APL+Win Version 79.6.06

Sept. 2, 2010

This document contains important information pertaining to APL+Win Version 10.0 (v79.6.06) Beta Release. This beta will run for 30 days after it is first run.

Installation:

Place the executables, aplw_v10Beta.exe and aplwr_v10Beta.exe, in the same folder with your current APL+Win system and make a copy of your aplw.ini and rename it to aplw_v10Beta.ini.

Run aplw_v10Beta.exe to start the program.

Note: The APL+Win v10 Beta runs in Windows XP, Windows Vista and Windows 7.

Documentation:

APL+Win v10.0 System Features.PDF - Overview of APL+Win v10.0 Features

Crash Recovery in APL+Win v10.0.PDF - Overview of the crash recovery mechanism

TraceLog Execution History in APL+Win v10.0.PDF - Overview of the TraceLog execution history feature

Crash Log File Encrypt-Decrypt in APL+Win v10.0.PDF - Overview of the crash log file encryption and decryption feature

APL+Win v79.6.06

********** Important **********

Attached are documents that will give you a preview of what's in release 10.0 in more detail than was presented at this year's APL+Win User Conference. Many of the things presented then were not yet finalized, in some cases they have changed substantially based on experience using them and/or implementation realities we ran into when finishing them. In addition to any general feedback you might have regarding new features, we're

particularly interested in having you focus your attention on the Inline Control Sequence (ICS) feature that's described in the document, APL+Win v10.0 System Features.PDF. We've received mostly positive feedback regarding this feature from a relatively small number of people we've discussed it with, but a small amount of intensively negative feedback as well. Because of this early mixed responses, we think it's necessary to gain additional feedback from more of our customers on the ICS feature. As such, the decision was made to make ICS an experimental feature in release 10.0 that may be missing or different in a subsequent release. There will be an explicit enabling parameter that must be included in the INI file to make it available just in the development system interpreter (APLW.EXE).

We don't have any doubt that the capability provided by ICS is sorely missing from APL. Normally, APL is a concise language. A programmer can usually express in one APL statement what would take several statements to express in C#/C++ or many other languages. However, when it comes to logical cascades like those supported by ICS expressions, the situation is exactly the opposite, and C#/C++ code win hands down. One may have to use several lines (or several awkwardly coded and hard to read statements separated by diamonds on one line) to express not-that-complicated of logical decisions in APL that can easily be expressed concisely and clearly in one simple C#/C++ expression.

However, we are not sure that's how the APL community at large is going to feel. If the reaction is overly negative, then ICS might not be the right way to provide that missing feature. Please let us know what you think of the ICS feature, in general and whether you think the whole feature set should be included in APL+Win, or if only the :AND and :OR keywords should be allowed in the context of conditional expression for :IF, :ELSEIF, :WHILE, etc., kinds of control statements. The more information you provide us, the more informed our decision will be.

Please send your comments to support@apl2000.com.

What's Fixed

Bug Fix: Code Walker allowed functions that localize □SA to be stepped into with the debugger, without respecting □SA settings.

Bug Fix: APL+Win terminated when tying a share file with a classic

share file tie function (□ftie and □fstie) with a left argument with the following two conditions:

1. the drive doesn't exist
2. the drive in the path name is incorrectly formed; i.e., the back-slash (\) is missing after the colon in the target drive.

Bug Fix: Starting APL+Win with the 2G startup parameter ("aplw.exe 2G") started APL+Win with a workspace bigger than 2GB.

Bug Fix: The Frame that was a child of the Selector page visually appeared disabled in versions 9.1 and 9.2 although the enabled property was 1.

Bug Fix: Prior versions of the APL+Win v10 beta suspended during function execution. For example, execution stopped on return from

a ⎕WI call to a property of RichEdit control that causes a callback event to be executed.

Bug Fix: The autocomplete property for the Edit class did not appear in the state property, Make function and def variable.

Bug Fix: A DESTINATION ERROR occurred when the :LEAVE statement was
executed inside a :TRY statement with a :FINALLY clause that was nested inside a :FOR loop.

Bug Fix: The :CONTINUE statement did not enter the :FINALLY clause.

Bug Fix: APL+Win terminated when calling ⎕NI "Accept" twice on the same connnection instead of returning an error (code) due to the wAEWOULDBLOCK error.

APL+Win v79.6.03
---

1. New ⎕EM system function

The ⎕EM system function returns just the error message part of ⎕DM

(the first line up to but not including the first ⎕TCNL).

2. New :FINALLY clause in Try-Catch control structure

In prior versions of APL+Win, there was no built-in sure-fire mechanism to handle resource cleanup issues (such as closing files, database connections, ⎕WI objects, etc.) when an

execution context is exited due to an error or other means. The :FINALLY clause was added to address this deficiency. :FINALLY can appear at the end of a function, at the end of a :TRY block, and handles all forms of exit from above contexts including branching, :RETURN, and errors.

After the :FINALLY clause completes, execution resumes as it was originally intended (e.g., at branch target line, exiting from function or :TRY statement, or re-throwing of an ⎕ERROR).

Below are several examples:

```
      ⎕vr 'finally1'
    ∇ finally1
[1]   :try
[2]   ⎕←'hello'
[3]   →0
[4]   ⎕←'next'
[5]   :finally
[6]   ⎕←'finally'
[7]   :end
    ∇

      finally1
hello
finally

      ⎕vr 'finally2'
    ∇ finally2
[1]   :try
[2]   ⎕←'hello'
[3]   →L1
[4]   ⎕←'next'
[5]   :finally
[6]   ⎕←'finally'
[7]   :endtry
[8]
[9]   ⎕←'Not executed because of branch above'
[10]  L1: ⎕←'Reached L1'
    ∇

      finally2
hello
finally
Reached L1

      ⎕vr 'finally3'
    ∇ finally3
[1]   :try
[2]   ⎕←'hello'
```

```
[3]   →0
[4]   :catchall
[5]   ⎕←'not executed because no error occurs'
[6]   :finally
[7]   ⎕←'finally'
[8]   :endtry
    ∇

      finally3
hello
finally

      ⎕vr 'finally4'
    ∇ finally4
[1]   :try
[2]   ⎕←'hello'
[3]   .
[4]   :catchall
[5]   ⎕←'executed because SYNTAX ERROR on line 3'
[6]   :finally
[7]   ⎕←'finally'
[8]   :endtry
    ∇

      finally4
hello
executed because SYNTAX ERROR on line 3
finally
```

These functions are in the "test finally clause.w3" workspace provided with this beta.

What's Fixed

Bug Fix: APL+Win crashed when calling the columndisplay property for the Listview class.

Bug Fix: When copying or loading workspaces saved in v9 and older, the timestamps for functions were reset to 1980 1 1 0 0 0 0.

APL+Win v79.6.02

1. The limitation of not being able to load some workspaces saved

in an earlier versions of APL+Win has been addressed in this release.

2. New ⎕THROW system function

New ⎕THROW function is similar to ⎕ERROR except it signals error natively rather than popping calling function off ⎕SI stack before signaling an error.

For example:

```
    ∇ throw;⎕elx
[1]   ⎕elx←'⎕throw ⎕dm'
[2]   :try
[3]     ⎕throw'TEST ERROR'
[4]   :catchall
[5]     ⎕throw ⎕dm
[6]   :endtry
    ∇

      throw
TEST ERROR
throw[3]   ⎕throw'TEST ERROR'
           ^
throw[5]   ⎕throw ⎕dm
           ^
      throw
      ^
```

Note that in the code above the ⎕THROW inside the :TRY clause is caught by the :CATCHALL clause. Calling ⎕ERROR in this context would have immediately thrown the error out of the calling function. But instead we trap the error in :CATCHALL clause and call ⎕THROW ⎕DM there which causes the error to be again signalled natively, on line [5] but this time since the error is not in the :TRY context, the ⎕ELX handler runs. It also calls ⎕THROW ⎕DM but since it is called by ⎕ELX it is treated as if ⎕ERROR was called and causes the error to finally be signalled outside of the function context.

More examples are in the file "QuadThrow Examples.pdf" distributed with this release.

APL+Win v79.6.01

This release has three major enhancements. They are:

1. Support for Larger Workspace Size

The old maximum workspace size was approximately 1600 MB (1.4 GB). APL+Win v10 beta supports larger workspace sizes; up to 2.6 GB on 32-bit machines (with 4-GT enabled) with a 32-bit Windows operating system and up to 3.6 GB on 64-bit machines with a 64-bit Windows operating system.

There is a wssize command-line parameter that specifies the default workspace size to start APL+Win. Without it, the default startup workspace size remains 75% of available virtual memory.

E.g.,

- On Win32 this is typically 75% of 3 GB = 2.25 GB
- On Win64 this is typically 75% of 4 GB = 3.00 GB

Important: The wsSizeLo and wsSizeHi command-line options described at the 2010 User Conference haven't yet been implemented. They will appear in the next beta.

For information on 4-GT, go to http://msdn.microsoft.com/en-us/library/bb613473%28VS.85%29.aspx?ppud=4

2. Faster System Performance

Scalar and array operations will run 30 to 50% faster than in prior versions of APL+Win.

3. Larger Arrays

In addition to larger workspaces, the limits for arrray shapes has been increased. This has historically been defined so that the maximum number of elements in all array types cannot exceed the hard limit of 214748352. The new limit is 2147483647, which is approximately a 10-fold increase. Note that the total object size in bytes cannot exceed 2GB (2147483616 bytes).

==========

You are encouraged to try out the new enhancements in this release with and without your application(s). Your contribution is vital as it will ensure that we address any problems that are present in this release.

Being that this is a beta, you should not save any workspaces of yours that are deemed critical to you business without first having them backed up. We also ask that you not redistribute any part of this release to others especially your customers as it is intended for your use only.

If you experience any problems or have any questions or comments about this release, please report them quickly to APL2000 Technical Support at support@apl2000.com.

Thank you for testing the APL+Win v10 Beta!

APL2000 Technical Support
support@apl2000.com