

Automate Secure Transfers with SAS® and PSFTP

Kyle Thompson, PPD, Morrisville, NC

Kenneth W. Borowiak, PPD, Morrisville, NC

INTRODUCTION

The ability to transfer files between remote systems has evolved over time from using File Transfer Protocol (FTP) to Secure File Transfer Protocol (SFTP), with the latter being a safer technology. We will discuss scripting of activities related to moving files via SAS® code using the PSFTP, which is the PuTTY SFTP client. PuTTY is a free open-source implementation of Secure Shell protocol (SSH). See the References at the end of the paper for the URL to download the executable.

BASIC WINDOWS COMMAND LINE IN SAS

We briefly review Windows Command Line syntax and its use within SAS programs. Basic commands include making and deleting folders, copying contents, referencing system paths or changing from one directory to a different directory, listing the contents of a file or directory, and executing a program. Similar commands are used with PuTTY.

The make directory (MKDIR) command can be used in conjunction with the change directory (CD) command to create a file or directory in a specific location. Upon opening the command line (Windows + R -> 'cmd' -> ENTER) the current directory may not be the desired directory to which a new file will be added. To navigate to the desired location of a new file, absolute and relative references can be used. To delete a directory or folder, (RMDIR) is used in place of (MKDIR). Note that (RMDIR) will remove a folder and ALL OF ITS SUBDOLDERS AND FILES. Alternatively, the (DEL) command deletes single files (and with the modifier /P will include a confirmation prompt). Another basic command line task is to copy the contents of one directory or folder into a different one. Prior to doing this, however, listing the contents of a directory or folder can be useful, and can be done with the directory (DIR) command. Files can be copied one at a time, in groups, or by prefix/suffix.

There are several ways to use the command line within a SAS program, such as the SYSTEM function, CALL SYSTEM routine, X statement/command, and %SYSEXEC. We will focus on using the SYSTEM function for issuing commands, as it allows for capturing of a return code for error trapping. System options such as NOXWAIT /XWAIT and XSYNC/NOXSYNC can be used to control how SAS interacts with the command line. Whether or not a user is required to manually exit the command prompt and whether or not control does not return to SAS until after the command line command has executed are, respectively, determined by these options.

We include the following example: Within a SAS session, create a new folder on the user desktop containing all .RTF files with a name beginning with 'T' from a folder named "TLF" in a different directory, and output a text file in the new folder with a list of all .RTF files that were copied and the date of their last modification or update.

For the following example, clarity is gained at the expense of efficiency. In other words, although the below code is not efficient, it is intended to be as clear as possible.

```
options noxsync noxwait;
**** Assign parameters ****;
%let newname      = New Fold; *New folder name;
%let newfoldloc   = C:\Users\sasuser\desktop; *New folder location;
%let copyfromloc  = U:\StudyFolder; *Copy-from folder directory;
%let copyfold     = TLF ; *Copy-from folder name;
%let fileextn     = rtf ; *Extension of files to copy;
```

```

%let prefix = T ; *Prefix of files to copy;

data _null_;
** Create new folder in given location **;
cmd1_1 = 'MKDIR "'; *Create the folder to copy into;
cmd1_2 = "&newfoldloc.\&newname."; *New folder name and location;
cmd1_3 = '" ' ; *Complete command for command line;

cmd1 = cmd1_1 || cmd1_2 || cmd1_3;
put cmd1; *Note embedded spaces in filepath need to be in double quotes;

rc1 = system( cmd1 ) ; *Use system function to execute command;
call sleep(5,1) ; *Put SAS to sleep for five seconds;
putlog 'WARNING- Return code is ' RC1 ; *Output execution result

** Obtain a .txt list of all files of a given prefix and/or suffix **;
cmd2_1='DIR /T:W ' ;
*Directory listing of date/time of last modification;
cmd2_2="&copyfromloc.\&copyfold.\&prefix.*.&fileextn." ;
*Location of source folder with name conditions;
cmd2_3=' > "' ; *Complete folder path and denote 'write to' ;
cmd2_4="&newfoldloc.\&newname."; *Destination location;
cmd2_5= '\directory.txt' ; *Location to place directory summary;
cmd2_6='"' ; *Complete command for command line;

cmd2=cmd2_1 || cmd2_2 || cmd2_3 || cmd2_4 || cmd2_5 ||cmd2_6 ; *Command to
be passed to command line;

put cmd2 ; *Write command to log;

rc2=system( cmd2 ) ; *Use system function to execute command;

call sleep(5,1) ; *Put SAS to sleep for five seconds;
putlog 'WARNING- Return code is ' RC2 ; *Output execution result;

** Copy files **;
cmd3_1='COPY /Y ' ; *Copy desired files, suppress overwrite prompt;
cmd3_2="&copyfromloc.\&copyfold.\&prefix.*.&fileextn." ;
*Location of source folder with name conditions;
cmd3_3=' "' ; *Complete folder path ; cmd3_4="&newfoldloc.\&newname.";
*Destination location;
cmd3_5='"' ; *Complete command for command line;

cmd3=cmd3_1 || cmd3_2 || cmd3_3 || cmd3_4 || cmd3_5 ;
*Command to be passed to command line;

put cmd3 ; *Write command to log;

rc3=system( cmd3 ) ; *Use system function to execute command;

call sleep(5,1) ; *Put SAS to sleep for five seconds;
putlog 'WARNING- Return code is ' RC3 ; *Output execution result
run ;

```

The log output is:

```

MKDIR "C:\Users\sasuser\desktop\New Fold"
Return code is 0
DIR /T:W U:\StudyFolder\TLF\T*.rtf > "C:\Users\sasuser\desktop\New Fold\directory.txt"
Return code is 0
COPY /Y U:\StudyFolder\TLF\T*.rtf "C:\Users\sasuser\desktop\New Fold"
Return code is 0

```

BASIC PuTTY COMMANDS

Typing the name of the PSFTP executable into the command line will open a PSFTP prompt. If `PSFTP.exe` resides on the `U:` drive then use:

```
U:\>psftp.exe
psftp: no hostname specified; use "open host.name" to connect
psftp> _
```

If the executable is not on the `U:` drive then the change directory `CD` command can be used to navigate to the location of the PSFTP executable. Then the executable's name is provided or the full path of the executable can be used.

To start and end a PSFTP session, the commands `OPEN` and `QUIT` are used, respectively. A session can begin at the PSFTP prompt (by typing in the prompt above) or can be opened along with calling the executable.

```
U:\>psftp.exe open username@host.name.
```

```
PuTTY Secure File Transfer (SFTP) client
Release 0.67
Usage: psftp [options] [user@]host
Options:
  -V          print version information and exit
  -pgpfp      print PGP key fingerprints and exit
  -b file     use specified batchfile
  -bc         output batchfile commands
  -be         don't stop batchfile processing if errors
  -v          show verbose messages
  -load sessname Load settings from saved session
  -l user     connect with specified username
  -P port     connect to specified port
  -pw passw   login with specified password
  -1 -2       force use of particular SSH protocol version
  -4 -6       force use of IPv4 or IPv6
  -C          enable compression
  -i key      private key file for user authentication
  -noagent    disable use of Pageant
  -agent      enable use of Pageant
  -hostkey aa:bb:cc:... manually specify a host key (may be repeated)
  -batch      disable all interactive prompts
  -sshlog file
  -sshrawlog file log protocol details to a file
```

Type `QUIT` in the PSFTP prompt to exit.

```
psftp> quit
U:\>_
```

In addition to the basic `OPEN` PuTTY command, parameters can be used to specify passwords, username, port numbers, and/or a batch file.

PuTTY Parameter	Purpose
-l	Specifies a user name
-pw	Specifies a password
-p	Specifies a port
-b	Specifies a file that contains a sequence of commands to be used

For example, if a password “weak123” is used to sign into ‘host.name’ with user name ‘doej’ and a file ‘command.bat’ is used to send commands to the PuTTY command prompt, an initial command could be:

```
“U:\psftp.exe” host.name -l doej -pw weak123 -b command.bat
```

Similar to the discussion above with respect to the windows command prompt, a change directory (local and remote), make directory, remove directory, delete file, and directory listing command are available within the PuTTY SFTP interface. UNIX-based systems are case-sensitive, so be sure you refer to files and folders exactly as named.

Within a session, a local and remote directory are maintained. PuTTY commands will operate on the remote directory, which can be displayed using PWD and changed using CD. Similarly, a local directory can be displayed using LPWD and changed using LCD.

The delete (DEL), directory listing (DIR), make directory (MKDIR), and delete directory (RMDIR) commands work in the same way described for the windows command line and apply to the remote directory (which can be viewed using the PWD command and changed with the CD command)

The PUT, GET, REPUT, and REGET commands are used to transfer files. The PUT command will transfer a file from the local directory to the remote directory, in other words it can upload a file from a local computer to a desired server and the GET command will do the reverse. It can download a file from the server to a local computer. A name change is possible (in the upload or download processes) by including first the current name then the destination name after the PUT or GET command.

REPUT and REGET act similar to PUT and GET, respectively, and are used after a file transfer failure. If a partial file is uploaded or downloaded, REPUT and REGET, respectively, can be used to resume the failed transfer at the point at which the file was left.

As an example, suppose the local directory is C:\Program Files\SAS and the remote directory is /forest/tree/branch/leaf and we want to send the file dummy123.txt in the local directory C:\users\sasuser\desktop to the remote directory /comp2/user2/study and name it StudyInfo.txt then the following commands could be used:

```
LCD “C:\users\sasuser\desktop”  
CD “/comp2/user2/study”  
PUT dummy123.txt Studyinfo.txt
```

If instead we wanted to download StudyInfo.txt from the remote server to the local directory, renaming it dummy123.txt then we could use:

```
LCD “C:\users\sasuser\desktop”  
CD “/comp2/user2/study”  
GET Studyinfo.txt dummy123.txt
```

If either the upload or downloads failed, I could follow-up with the command:

```
REPUT dummy123.txt Studyinfo.txt
```

or

```
REGET dummy123.txt Studyinfo.txt
```

to resume the upload or download, respectively.

USING SAS TO AUTOMATE THE TRANSFER OF FILES WITH PuTTY

With the basics of the SAS-Command Line interface and PuTTY parameters/commands covered, the two can be used in conjunction to automate the secure transfer of files. To begin, we suggest keeping username and password information separate from the SAS program used to automate the transfers. In the outline we cover, this information is located in a separate file (in a secure location, which can be accessed by only approved users).

A four part process completes the automation of transfers. First, a secure, external file is set (which can be accessed, maintained, and updated by only approved users). When the SAS transfer program executes, it pulls in the (updated) username and password information from the external file, which it is suggested will only occur for approved users. Next, a system task can be scheduled to run the SAS transfer program on an ongoing basis. Third, a batch file is created (this needs to occur only once, during an initial set-up of the transfer parameters, however, could be included in each run of the transfer program if the parameters are dynamic, for example if the current date determines the file type(s) to upload/download). Lastly, with the batch file ready and access information loaded, the proper PuTTY command(s) can be executed during the SAS transfer program's scheduled execution.

A simple (secure) text file could be used to create and maintain an updated user name and password for access to the server to be used during transfers. In our example, the (secure) text file `Study_Credentials.txt` has two records:

```
1      USER#DOEJ
2      PW#weak123
```

The "USER#" and "PW#" aid SAS in reading-in the correct information. If the name of the SAS program to automate secure transfers is called `Post_PSFTP.sas` then a Window's scheduled job can be set-up using the Task Scheduler with a .BAT file that calls this program.

Within a text editor, type the path to the local installation of SAS, then use the option `SYSIN` to select the desired SAS program (to see the LOG or LST file, the options `-LOG` and `-PRINT` can be used, respectively). In the below example, we name the .BAT file, "runPSFTP.bat".

```
"C:\Program Files\SAS 9.3\x86\SASFoundation\9.3\sas.exe" -SYSIN
"U:\Post_PSFTP.sas" -LOG "U:\Post_PSFTP.log" -PRINT
"U:\Post_PSFTP.lst"
```

Next, open the Task Scheduler, select "Create Task", name the task and set the schedule within the 'New Trigger' (press 'New...' in the Trigger tab) window. Browse to the appropriate .BAT file within the 'New Actions' window (press 'New...') in the 'Actions' tab (in our example, "runPSFTP.bat"). Finally, select 'OK' and save the task.

Now that the automatic transfers are scheduled, it remains to finalize the program that sends or downloads files via PSFTP. Although a batch file containing commands for PuTTY could be created outside of the SAS program used to automate the secure transfers, it can be useful to create it within the program if the commands or parameters passed to PSFTP.exe change over time or otherwise. In other words, if the commands sent to PuTTY can change, it is useful to write them within the transfer program.

We use the following SAS code to accomplish the transfer. After assigning some values to macro variables, the code grabs the username and password from the secure file `Study_Credentials.txt`. Note that it is assumed the credential file has a username following the string 'USER#' and a password following the string 'PW#'.

```
%let creds      = U:\Study_Credentials.txt ;
%let outfile     = Dummy123.TXT;
%let PSFTPloc    = U:\PSFTP.exe ;
```

```

*Get credentials;
data _null_ ;
  infile "&creds." truncover;
  input var $100. ;

  if upcase(var) eq: 'USER' then call symputx('username', scan(var,2,'#')) ;

  if upcase(var) eq: 'PW' then call symputx('password', scan(var,2,'#')) ;

run ;

*The destination path;
%let DESTINATION =/abc/tfil01/client/773/eload/elstudy/oclp9 ;

*The server connection name;
%let SERVER =abc43.xyz.local;

* Assumption is user is pushing a file from SOURCE to DESTINATION via PSFTP;
%let source=U:\&outfile.;

* Create the script/batch file for PuTTY;
%let BAT = PSFTP_bat;
filename btscpt "u:\&BAT..scr";
data _null_;
  file btscpt;
  put 'cd "' "&DESTINATION." '" ;
  put "put &source. " ;
  put "quit";
  put "exit";
run;

* FTP files;
data _null_ ;
  length PSFTP Command SysCmd $1000 ;

  PSFTP= "&PSFTPloc.";

  Command = '"' || strip(PSFTP) || '"' || ' USERID@SERVER. -pw PASSWORD -b BAT';

  SysCmd = strip(command);

  SysCmd = tranwrd(SysCmd, "USERID", "&username");
  SysCmd = tranwrd(SysCmd, "SERVER", "&SERVER");
  SysCmd = tranwrd(SysCmd, "BAT", "u:\&BAT..scr");
  SysCmd = tranwrd(SysCmd, "PASSWORD", "&password");

  put SysCmd= ;

* VERIFY PSFTP Success and report out;
rc=system( SysCmd) ;
put rc;

  if rc = 0 then Posting = datetime() ;
  else Posting = .F; *<---means file failed;

  put Posting= datetime20. ;

run ;

```

After obtaining the secured credentials, the destination, server connection name, and source file are assigned. Then the commands sent to PuTTY are assigned to a batch file. We gave the file a .scr extension to delineate it from the

scheduled transfer batch file, whose name is assigned to a macro variable. Note the use of the CD command that navigates to the correct upload destination and the PUT command that uploads the 'Dummy123.txt' file. This data step could easily be modified to download a file, using a GET command, or to conditionally execute an upload/download based on other information. In the last step, the transfer is run. Note the main command sent to PuTTY is: '<PuTTY executable path> USER@SERVER. -pw <password> -b <batch /script file>'. The code replaces the key words in this command with the transfer-specific parameters.

The last data step (which could be modified to dynamically create a batch/script file, based upon time or a separate dataset) creates the batch/script file PSFTP_bat.scr

```
1      cd "/abc/tfil01/client/773/eloaders/elstudy/oclp9"
2      put U:\Dummy123.TXT
3      quit
4      exit
```

Note that the REPUT and REGET commands could be included if the system return code is a failure, for example by including a condition on RC greater than zero. If it is greater than zero, a separate batch/script file could be created with REPUT (or REGET for downloads) in place of PUT (or GET) in the original batch/script file. It is not included here so that the administrator of the job has to manually rerun if the transfer fails.

By combining Scheduled Tasks, the ability of SAS to issue commands to the windows command line and manipulate external files, and the use of simple PSFTP commands, a multitude of different automated secure transfers can be programmed.

CONCLUSION

Any endeavors involving the repetitive movement of data across systems on a scheduled or frequent basis should involve automation to mitigate risk of error. Using SAS to facilitate the automation is ideal when the software is already being used for file creation or using the files for analysis. We demonstrated in this paper that SAS interacts well with PSFTP for the movement of files.

REFERENCES

SAS OnlineDoc®

"Let the system do the work! Automate your SAS code execution on UNIX and Windows platforms", N. Pandya and V. Paidia, <http://www.lexjansen.com/pharmasug/2011/AD/PharmaSUG-2011-AD11.pdf>

Chapter 6: Using PSFTP to transfer files securely via
<http://the.earth.li/~sgtatham/putty/0.52/html/doc/Chapter6.html#6.2>

The PSFTP executable can be downloaded from: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

ACKNOWLEDGMENTS

The authors would like to thank Palpasa Manandhar for her careful review of this paper.

DISCLAIMER

The content of this paper are the works of the authors and do not necessarily represent the opinions, recommendations, or practices of PPD.

CONTACT INFORMATION

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

Kyle Thompson
PPD
3900 Paramount Parkway
Morrisville, NC 27520
Work Phone: +1 919 456 6498
E-mail: kyle.thomspon@ppdi.com

Ken Borowiak
PPD
3900 Paramount Parkway
Morrisville, NC 27520
Work Phone: +1 919 456 5373
E-mail: ken.borowiak@ppdi.com

Other brand and product names are trademarks of their respective companies.