Batch Script - Debugging

Very often than not you can run into problems when running batch files and most often than not you would need to debug your batch files in some way or the other to determine the issue with the batch file itself. Following are some of the techniques that can help in debugging Batch Script files.

Error Messages

To discover the source of the message, follow these steps −

Step 1 − REM out the @ECHO OFF line, i.e. REM @ECHO OFF or :: @ECHO OFF.

Step 2 − Run the batch file with the required command line parameters, redirecting all output to a log file for later comparison.

```
test.bat > batch.log 2>&1
```

Step 3 − Search the file batch.log for the error messages

Step 4 − Check the previous line for any unexpected or invalid command, command line switch(es) or value(s); pay special attention to the values of any environment variables used in the command.

Step 5 − Correct the error and repeat this process until all error messages have disappeared.

Complex Command Lines

Another common source of errors are incorrectly redirected commands, like for example "nested" FIND or FINDSTR commands with incorrect search strings, sometimes within a FOR /F loop.

To check the validity of these complex commands, follow these steps −

Step 1 − Insert "command check lines" just before a line which uses the complex command set.

Following is an example wherein the ECHO command is inserted to mark where the output of the first TYPE command ends and the next one starts.

```
TYPE %Temp%.\apipaorg.reg
ECHO.============================================== TYPE %Temp%.\apipaorg
| FIND
```

```
"[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TCPIP\Parameters\Int
```

Step 2 − Follow the procedure to find error message sources described above.

Step 3 − Pay special attention to the output of the "simplified" command lines: Is the output of the expected format? Is the "token" value or position as expected?

Subroutines

Subroutines generating error messages pose an extra "challenge" in finding the cause of the error, as they may be called multiple times in the same batch file.

To help find out what causes the incorrect call to the subroutine, follow these steps −

Step 1 − Add and reset a counter variable at the beginning of the script −

```
SET Counter = 0
```

Step 2 − Increment the counter each time the subroutine is called, by inserting the following line at the beginning of the subroutine

```
SET /A Counter+=1
```

Step 3 − Insert another line right after the counter increment, containing only the SET command; this will list all environment variables and their values.

Step 4 − Follow the procedure to find error message sources described above.

Windows Versions

If you intend to distribute your batch files to other computers that may or may not run the same Windows version, you will need to test your batch files in as many Windows versions as possible.

The following example shows how to check for various operating system versions to check the relevant windows versions.

```
@ECHO OFF
:: Check for Windows NT 4 and later

IF NOT "%OS%"=="Windows_NT" GOTO DontRun
:: Check for Windows NT 4
VER | FIND "Windows NT" >NUL && GOTO DontRun
:: Check for Windows 2000
VER | FIND "Windows 2000" >NUL && GOTO DontRun
```

```
:: Place actual code here . . .
:: End of actual code . . .
EXIT

:DontRun
ECHO Sorry, this batch file was written for Windows XP and later versions c
```