APL+Win Version 79.6.03

June 10, 2010

This document contains important information pertaining to APL+Win Version 10 (v79.6.03) 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.

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
---

1. APL+Win crashed when calling the columndisplay property for the Listview class.

2. When copying or loading workspaces saved in v9 and older, the function timestamps 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