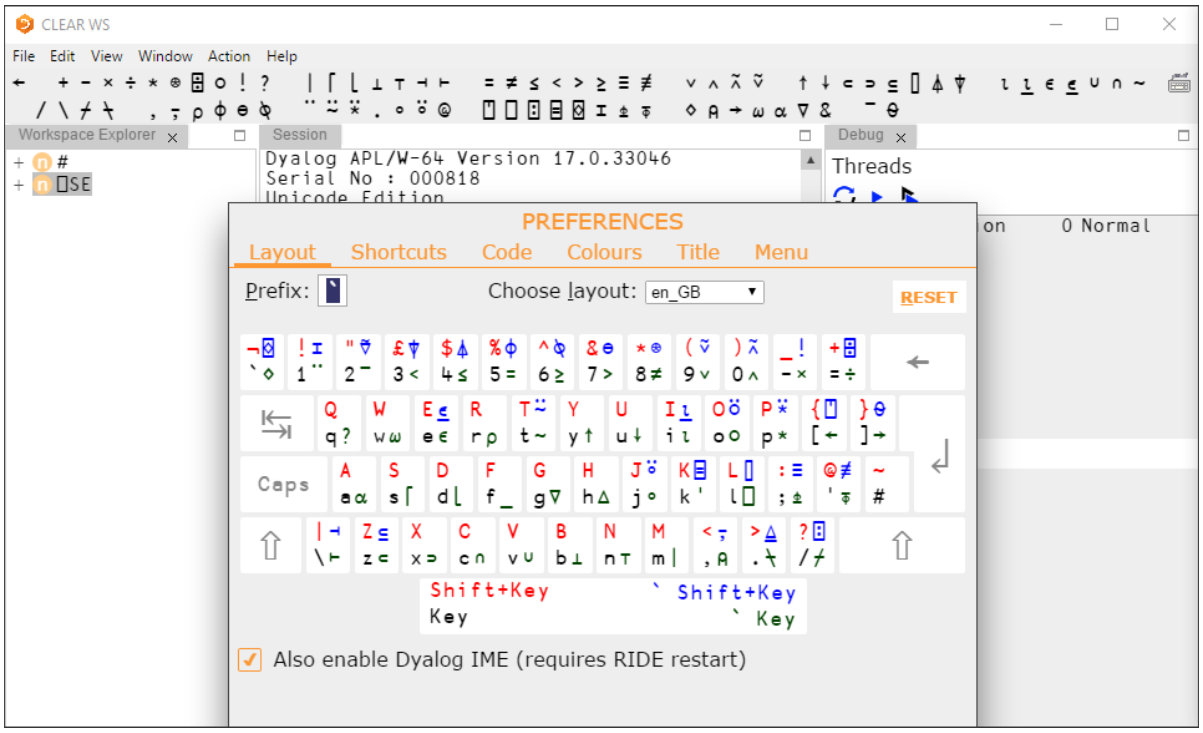
Most enhancements to Version 17.0 have targeted interfaces and tools surrounding APL; there is only a small number of enhancements to the core APL language.

The most significant enhancement is known as Total Array Ordering: the extension of the primitive functions up- and downgrade (`⍋⍒`) and interval index (`⍸`) to nested arrays of any depth. In earlier releases, the arguments had to be simple homogenous arrays containing only numbers or characters.

The functional coding style known as dfns requires that every statement in a function return a result. In version 17.0, the handful of system functions that did not return a result have been enhanced to return shy results.

Finally, a small bit of polish worth mentioning – when an APL expression fails, version 17.0 is able to pinpoint the precise position within the failing expression of the function that issued an error (the "caret" points to the failing function).

Remote/Portable Development Environment



```
Dyalog APL/W-64 Version 17.0.33046
Serial No : 000818
Unicode Edition
```

Prefix: `

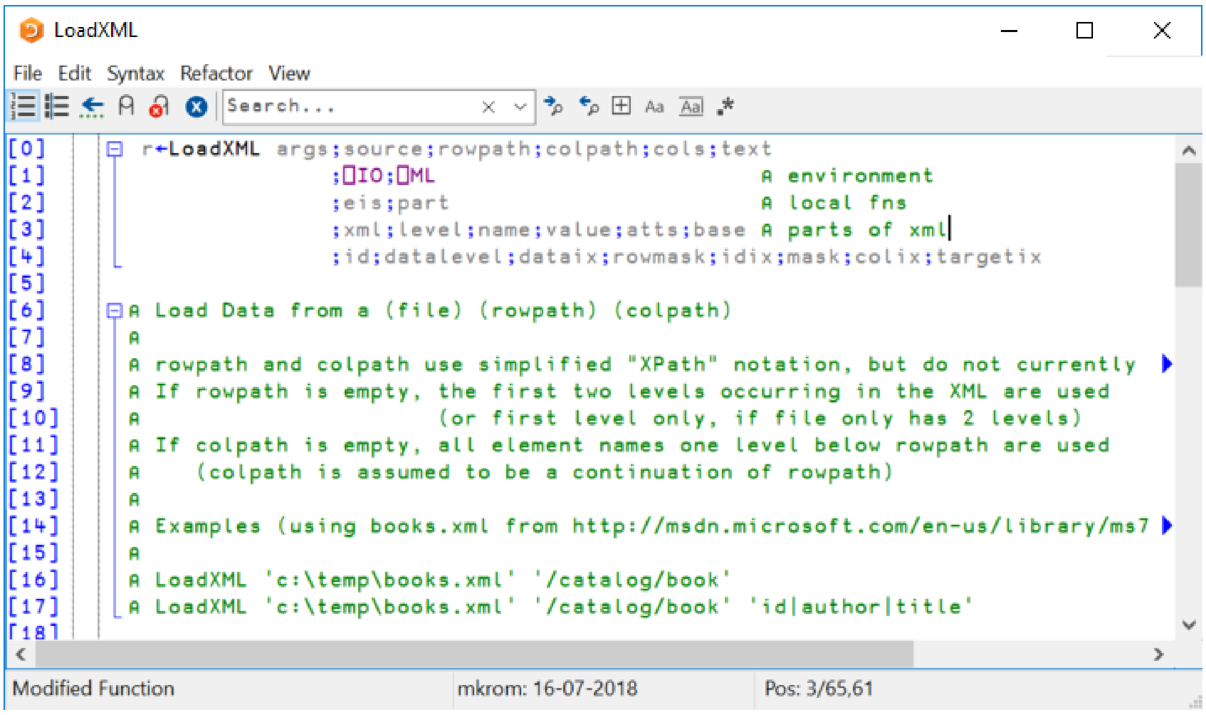
Choose layout: en_GB

[✓] Also enable Dyalog IME (requires RIDE restart)

A significantly enhanced version of the RIDE, our cross-platform development environment, has been released with Dyalog version 17.0. In addition to a crisp new look, enhanced performance when editing large functions and scripts, and many new editor features, RIDE 4.1 reinstates the floating editor windows that were temporarily lost in the migration to a new, more portable platform for the RIDE itself (Elektron).

Dyalog version 17.0 also supports using the RIDE without first installing it on each client computer; if the RIDE is installed on the machine where APL runs, then the interpreter can act as a web server and make the RIDE available using any compatible web browser. The "Zero Footprint" RIDE (so named because no client-side installation is required) has been tested with the latest versions of Google Chrome, Mozilla FireFox and Microsoft Edge.

Source Code Management



```
 r←LoadXML args;source;rowpath;colpath;cols;text
              ;⎕IO;⎕ML                         ⍝ environment
              ;eis;part                        ⍝ local fns
              ;xml;level;name;value;atts;base ⍝ parts of xml
              ;id;datalevel;dataix;rowmask;idix;mask;colix;targetix

⍝ Load Data from a (file) (rowpath) (colpath)
⍝
⍝ rowpath and colpath use simplified "XPath" notation, but do not currently
⍝ If rowpath is empty, the first two levels occurring in the XML are used
⍝                       (or first level only, if file only has 2 levels)
⍝ If colpath is empty, all element names one level below rowpath are used
⍝    (colpath is assumed to be a continuation of rowpath)
⍝
⍝ Examples (using books.xml from http://msdn.microsoft.com/en-us/library/ms7
⍝
⍝ LoadXML 'c:\temp\books.xml' '/catalog/book'
⍝ LoadXML 'c:\temp\books.xml' '/catalog/book' 'id|author|title'
```

Saved APL workspaces, which are single files containing a snapshot of code and data which can re-loaded in a single operation, are a convenient mechanism that remains popular with personal and casual users of APL. Professional users are increasingly moving towards using Unicode text files for source code. Support for source in text files is significantly enhanced in version 17.0, with the addition of tools that will monitor source folders for changes and automatically respond to actions performed in external source code management systems.

As an example of an enhancement that has been implemented to better support source in text files and external source code management systems, the declaration of local variables can now be spread out over multiple lines of code, to reduce the likelihood that changes made by multiple developers working on independent enhancements to a function will cause a "merge conflict" on the function header line when using source code management tools to auto-merge changes.

Integration

Under Microsoft Windows, Dyalog has a long history of providing mechanisms for tight integration with components written in other languages. The first integration technology was Dynamic Data Exchange (DDE) in the 1990s followed by COM/OLE/ActiveX and, more recently, Microsoft.NET assemblies implemented in Dyalog. Version 17.0 adds a cross-platform mechanism for wrapping APL code as regular shared objects or dynamic link libraries that can be called using any standard foreign function interface on just about any platform. The mechanism produces **.so** files under UNIX/Linux, **.dylib** files under Apple macOS, and **.dll** files under Microsoft Windows.

For integration of components that can be more loosely integrated, or are running on another machine or platform, Dyalog has provided a framework for SOAP-based web services known as SAWS (Stand-Alone Web Service framework). SAWS is now complemented by JSONServer, a framework for RESTful services based on the JSON protocol – an extremely simple mechanism for quickly providing selected functions from an APL application as a service that can easily be called from any language, running anywhere.

The shared library mechanism also supports the use of JSON as a serialization format for arguments and results, which means that the same APL code can easily be exposed as a shared library for use within a single machine, or as a web-based service if that is more appropriate, for example if the service is running in the cloud.