How do I turn on debug trace to produce a log file?

Question:

Is there a runtime trace log feature? How do I turn it on? What configuration options are available?
If I have multiple processes running, is there any way to distinguish which log file output goes to, or does all output go to the same log file?

Answer:

To produce a trace log, set iscobol.tracelevel to a non-zero value. With the default value of 0, the system does not create a log since it would be empty.

The following are some useful settings:

iscobol.tracelevel=3 includes config settings and program starts and ends.
iscobol.tracelevel=7 includes config, program starts/ends, and paragraph starts/ends
iscobol.tracelevel=11 includes config, program starts/ends and file i/o (i.e. everything except for paragraph starts/ends)
iscobol.tracelevel=15 is the maximum trace setting. This includes config, program and paragraph starts/ends, and file i/o.

Set iscobol.logfile to a file of your choice.

The isCOBOL framework uses the java.util.logger package, and there are many configuration options.

For example, you can specify "%h" in the iscobol.logfile and it will be replaced by the user's home directory.

You can specify a "%u" in the iscobol.logfile and it will be replaced with a unique number at runtime to resolve conflicts.

The %u is replaced by a unique number, 0, 1, 2, ... The logic to determine the unique number is to use the lowest number that is not in current use by a process. The log files are "locked" by creating a ".lck" file, and are unlocked by deleting that ".lck" file. So if the filename is fred%u.log and fred0.log.lck exists, the process will create fred1.log (and fred1.log.lck). If fred0.log.lck does not exist then the process will overwrite fred0.log. It does not get appended to.

(The javadoc for FileHandler says "If the FileHandler tries to open the filename and finds the file is currently in use by another process it will increment the unique number field and try again. This will be repeated until FileHandler finds a file name that is not currently in use")

See http://java.sun.com/javase/6/docs/api/index.html?java/util/logging/FileHandler.html for other pattern components and logging properties.

On UNIX/Linux to include a process ID in the log filename, create a shell script and use $$ to substitute the process id of the current shell. For example, to create a log file named myapp followed by an underscore and the process id of the shell, specify "-Discobol.logfile=myapp_$$.log"

on your java command line.

Note that these log files will accumulate until they are deleted or until the process id wraps around.

If you do not set iscobol.logfile then the trace log will be written to $ISCOBOL/bin/isrun.log where $ISCOBOL is the isCOBOL installation directory.

The iscobol.logfile value should not be enclosed in double-quotes, even if there are embedded spaces in the path. On Windows, you can use forward slashes or double-backslashes. For example, any of the following will work:

```
iscobol.logfile=C:\\parent dir\\sub dir\\myapp.logiscobol.logfile=C:/parent dir/sub dir
/myapp.logiscobol.logfile=/parent dir/sub dir/myapp.logiscobol.logfile=%h/myapp%u.log
```