MWSUG 2019 - PO-045

Have Your SAS® Program And Schedule It Too!

Mario Tejada, Data Solutions Developer

NORC @ the University of Chicago

ABSTRACT

In some SAS environments, it is common to use the Windows Task Scheduler to launch production jobs on a set schedule. In order to create a basic task, one needs to manually open Task Scheduler and enter the desired parameters for the job. This paper will explore the possibility of using SAS to drive the scheduling process. Instead of going into Task Scheduler, the programmer can use SAS to set the parameters of a job and programmatically register that task in Windows using PowerShell.

This paper assumes a basic understanding of SAS data step programming, SAS Macros and administrator-level access to run Windows Operating system (PowerShell) commands. Some familiarity with creating scheduled tasks is assumed. Programs on this paper were tested using SAS 9.4M5 and PowerShell version 5.1 on Windows 10.

INTRODUCTION

Setting up a scheduled task on Windows usually takes several steps: (1) Opening Task Scheduler, (2) Creating a task by following several more steps to set various parameters on the Task Wizard and (3) Registering the task by supplying the credentials to use for the job. The number of steps can be cut down by using PowerShell's Register-ScheduledJob cmdlet:

```
Register-ScheduledJob -Name "MyScheduledJob" `
                      -FilePath "Path_To_Scriptfile.ps1" ` <#SAS-generated#>
                      -Trigger $datetime_triggers `
                      -ScheduledJobOption $job_options `
                      -Credential $cred
```

Note that the tilde "`" character was used to split lines for better readability.

DISCUSSION

A SAS data _null_ step will be used to generate the PowerShell code shown above and then will be executed using an unnamed pipe. The syntax for this device-type pipe is shown below:

FILENAME *fileref* PIPE '*program-name*' *option-list*;

Fileref is any valid fileref, as described in Referencing External Files.

PIPE is the device-type keyword that tells SAS that you want to use an unnamed pipe.

program-name specifies the external Windows application program.

option-list can be any of the options valid in the FILENAME statement, such as the LRECL= or RECFM= options.

From http://support.sas.com/documentation/cdl/en/hostwin/63047/HTML/default/viewer.htm#n16puwsro9pakgn1iamv1vwvaqx6.htm

In the sections that follow, code snippets that make up the full program will be described. As a simple example, we attempt to programmatically schedule this program:

```
Program name: print_SAS_Cars_dataset.sas

ods pdf
  file= 'C:\Users\username\Documents\SAS_Paper_2019\practice_output\test_output.pdf';
 proc print data=sashelp.cars; run;
ods pdf close;
```

First, we set up a PowerShell script that will execute the SAS program above:

```
Program name: Path_To_Scriptfile.ps1

$sasexe94="E:\Program Files\SASHome\SASFoundation\9.4\sas.exe"
$file_path=" C:\Users\username\Documents\SAS_Paper_2019\print_SAS_Cars_dataset.sas";

Start-Process "$sasexe" -ArgumentList $file_path
```

Second, the parameters for the scheduled job are stored in macro variables:

```
%let username = domain\username;
%let working_folder = C:\Users\username\Documents\SAS_Paper_2019;
%let jobname= MyScheduledJob;
%let freq = daily; *options: daily, hourly, weekly;
%let time = 1:45pm;
```

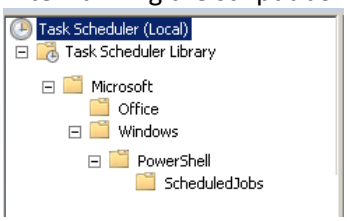
Third, set up the data _null_ step that would generate the PowerShell script which would register the scheduled job:

```
filename process pipe "powershell -noprofile -executionpolicy bypass -Command ""
.\PS_Schedule_MySAS_program.ps1""";
data loglines;
      infile process firstobs=4;
      length longtext $ 1000;
      input @1  longtext;
run;

**[1.1] remove password file **;
X "del C:\Users\username\Documents\SAS_Paper_2019\SAS_CRED\ MY_PSCRED.txt";
```

Fourth, call the above script using a pipe to register the job and then delete the temporary text file that stores the encrypted password.

After running the script above, the scheduled job will be added to Task Scheduler:



Important notes about this script

The user account should have admin rights on the machine where the scheduled job will be run. When naming the job, if there is an existing task with the same name, then the task registration fails.

Please see references for information on creating the temporary credentials file.

CONCLUSION

This e-poster outlined the steps needed to schedule a SAS program by leveraging PowerShell's Register-ScheduledJob cmdlet. Armed with this information, you can now have your SAS program and schedule it too!

REFERENCES

More information about Register-ScheduledJob can be found on the Microsoft website:
https://docs.microsoft.com/en-us/powershell/module/psscheduledjob/about/about_scheduled_jobs?view=powershell-5.1

More information
https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.security/convertto-securestring?view=powershell-6#examples

ACKNOWLEDGMENTS

The author would like to thank Joe Matise for his support and valuable input to this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Mario Tejada
Enterprise: NORC @ the University of Chicago
Address: 55 E Monroe St.
City, State ZIP: Chicago, IL 60603
Work Phone: 312-357-3894
E-mail: tejada-mario@norc.org
Web: www.norc.org

Appendix

Full Code

```
%let username = domain\username;
%let working_folder = C:\Users\username\Documents\SAS_Paper_2019;
%let jobname= MyScheduledJob;
%let freq = daily;
%let datetime = 1:45pm;


** Script that actually runs the SAS program **;
%let runsas_scriptfile = 'C:\Users\username\Documents\SAS_Paper_2019\
Path_To_Scriptfile.ps1';

** Script for registering Scheduled Task **;
%let reg_scriptfile = &working_folder\PS_Schedule_MySAS_program.ps1;

X "cd &working_folder.";

filename script "&reg_scriptfile.";
data _null_;
  file script ;
  put "$my_cred_folder = '&working_folder.\SAS_CRED'";
  put "$username = '&username.'";
  put '$password = Get-Content "${my_cred_folder}\MY_PSCRED.txt" | ConvertTo-
SecureString';
  put "$cred = new-object -typename System.Management.Automation.PSCredential -
argumentlist $username, $password";
  put "$opt = New-ScheduledJobOption -RequireNetwork -RunElevated";
  put "$newtrigger=New-JobTrigger -Daily -At 2:45PM";
  put "Register-ScheduledJob -name TEST_SAS_Generated_Scheduled_Task3 -Trigger
$newtrigger -Credential $cred -FilePath &runsas_scriptfile. -ScheduledJobOption
$opt";
run;
```