PharmaSUG 2018 - Paper QT-06

When Powerful SAS® Meets PowerShell™

Shunbing Zhao, Merck & Co., Inc., Rahway, NJ, USA

Jeff Xia, Merck & Co., Inc., Rahway, NJ, USA

Chao Su, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

PowerShell is an MS Windows-based command shell for direct interaction with an operating system and its task automation. When combining the powerful SAS programming language and PowerShell commands/scripts, we can greatly improve our efficiency and accuracy by removing many trivial manual steps in our daily routine work as SAS programmers. This paper presents five applications we developed for process automation. 1) Automatically convert RTF files in a folder into PDF files, all files or a selection of them. Installation of Adobe Acrobat printer is not a requirement. 2) Search the specific text of interest in all files in a folder, the file format could be RTF or SAS Source code. It is very handy in situations like meeting urgent FDA requests when there is a need to search required information quickly. 3) Systematically back up all existing files including the ones in subfolders. 4) Update the attributes of a selection of files with ease, i.e., change all SAS code and their corresponding output including RTF tables and SAS log in a production environment to read only after database lock. 5) Remove hidden temporary files in a folder. It can clean up and limit confusion while delivering the whole output folder. Lastly, the SAS macros presented in this paper could be used as a starting point to develop many similar applications for process automation in analysis and reporting activities.

INTRODUCTION

Windows PowerShell is a MS Windows-based command shell that provides automation capabilities to end users. Many trivial manual steps in file management could be eliminated by using this valuable utility. SAS provides users the capability to call PowerShell commands by using FILENAME PIPE syntax, or to create a script to run by using DATA OUTPUT. When combining the powerful SAS programming language with PowerShell's rich scripting language, we are able to improve the efficiency and accuracy greatly by eliminating many trivial manual steps in our daily routine work as SAS programmers. To promote the usage of PowerShell in SAS users, Chris Hemedinger from SAS Institute provided a few useful examples in an official blog to explain how to combine SAS with Windows PowerShell. This paper presents five scenarios of using SAS to call PowerShell functions, which can be served as a starting point to get benefits from both PowerShell and SAS for SAS programmers in the pharmaceutical industry.

DISCUSSION

Firstly, SAS must have access to the PowerShell. Various shortcuts are found in Start > All Programs > Accessories > Windows PowerShell. After PowerShell window appears, the user has to run the following commands to update some security policy: Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser. The PowerShell Command/Script is now effectively enabled.

Scenario 1

A batch of RTF files are requested to be converted into PDF files and delivered to customers. A macro (some key part) for this task was developed as shown below.

```
%macro ps0rtf2pdf( in_path=
                   ,out_path=
                   ,rtfname=
) / minoperator;

…

* Generate PowerShell Script;
%put NOTE-Generating Powershell Script &workdir\_rtf2pdf.ps1;
data _null_;
  length line $255 len 8;
  file "&workdir\_pstemp.ps1" lrecl=255;
  put " ";
  line = "$word_app = New-Object -ComObject Word.Application";
  len = length(line);
  put line $varying1024. len;
  line = "Get-ChildItem -Path &in_path -Filter &rtfname..rtf | ForEach-Object
{";
  len = length(line);
  put line $varying1024. len;
  line = "$document = $word_app.Documents.Open($_.FullName)";
  len = length(line);
  put line $varying1024. len;
  line = "$pdf_filename = ""&out_path.\$($_.BaseName).pdf""";
  len = length(line);
  put line $varying1024. len;
  line =
"$Document.ExportAsFixedFormat($pdf_filename,17,$false,1,0,0,0,0,$true,$true,1
,$true,$true,$false)";
  len = length(line);
  put line $varying1024. len;
  line = "$document.Close()}";
  len = length(line);
  put line $varying1024. len;
  line = "$word_app.Quit()";
  len = length(line);
  put line $varying1024. len;
run;

* Executing Script;
data _null_;
x powershell -file "&workdir\_pstemp.ps1" "";
Run;
…
%mend ps0rtf2pdf;
```

Using this macro, a group of selective RTF files can be converted into PDF files with bookmarks kept automatically. Figure 1 shows one example of the application of this macro.

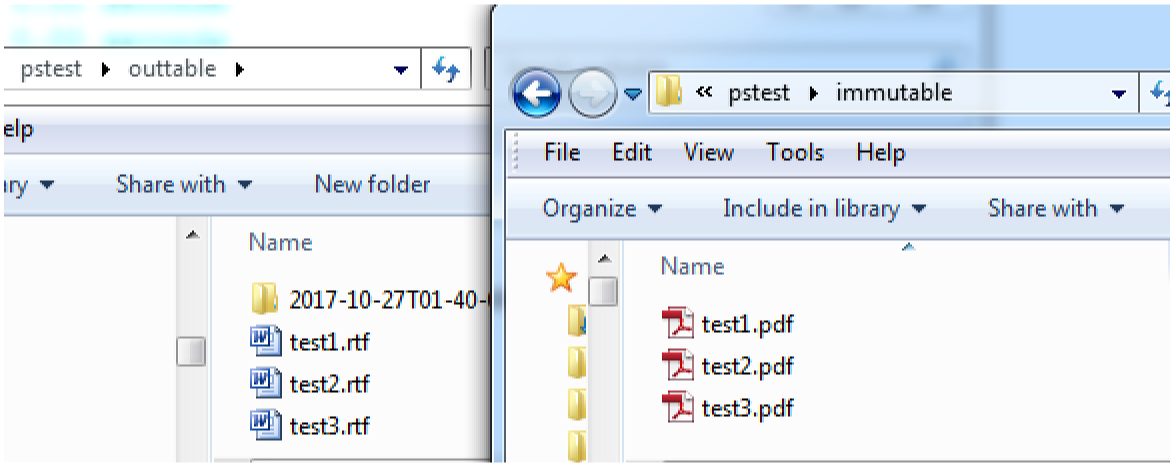


Figure 1. Screenshot of an application to automatically convert RTF files into PDF files in a folder.

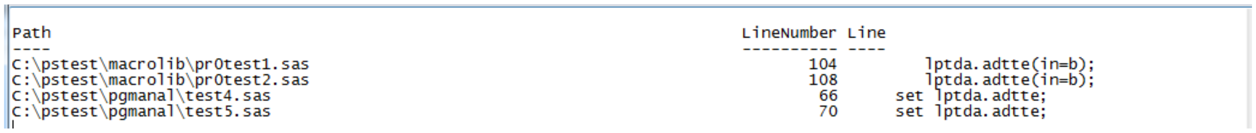
Scenario 2

In SAS programs, search is needed to be applied to check whether a certain ADaM dataset is used or a certain SAS proc is executed. It is especially necessary in urgent situations such as meeting FDA requests for searching required information quickly. The SAS program below can help to complete this task easily.

```
%let loc=C:\pstest\;
%let path=C:\pstest\macrolib;
%let file=testall.txt;
%let pattern=%str(\b\w+\b.adtte);
%let file_type=sas;

options noquotelenmax;
filename process pipe
  "powershell -Command ""get-childitem -path &path -filter *.&file_type -
recurse| select-string -pattern &pattern| format-table  path, linenumber, line
-Auto |
  Out-File &loc.&file -width 512""";
data _null_;
    infile process lrecl=600;
run;
```

In the program codes above, a regular expression can be assigned to the pattern and the PowerShell command is compatible with the regular expression. It provides more flexibility to our search program. Figure 2 demonstrates one exemplary result in which programs are found where ADTTE was referenced.



```
Path                                                     LineNumber Line
----                                                     ---------- ----
C:\pstest\macrolib\pr0test1.sas                                 104         lptda.adtte(in=b);
C:\pstest\macrolib\pr0test2.sas                                 108         lptda.adtte(in=b);
C:\pstest\pgmanal\test4.sas                                      66     set lptda.adtte;
C:\pstest\pgmanal\test5.sas                                      70     set lptda.adtte;
```

Figure 2. Screenshot of the searching result in all SAS files in a folder: regular expression `pattern=%str(\b\w+\b\.adtte)`

Scenario 3

It is necessary to systematically back up all existing files, including the ones in the subfolders. The below macro could be very handy and efficient in such a case. See below for the key part of the macro.

```
%macro ps0backup( root_path=
                 ,sub_lst=
) / minoperator;

…

* Generate PowerShell Script;
%put NOTE-Generating Powershell Script &root_path\_ps0bk.ps1;
data _null_;
  length line $255 len 8;
  file "&root_path\_ps0bk.ps1" lrecl=255;
  put " ";
  line = "Param([Parameter(Mandatory=$true)][Alias('sf')][String]$subfolder)";
  len = length(line);
  put line $varying1024. len;
  line = "$source = ""&root_path.\$subfolder""";
  len = length(line);
  put line $varying1024. len;
  line = "$date=get-date -f yyyy-MM-ddThh-mm-ss";
  len = length(line);
  put line $varying1024. len;
  line = "new-item -path $source -type directory -name $date";
  len = length(line);
  put line $varying1024. len;
  line = "robocopy $source  $source\$date ";
  len = length(line);
  put line $varying1024. len;
run;

* Executing Script;
%let totsub=%sysfunc(countw(&sub_lst,%str( )));
%do i=1 %to &totsub;
    %let subfolder=%scan(&sub_lst,&i,%str( ));
    %let root_path_=&root_path\&subfolder;
    %if %sysfunc(fileexist(&root_path_)) > 0 %then %do;
        data _null_;
        x powershell.exe -file %superq(root_path)\_ps0bk.ps1 -
sf %superq(subfolder);
        Run;
    %end;
    %else %do;
       %put &err1&err2: Specified subfolder, &root_path_, does not exist;
    %end;
%end;
data _null_;
x powershell.exe -command Remove-Item  "%superq(root_path)\_ps0bk.ps1";
Run;

%mend ps0backup;
```

Figure 3 gives one example of calling

```
%ps0backup( root_path=C:\pstest
           ,sub_lst=
           );
```

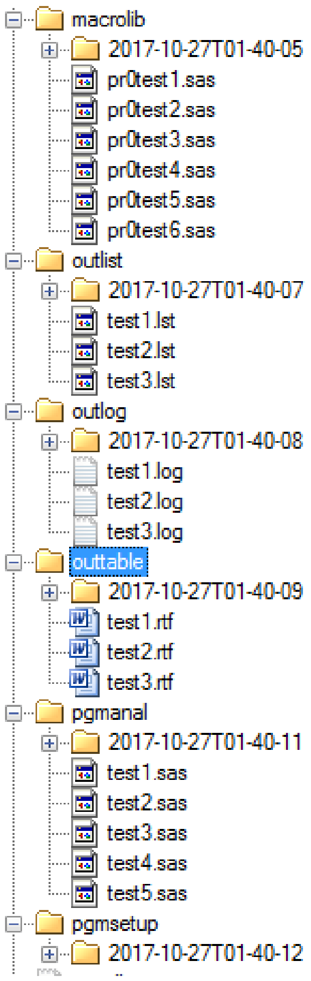

macrolib
2017-10-27T01-40-05
pr0test1.sas
pr0test2.sas
pr0test3.sas
pr0test4.sas
pr0test5.sas
pr0test6.sas
outlist
2017-10-27T01-40-07
test1.lst
test2.lst
test3.lst
outlog
2017-10-27T01-40-08
test1.log
test2.log
test3.log
outtable
2017-10-27T01-40-09
test1.rtf
test2.rtf
test3.rtf
pgmanal
2017-10-27T01-40-11
test1.sas
test2.sas
test3.sas
test4.sas
test5.sas
pgmsetup
2017-10-27T01-40-12

Figure 3. Screenshot of an application to systematically back up all existing files in subfolders with names in the date format.

Scenario 4

We need to change all SAS codes and their corresponding outputs including RTF tables and SAS logs in a production environment to “read-only” after the database lock. The below macro (only key parts are shown) will assist us to update the attributes of a selection of files.

```
%macro ps0readonly( root_path=
                   ,sub_lst=
                   ,type=
) / minoperator;

…

%let totsub=%sysfunc(countw(&sub_lst,%str( )));
%do i=1 %to &totsub;
    %let subvar=%scan(&sub_lst,&i,%str( ));
    %let root_path_=&root_path\&subvar;
   %if %sysfunc(fileexist(&root_path_)) > 0 %then %do;
        data _null_;
         %if %length(&type) eq 0 %then %do;
         x powershell.exe -command Set-ItemProperty -Path
"%superq(root_path_)\*.*" -Name IsReadOnly -Value $True;
         %end;
         %else %do;
         x powershell.exe -command Set-ItemProperty -Path
"%superq(root_path_)\*.%superq(type)" -Name IsReadOnly -Value $True;
         %end;
        Run;
    %end;
    %else %do;
       %put &err1&err2: Specified subfolder, &root_path_, does not exist;
    %end;
%end;

%mend ps0readonly;
```

Scenario 5

Removing any hidden temporary files in a folder is required. It can eliminate confusion while delivering the whole output folder. The below SAS program can help us to accomplish this task easily.

```
%let path=C:\pstest\outtable;
%let pattern=%str(~$*.rtf);

options noquotelenmax;
filename process pipe
  "powershell -Command ""get-childitem -path &path -filter &pattern -force -
recurse| remove-item -force""";

data _null_;
    infile process lrecl=500;
run;
```

CONCLUSION

As SAS programmers, we can improve our efficiency and accuracy greatly by eliminating many trivial manual steps in our daily routine work when starting PowerShell from SAS. SAS macros presented in this paper can help other users develop many similar applications for process automation in analysis and reporting activities.

REFERENCES

Chris Hemedinger SAS online Communities Blog.

ACKNOWLEDGMENTS

The authors would like to thank Cynthia He and Nancy Meng for their great support and valuable input to this paper.

CONTACT INFORMATION

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

Name: Shunbing Zhao
Enterprise: Merck
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607
Work Phone: 732-594-3976
Fax:
E-mail: shunbing.zhao@merck.com
Web: www.merck.com

Name: Jeff Xia
Enterprise: Merck
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607
Work Phone: 732-594-6439
Fax:
E-mail: jeff.xia@merck.com
Web: www.merck.com

Name: Chao Su
Enterprise: Merck
Address: 126 E. Lincoln Avenue
City, State ZIP: Rahway, NJ 07065-4607
Work Phone: 732-594-6459
Fax:
E-mail: chao.su@merck.com
Web: www.merck.com