Lab Note #13: EPOXI run-aways

Recently, together with one of our customers, we encountered a problem in using epoXi and eXegete Client to replace VPLUS within an application. The problem's symptom resembled a system lockup or network hang. It developed when a user closed eXegete Client via the top-right hand 'X' or merely powered off his or her PC, rather than exiting by using the application's normal exit command. Looking closer, we found that this was causing the application program to become CPU bound, using up to 98% of the CPU time. We call this a run-away process.

So why did an application that has been working for years now fail when running with epoXi and eXegete Client? The answer is found among the differences between "host-terminal" and "client-server" computing.

When using terminals or terminal emulation software such as Reflection, Win92, Zebra, etc., the actual connection where data exchange occurs is with the MPE process VTSERVER. When a terminal disconnects, VTSERVER detects this condition and terminates all related processes and ends the session.

However, with epoXi and eXegete Client (as well as any other client-server structure), the data exchange occurs between these two processes through a network socket. When a disconnection from the socket occurs, the server's application software must handle the resulting error itself. When epoXi detects a disconnection or other socket error, it returns an error status to the application, which in turn must decide what to do. In the case study, we found that the cause was improper handling of the returned status from the procedures VSHOWFORM and VREADFIELDS in the application source code. The application was expecting but not receiving specific data in the VPLUS buffer to conclude processing. When the disconnection occurred, the status was not checked, leading the process to a tight loop of VSHOWFORM followed by VREADFIELDS, and causing the application program to dominate the system.

Although a future version of epoXi may address this issue by terminating the application, the current technique for solving this type of problem is to include appropriate error handling code after calls to system procedures. When using epoXi and eXegete Client to enhance and modernize your applications, please be sure to handle the following errors:

When a disconnect is detected in VSHOWFORM, EPOXI sets the VPLUS status to 130

```
Internal error: Terminal write failed. (FSERR 32)
```

When a disconnect is detected in VREADFIELDS, EPOXI sets the VPLUS status to 160

```
Internal error: Terminal read failed. (FSERR 32)
```

Lab Note #13: EPOXI run-aways, 05 July 2002 Keven Miller
Feedback and topic suggestions are welcome and can be emailed to mailto:technote@exegesys.com