Debugging NT's Registry

BY GUY C. YOST

When debugging complex and hidden NT system registry issues, you'll need the right tools to obtain a detailed level of information without being intrusive to the OS or application/driver in question. One source for such tools, www.sysinternals.com, recently helped me when I encountered a custom database application that used registry entries to direct the application's file operations and specify the IP address of the back-end SQL database.

My client had just changed the IP addressing scheme of their servers and subsequently had problems getting to the same server via the new IP address. At first, I thought the problem would be simple to fix: Simply update the application's configuration files or registry entries to point to the new IP address. However, when we brought up the client, the configuration settings didn't let us specify an address for the database server. We contacted the programmer in Michigan who candidly admitted that he was still working on the functionality of the configuration utility. He suggested that for the time being we could just update a registry entry using REGEDT32.

I found in the registry where the program specified the SQL server address and updated it with the new address as directed by the application's creator. When we started the application, however, the SQL server could not be found. We ping'd the server and checked primitive port operations of the SQL service by using a kind of telnet utility that allowed us to manually handshake a connection and make a few database calls. That was enough to point us back to the client for further investigation. I brought up REGEDT32 again and noticed that the IP address had reverted back to the old setting. I asked the programmer if the client read a configuration file and wrote the address setting to the registry while it was loading. He confidently indicated that the program simply looks to the registry for the setting and uses it — no configuration files or writing to the registry were involved.

Extremely perplexed, I downloaded the Registry Monitoring utility REGMON from www.sysinternals.com. This utility can capture and record all registry activity and has a convenient "always on top" feature as well as a filter that allows you to specify what type of registry activity you want to monitor. I set the filter to capture only write operations (see Figure 1) and ran the client after manually changing the IP address back to the new setting.

As I suspected, the client indeed wrote the original address to the registry that REGMON captured, as shown in Figure 2. Note that the highlighted operation was a "SetValue" request and the IP address is shown as the value.

Having proved that a persistent write operation was causing our problem didn't help our Michigan developer to provide an explanation.

Figure 1: Setting REGMON's Capture Filter

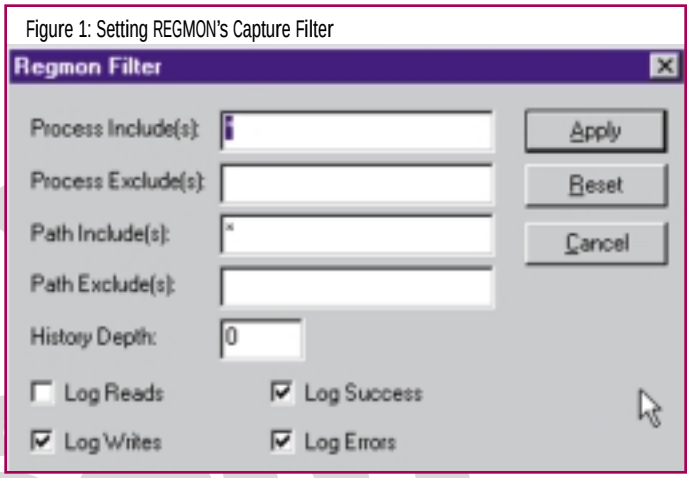


Figure 2: REGMON's Captured Data

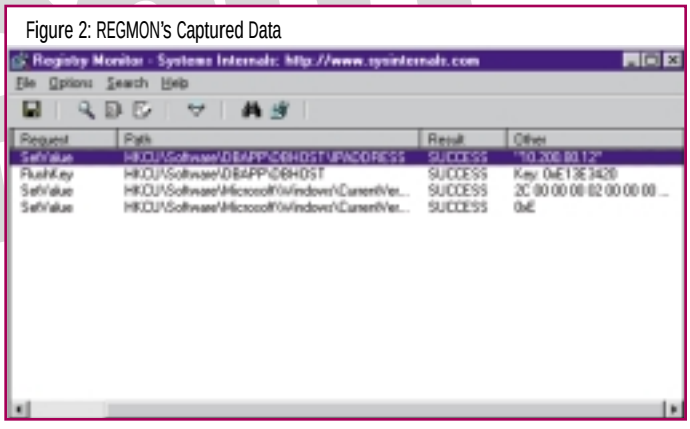
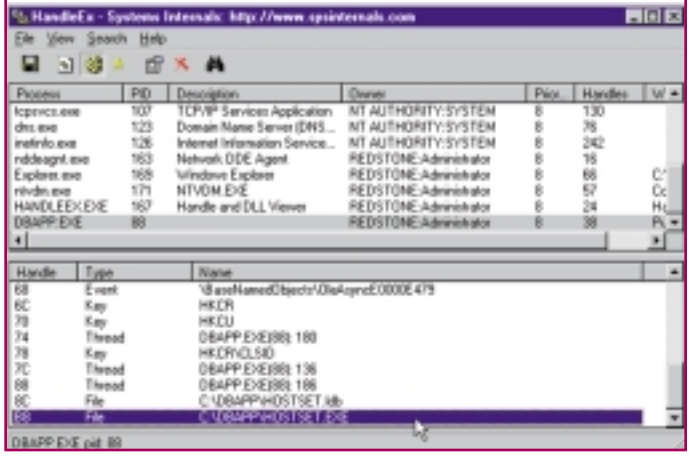


Figure 3: Using HandleEX to find Child Processes



He asked us to check if any batch files were calling REGEDT32 from the command line and writing the data that way; however, the application icon was associated directly with his executable rather than a batch file, which told us the operation was being invoked by his program.

To speed up our troubleshooting process, I went back to the Sysinternals Web site because I remembered seeing a description for another utility called HandleEx that read "Ever wonder which program has a particular file or directory open? Now you can find out."

HandleEx is a great little program that shows you information about which file handles and DLLs have opened or loaded as a result of running a program or process. Notice in Figure 3 that its display consists of two windows. The top lists the currently active processes, including the names of their owner's accounts. HandleEx supports two modes, handle and DLL; the bottom window lists either the handles that the process selected in the top window has opened or the DLLs that the process has loaded, depending on the mode selected. HandleEx also has a search feature that will quickly find which processes have particular handles opened or DLLs loaded. Version 2.0 of this utility also includes a "kill" command for terminating processes and a "properties" command for viewing additional information about a process or DLL.

I downloaded HandleEx and ran it along side REGMON to see what files and DLLs were being accessed during application startup. Notice in Figure 3 how DBAPP.EXE is highlighted in the top section and how I was able to find that a second child executable called HOSTSET.EXE was being called by DBAPP. I then highlighted HOSTSET.EXE and found a file called HOSTSET.LDB that it was reading to obtain the IP address. I changed the contents of the LDB file and was able to access the SQL server from the troubled client.

The programmer apologized profusely about forgetting to tell us about that little executable and associated setup file; he had forgotten that he had incorporated that utility late one night after an apparently rough day at the office.

SUMMARY

There are many times that IT professionals exclaim "Wouldn't it be nice if there was some little utility that would do _____?" Sometimes, the need for that utility is so great that you contemplate writing your own. However, before you do, be sure to check out for their up-to-date offerings.

I applaud the abilities of Mark Russinovich and Bryce Cogswell, the programmers of these utilities, for their generous philosophy of making their utilities easy to obtain and use. ts

NaSPA member Guy C. Yost is the owner of Redstone Consulting, an IT management consulting firm in New York. He has authored several books on networking for Que Publishing, including Learning NetWare 4.1, *and* NetWare 4.1 SmartStart, *and contributes to* Technical Support *magazine as an author, columnist and technical editor. Guy also develops and conducts seminars on networking with Windows NT, UNIX, NetWare and Internet/intranet technologies across the United States and Canada. He can be reached at (518) 674-5606 or gyost@logical.net.*