PharmaSUG 2015 - Paper QT31

Copying Files Programmatically in SAS® Drug Development (SDD)

Wayne Woo, Novartis Vaccines, Cambridge, MA

ABSTRACT

SAS® Drug Development (SDD) is a statistical computing environment and repository that provides a 21 CFR Part 11 compliant system for analysis and reporting of clinical trials data. Sometimes, the tradeoff to implement a rigorous controlled system involves sacrificing features that have become commonplace in the toolkit of programmers working in other systems. One such feature unavailable in SDD is the use of the X command, thus forbidding use of the operating system's file copying command via %SYSEXEC and the FILENAME PIPE device. This led the author to search for workarounds to programmatically copy non-SAS-dataset files within SDD. This paper explores coding techniques that can be used to programmatically copy files within SDD version 3.5.

INTRODUCTION

Almost all SAS programmers know that searching the internet provides access to a wealth of information and ideas for the productive use of SAS. Upon realizing that the material in this paper was not covered adequately in existing web resources, the author wishes to make this contribution to the public forum.

This paper assumes the reader is a user familiar with SDD version 3.5. SDD 3.5 uses SAS 9.2 at its core. The coding techniques pertain to non-SAS files within SDD because PROC COPY and PROC DATASETS work normally within SDD. This means they can handle "SAS files" (e.g. datasets, catalogs, views, item stores, etc.) but not files such as SAS program files (i.e. ".sas" extension), nor log and listing files. Some other examples of non-SAS files include Microsoft Word or Excel documents, Adobe PDF documents, Notepad (.txt) files, etc.

The motivation for exploring coding techniques is two-fold:

- a need to perform the copy operation programmatically rather than manually through the SDD interface; and
- the SDD macro %SASDRUGDEV_COPYFILE does not work when the destination file is existing

TECHNIQUE #1: USING SDD MACROS

SDD comes loaded with a set of macros that facilitate programmatic interfacing with the SDD system, primarily for administration purposes. The SDD macros use a syntax like that of the SAS macro system in BASE SAS. The operations performed by the macros are subject to the permissions of the user calling the macro. Each macro call returns a code in the global macro variable _SDDRC_. The first technique uses the following SDD macros:

SDD Macro	Purpose
%SASDRUGDEV_LOGIN	Login information to provide the permission context
%SASDRUGDEV_FILEEXISTS	Check if file exists
%SASDRUGDEV_CREATELOCALFILE	Create a temporary file
%SASDRUGDEV_UPDATESDDFILE	Copy a file over an existing file
%SASDRUGDEV_COPYFILE	Copy a file to a location where file does not exist
%SASDRUGDEV_LOGOUT	Logout at end of macro operation

The source code:

```
%SASDRUGDEV_LOGIN (URL for SDD instance,sdduserid,sddpassword);
%LET localpath=%SYSFUNC(PATHNAME(WORK));
%SASDRUGDEV_CREATELOCALFILE(LOCALPATH=&localpath./file,
   SDDPATH=&sourcepath./file);
%SASDRUGDEV_FILEEXISTS(SDDPATH=&destpath./file);
%MACRO doit;
   %IF &_SDDRC_ = 1 %THEN %DO;
      %SASDRUGDEV_UPDATESDDFILE(LOCALPATH=&localpath./file,
         SDDPATH=&destpath./file);
   %END;
   %ELSE %DO;
```

```
10        %SASDRUGDEV_COPYFILE(SRCPATH=&sourcepath./file,DESTPATH=&destpath./file);
11     %END;
12  %MEND doit;
13  %doit ;
14  %SASDRUGDEV_LOGOUT();
```

EXPLANATION

Any place in the code where “file” appears represents the name of the non-SAS file, e.g. xyz.pdf.

Lines 01 and 14 log into and out of SDD to provide the permission context under which the SDD macros run.

Line 02 determines a temporary location on the SAS execution server where we can write a temporary copy of the file

Line 03 copies the file from the SDD location (&sourcepath) to the temporary location (&localpath)

Line 04 checks if the file exists at the destination (&destpath)

Lines 06 and 07 check if the destination file exists (&_SDDRC_=1) and if yes, uses %SASDRUGDEV_UPDATESDDFILE to copy the file because %SASDRUGDEV_COPYFILE does not work if the file exists at the destination (either versioned or not versioned)

Line 10 uses %SASDRUGDEV_COPYFILE because %SASDRUGDEV_UPDATESDDFILE does not work if the file does not exist at the destination.

TECHNIQUE #2: USING RECFM=N

The RECFM=N option for the INFILE and FILE statements allows for reading and writing files at the binary byte level. SDD version 3.5 runs SAS 9.2 on a Unix-based instance, so the author verified that RECFM=N is an available option for this operating system configuration. The source code below reads the source file byte-by-byte and writes each byte to the destination file.

```
01  FILENAME srcpath "&sourcepath";
02  FILENAME outpath "&destpath";
03  DATA _null_;
04     FILE  outpath(file) RECFM=N;
05     INFILE srcpath(file) RECFM=N;
06     INPUT c $CHAR1.;
07     PUT c $CHAR1. @@;
08  RUN;
```

EXPLANATION

Any place in the code where “file” appears represents the name of the non-SAS file, e.g. xyz.pdf.

Lines 01 and 02 define parameters &sourcepath and &destpath as folders.

Lines 04 and 05 specify the file name, without quotation marks, using the aggregate storage location coding convention.

Line 06 reads one byte at a time. The $CHAR1 informat reads the contents as-is.

Line 07 writes one byte at a time, using the @@ pointer control to prevent insertion of a line break as the DATA step is repeated for the entire contents of the file.

FINE POINTS

Regardless of the technique used, the destination file will have a new “modified” datetime stamp consistent with a manual copy/paste operation.   The SDD audit trail for the destination file will also show that either a new file was created or an existing file updated, again consistent with a manual copy/paste operation.

As a note, the author did not investigate other techniques, particularly the set of F* functions (i.e. FOPEN, FGET, FREAD, FPUT, FCLOSE).   In SAS 9.4, a function FCOPY is made available, but with SDD, we need to be mindful of both the SDD version and the BASE SAS version that is running underneath.

CONCLUSION

SDD is a robust system for implementing a controlled statistical computing environment and data repository. However, there are tradeoffs and sacrifices for features that are well known to SAS coders using traditional operating system-based platforms. Luckily, there are usually workarounds and clever techniques to compensate and replace the functionality.

REFERENCES

SAS Institute Inc. 2012. SAS ® Drug Development 3.5: Macros User's Guide. (Second printing) Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The author thanks Ben Bocchicchio for his advice on implementing the techniques presented.

CONTACT INFORMATION

Your comments and questions are welcome. Contact the author at:

Wayne Woo
Cambridge, MA 02139
E-mail: wayne.f.woo@gsk.com