

Intelligently Launching Microsoft Excel from SAS, Using SCL Functions Ported to Base SAS

Christopher A. Roper, National Committee for Quality Assurance, Washington, D.C.

Abstract

The SAS Companion for the Microsoft Windows Environment illustrates one technique for opening Microsoft Excel from within SAS using DDE. This technique has several disadvantages. First, it does not check whether Excel is already open, it can unintentionally open multiple Excel sessions and this can be problematic. Second, it does not know with certainty when it has opened Excel, it relies on the SLEEP function to pause processing until a set amount of time has passed. Hopefully, the amount of time is sufficient for Excel to open successfully, but there is no guarantee. Finally, this set amount of time must be predetermined and sufficient for the worst case scenario (i.e. the slowest machine on which the code might possibly be run). This means the pause is always set higher than the minimum amount of time required for Excel to open, does not take advantage of the best resources available, and causes the end user to wait unnecessarily. While this technique was the best available when the SAS Companion for the Microsoft Windows Environment was published, the porting of some of the SCL functions to Base SAS has enabled another solution to this problem, one that is much more intelligent. With these functions it is possible to have SAS check if Excel is open. If Excel is not open, then open Excel, and continually check if Excel has finished opening successfully. As soon as Excel has finished opening, then proceed with the SAS processing. This technique will not unintentionally open multiple copies of Excel, and will not waste time by pausing processing for a set number of seconds.

System Options:

OPTI ONS NOXSYNC;

The NOXSYNC execution option will cause SAS to continue processing while Excel is opening. This allows the SAS program to continue testing to see if Excel is open, and then proceed as soon as Excel has completely opened.

Excel DDE Triplet:

The syntax for a standard DDE triplet takes the following form:

'application-name/topic!item'

Microsoft defined the DDE topic SYSTEM to enable commands to be executed within Excel. Because there is no further refinement necessary to submit commands to Excel, there is no value to provide to the ITEM parameter. Thus the DDE triplet for sending commands to Excel contains only two components; i.e.

```
filename cmdexcel DDE  
' EXCEL | SYSTEM ;
```

FOPEN SCL Function:

The SCL function FOPEN has been ported to Base SAS. This function is designed to open external files. Frequently this is done to read and/or write text files. The first argument to the function is the fileref of the external file. The second argument is the open mode, i.e. the desired type of access to the external file. The default is I, (INPUT), which allows read-only access. However, using a sequential access method such as S for sequential input (read only) or W for sequential update (read/write), allows capturing the information that SAS can use to determine when Excel is open. If Excel is open the value returned by the FOPEN function will have a non-negative integer value. If Excel is not open, the FOPEN function will return a value of zero. The following code snippet illustrates the syntax required to determine if Excel is open.

```
fid = fopen(' CMDEXCEL' , ' S' );
```

The START Command:

If the value of FID (returned by the FOPEN function) is zero, we know that Excel is not

open. Therefore, we need to execute some command to launch Excel. The DOS command START is very useful in opening Windows based applications. It uses the Windows Registry file to determine the location of the executable file for the application referenced (in this case, Excel). Therefore, there is no need to know or determine the exact path of the executable file for Excel. This doesn't work with all applications, but it does with many, including Microsoft applications and the SAS System.

```
rc=system("Start Excel");
```

Safety Nets:

Because no system is perfect, a safety net is always a good idea. If FID were to never get a value, this code could loop indefinitely (suppose Excel is not installed on your computer!). Therefore, this next code segment will set a value for FID after a predetermined amount of time has passed. The variable STOP will be a SAS datetime variable set to 5 seconds greater than the current time. If Excel has not opened within this 5 seconds, the program will stop executing.

```
start = datetime();  
stop = start + 5;
```

Use the FOPEN SCL function again to test if Excel is open. As soon as Excel is open, FID will get a positive integer value. The variable TIME will continually be updated with the current SAS datetime value. This will be used to test if the process is taking longer than the maximum time allowed, in this case 5 seconds.

```
do while (fid le 0);  
  
fid = fopen('CMDEXCEL', 'S');  
time = datetime();
```

If TIME exceeds the 5 seconds limit, the program assigns the value of TIME to FID. This will cause the DO WHILE loop to terminate, and prevent an infinite looping process.

```
if time ge stop then  
    fid = time;  
  
end; /* do while (fid le 0) */
```

Conclusion

This program demonstrates how to use the SAS System to intelligently open Excel. With this technique, a fast machine does not have to be paused for the length of time it would take the slowest machine to open Excel. This program can control whether or not to open multiple copies of Excel because it knows whether or not a copy is already open. If Excel is already opened, it does not force the end user to wait for Excel to be opened a second time.

Author Contact

Christopher A. Roper
National Committee for Quality Assurance
2000 L St. N.W. Suite 500
Washington, D.C.
(202) 955-3562
croper@ncqa.org

Acknowledgements

SAS is a trademark and registered trademark of SAS Institute in the USA and other countries

Excel is a trademark and registered trademark of the Microsoft Corporation in the USA and other countries