



Occasionally a COBOL program becomes "hung". Examining the process via the "ps" command may show that it is accumulating CPU time, in other words, "spinning", or instead that it is not accumulating CPU time, and is just sitting (perhaps waiting for some event). In either case, it does not exit normally, and must be manually killed. It is important to debug hung COBOL processes, when possible, so as to find the root cause and avoid such problems in the future.

Fault Finder is explained in the Server Express documentation, available on the Micro Focus web site. Start at:

[SupportLine Web Service](#)

Then select "Documentation", and then select a version of Server Express. From there, click "Debugging Guide", then "Fault Finder". It is important to review this section of the documentation before employing this tool.

To collect Fault Finder output, first make sure the Micro Focus runtime tuneable "faultfind_level=20" is specified.

Runtime tuneable are specified in a "runtime configuration file". As explained in the Server Express documentation, a run-time configuration file is a text file you can edit with a standard text editor. It is an optional file (no error will be issued if it does not exist) and its default name is \$COBDIR/etc/cobconfig. But as an alternative to using this global config file, you can temporarily specify a file in another location using the environment variable COBCONFIG; for example:

```
COBCONFIG=/home2/myfiles/myconfig  
export COBCONFIG
```

You have to make sure the COBCONFIG environment variable is set within the environment of the COBOL process to be debugged. The syntax within the file is like this:

```
set faultfind_level=20
```

Having established the configuration file, run for awhile, and the next time a hung COBOL process occurs, log into a separate window or session on the machine, take note of the PID of the hung program, and invoke "cobffnd" on that PID, using a command similar to:

```
cobffnd -ffault_find_report_pid%p.txt <pid>
```

You must be logged in as the same user who owns the hung process, or as root, else you won't have permission to attach to the process and debug it. If using a root login, remember you must have COBDIR, and the library path (LD_LIBRARY_PATH or LIBPATH or SHLIB_PATH depending on the UNIX variant), and PATH set up to use Server Express.

The above cobffnd command creates an output file in the current directory (of the person issuing the cobffnd command, not of the hung COBOL process). The pertinent process-id is specified as an argument on the cobffnd command line.

If the COBOL program is in the form of a .int or .gnt, the PID you are interested in will actually be the "rts32" or "rts64" from \$COBDIR/bin, which is the COBOL runtime system executable that dynamically loads the .int or .gnt.

The cobffnd command will not kill the process, so right after running cobffnd, you can invoke the system debugger on the same PID and obtain a stack trace. For example on AIX where the system debugger is named "dbx":

```
dbx -a <pid>  
(dbx) where
```

Capture (i.e. copy-and-paste) the stack trace shown on the screen by the "where" command. Then type "quit" from the (dbx) prompt; this will kill the hung process. Or if you'd like, you can exit the debugger without killing the process by using the dbx "detach" command.

On Red Hat Enterprise Linux, for example, the debugger is named "gdb", and for arguments it wants the program name then the pid, for example:

```
gdb rts32 <pid>
```

The command within gdb to reveal the stack trace, is "info stack".

Debugging information obtained this way may directly reveal the root cause of the problem, but if not, it can at least provide valuable clues. When working with Micro Focus Supportline, obtain one or two examples of hung processes, with their fault finder output and stack trace output, and attach them to the Supportline incident ... this will provide valuable information.

NOTE: *this is based on Knowledge Base article reference: 25388*