Debugging an Application System using the APLNext Supervisor

Halt/Step Debugging Tool

APL+Win application system programmers are familiar with the 'halt/step' method of debugging APL+Win application system program code. The method is ideal for isolating the area of application system code which needs modification or correction. This 'halt/step' technique continues to be available when an application system incorporates the APLNext Supervisor.

APLNext Supervisor Implementation Background

The APLNext Supervisor can be implemented by an application system programmer with all APL+Win code or a mixture of C# and APL+Win code. The 'Controlling Application' portion can be written in APL+Win or C# (or any other .Net language). The 'Kernel' portion of the application system is written in APL+Win. The purpose of the APLNext Supervisor is to facilitate the running of APL+Win 'Kernel' code in multi-threaded mode. The APLNext Supervisor documentation, installed along with the tool, describes this arrangement in greater detail.

Debugging the 'Controlling Application'

If the 'Controlling Application' is written in APL+Win, one can 'step through' this portion line-by-line using the APL+Win session containing the 'Controlling Application'. If the 'Controlling Application' is written in C#, one can 'step through' this portion line by line using Microsoft Visual Studio in debug mode. The required keystrokes are similar. The application system programmer will place a program stop in the appropriate region of the 'Controlling Application' source code.

Debugging the 'Kernel'

Since the 'Kernel' portion of the application system is written in APL+Win, the APL+Win session containing a running instance of the 'Kernel' workspace can be debugged within that APL+Win session. The 'Kernel' workspace functions are executed within an instance of APL+Win as an ActiveX server, so it is necessary to register the APL+Win 'developer' version on the machine used for debugging the 'Kernel' source code.

In order to conveniently debug the APL+Win 'Kernel' source code, the running instance of the 'Kernel' workspace must be made persistent and visible and an appropriate program stop must be placed within an application-programmer-selected function in the 'Kernel' workspace.

The running instance of the 'Kernel' workspace is made persistent and visible by using existing APLNext Supervisor XML-format configuration text options. Setting the value of the 'debug' tag '1' will prevent the 'timeout' tag value from applying to the execution of the 'Kernel' workspace instances. Setting the value of the 'visible' tag to '1' will make the instances of the 'Kernel' workspace visible. These values can be set when the 'Controlling Application' using the APLNext Supervisor 'OpenConfigFromFile' or 'OpenConfigFromText' methods. In production mode, these values should generally be reverted to '0'.

It may also be helpful during application system debugging of the ‘Kernel’ workspace to set the ‘minpool’ and ‘maxpool’ tag values to ‘1’.

When the application system is run in this debug mode, the instances of the APL+Win-based ‘Kernel’ workspace will be visible and they will persist until the processing they perform is complete or they are closed by the application system programmer. If the ‘Kernel’ workspace has been saved with an appropriate program stop or if the application programmer has added a line of APL+Win code which creates the appropriate program stop at run-time, the debugging operation can begin from that point on in the execution of the ‘Kernel’ workspace.