Paper 049-29

Switch Data Source and Output Destination without Changing Programs

David Shen, ClinForce Consulting, Philadelphia, PA
Zaizai Lu, AstraZeneca Pharmaceutical, Wilmington, DE

ABSTRACT

Two major issues have to be resolved before the programs developed in test environment can run properly in production environment: 1) new data source, 2) locations to save output and log files, should be re-defined for production purpose. Usually all the programs need to be modified for libnames and filenames even if they have been quality checked in test environment. Paper work should be done for tracking purpose. This paper provides some strategies on how to intelligently switch data library and output destination without any changes to the programs. A macro called %Runner is presented here. This macro acts as a program-running platform: (1) It allows users to redirect data sources, change output filenames, locations, and ODS destinations without modifying a piece of the original programs. (2) It checks the locations to save output files. The new file locations can be automatically created by the macro if they are not available at run time. (3) It can run a batch of programs in the specified program location, the macro can search and run programs, save their output, and logs to individual files. Therefore, once programs are migrated from test to production, it's not necessary to change any code for these programs. And undoubtedly, %Runner has wider applications than mere file migration.

INTRODUCTION

"Reusability, Extendability, Maintainability, and Reliability …,the greatest of theses is Reliability.",
Dr. LeRoy Bessler indicated in the SUGI26 paper "Strong Smart Systems: Software-Intelligent Development for Reliable, Reusable, Extendable, Maintainable Applications", "The key to reliability is simple: once your program is working right, never touch it again. The only safe program change is no change". The design of the macro %Runner allows file management parameters to be stored in a separate file. Data sources, output file names and their saving locations and ODS destinations can be passed to the programs by macro variables without any modifications to the original programs. When a program works properly in test environment, it should also work well in production environment. With %Runner, it is not necessary to adjust any code during the migration. %Runner is flexible and powerful, too. It can run all the programs or a specified number of programs in one location. It creates file destination locations automatically when it finds that they don't exist. No more human supervision is needed.

MANAGE DATA AND FILES DYNAMICALLY

When programs developed in test environment are migrated to a stable production environment, user intervention usually is required to:

- Switch data source library, i.e. from test data to production data.
- Change output file name and location, e.g. from s:\project\study\test\output\draft rpt.rtf to s:\project\study\prod\output\final rpt.rtf.

In addition, the file-saving locations should be manually checked and created if they do not exist before the programs run. The macro %Runner can dynamically assign the programs with new data source library and output file names, and destination paths when necessary, without any changes to the original programs. With a specified program location, %Runner can run all or a specified number of programs automatically. Here is the macro %Runner, with some explanatory notes at the end, which explain in detail each of the numbers in square brackets throughout the macro:

```
%MACRO RUNNER(_sasloc=,
              _lstloc=,
              _logloc=,
               datapath=,
               csvfile=,
               csvonly=);

options noxwait;

%if &_lstloc eq  %then %do;[1]      *if _lstloc null;
  %let _lstloc=&_sasloc;            *take _sasloc value;
%end;

%if &_logloc eq %then %do;          *if _logloc null;
  %let _logloc=&_lstloc;            *take _lstloc value;
%end;

%let Progn=;                        *set # of prog null;

data saslst (keep=sasprog);[2]      *list sas® programs;
  rc=filename('prog', "&_sasloc");  *define filename;
  dsid=dopen('prog');               *try to open sas dir;
  if dsid then do;                  *if dir exist;
    n=dnum(dsid);                   *get # of files;
    do i=1 to n ;                   *collect file names;
       sasprog=dread(dsid, i);
       sasprog=upcase(trim(left(sasprog)));
       if scan(sasprog, 2, '.')='SAS'
          then output;              *sas programs only;
    end;
  end;
  else do;                          *if dir not exist;
    put / "*-SAS Program Location Does not Exist.-* ";
  end;
  rc=dclose(dsid);                  *close dir;
run;

data saslst;
  length sasprog $100;
  set saslst;
  outfile=' ';
run;

proc sort data=saslst;
  by sasprog;
run;

%if &csvfile ne  %then %do;  [3]    *csvfile assigned;
proc import datafile="&csvfile"
     out=sascsv dbms=csv replace;  *read data from csv;
     getnames=yes;                 *get column name;
     datarow=2;                    *data from 2nd row;
run;
```

```
data sascsv;
 length sasprog $100;
 set sascsv;
 sasprog=upcase(trim(left(sasprog)));
run;

proc sort data=sascsv;
  by sasprog;
run;

data saslst;
  length outfile $500;
  merge saslst (in=in_lst)
        sascsv (in=in_csv);
  by sasprog;
  %if %upcase(&csvonly)=YES %then %do; [4]
    if  in_lst and in_csv;
  %end;
  %else %do;
     if in_lst;
  %end;
run;
%end;

data _null_;               *program names to macro vars;
  set saslst end=last;
  call symput ('_Prog'||left(_N_), sasprog);
  call symput ('_outf'||left(_N_), outfile);
  if last then call symput('Progn', trim(left(_N_)));
run;

%if &progn ne %then %do;
data _null_;        [5]                   *check lst dir;
  rc=filename('lst', "&_lstloc");
  dsid=dopen('lst');
  if not dsid then do;
     call system("mkdir &_lstloc");
     put "*-New LST Dir &_lstloc Has been Created.";
  end;
  rc=dclose(dsid);

 rc=filename('log', "&_logloc");       *check log dir;
  dsid=dopen('log');
  if not dsid then do;
     call system("mkdir &_logloc");
     put "*-New LOG Dir &_logloc Has been Created.";
  end;
   rc=dclose(dsid);
run;
```

```
%do Runner=1 %to &progn;
%put *============================================*;
%put *== RUNNER=&runner    PROGRAM=&&_prog&runner;
%put *============================================*;
%let loglst=%scan(&&_prog&runner, 1, '.');
filename logname "&_logloc\&loglst..log";
filename lstname "&_lstloc\&loglst..lst";

%let filepath=&&_outf&runner; [6]
%if &filepath ne %then %do;
    %let  path=%substr(&filepath, 1, %length(&filepath)
          -%length(%scan(&filepath, -1, '\'))-1 );
    %put ---Path To Save File=&path ---;
    filename path "&path";

   %if %sysfunc(fileref(path))=0 %then %do;
       %put ---Output File Location Existed---;
   %end;
   %else %do;
       %sysexec "mkdir &path";
       %put ---New &path Created To Save File---;
   %end;
filename filepath "&filepath"; [7]
%end;

%if &datapath ne %then %do;
libname datapath "&datapath";
%end;

%let sasprog=&_sasloc\&&_prog&runner;

proc printto log=logname   new; [8]      *record log;
proc printto print=lstname new;         *record lst;

options nosymbolgen nomacrogen nomprint nomlogic; [9]


%inc "&sasprog";
run;
proc printto log=log; run;  [10]
proc printto print=print;
run;
proc datasets lib=work kill nowarn nolist;
quit;

libname datapath clear;
filename filepath clear;
%end;
%end;

%else %do;
  %put *-------------------------------------*;
  %put *--No SAS Program Found in &_SASLOC--*;
  %put *-------------------------------------*;
%end;

%mend;
```

Explanatory Notes on %Runner:

[1] Usually, lst and log files take program's name with different extensions. If the program's name is abc123.sas, then its lst and log file name will be abc123.lst and abc123.log, separately. There are 3 possible ways to store these files:
Case 1: store program, lst and log files to the same program SAS folder.
Case 2: put program in SAS folder, and both lst and log files to a separate folder LSTLOG
Case 3: put program in SAS folder, lst in LST folder, and log in LOG folder.

%Runner provides these possibilities. If only _sasloc is assigned and leave _lstloc and logloc null, then _lstloc will take the value of _sasloc, and _logloc will take the value of _lstloc subsequently. Thus ,lst and log will be sent to the same program folder. If _sasloc and _lstloc are assigned, then _logloc will take the value of _lstloc, and save log to lst folder. If all these three parameters are assigned, then program, lst and log will be in 3 different locations.

[2] Check the program location, then scan and list all the SAS programs. If the specified location doesn't exist, then a message will be written to log to indicate the location is not available and quit.

[3] When a csv file is not assigned, it will run without any changes on filings. Otherwise, the csv file will be read into a sas dataset. The output file requirements will be passed to the specified running program by %runner. The output file name can be changed and its location will be redirected automatically.

[4] Two choices are provided to run either all programs in the specified location or just run partial programs. If csvonly=yes, then only the programs appeared on the csv file will be run.

[5] If no program found in the location, then it will send a message to log and quit. Otherwise, it will check lst, log locations, and create them if they don't exist yet. This can save time and trouble to manually check and create these file locations before run the programs.

[6] If outfile column doesn't contain the information for the program, then program will use its original information to manage output files. For example, if there are 10 program abc01.sas-abc10.sas in one location and csv file contains:

SASPROG	OUTFILE
abc01	s:\study\prod\finalrpt01.rtf
abc02	s:\study\prod\finalrpt02.rtf
abc03	
abc05	s:\study\prod\finalrpt05.rtf

If csvonly=yes, then only 4 programs will be run. Program abc03 output file will still be s:\study\test\draftrpt03.rtf. The other 3 output files would be sent to production with their new names. If csvonly is not checked, then all 10 programs will be run, but only 3 programs change their output file destinations as specified.

[7] Define filename for filepath, or\and libname for datapath if the values of their macro variables have been assigned.

[8] Redirect lst and log output to their individual files from Windows, and save them in the specified locations.

[9] Set up options and run the programs.

[10] Restore the system, remove any intermediate datasets, and dessign filename and libname.

How to Use %Runner:

To establish the communications between %Runner and the programs to be run, the libname and filename statements in the programs should be included in a macro called %SetInOut. The default running environment of the programs is set to test. The programs can also run without calling %Runner. But not using %Runner brings about lack of automation and efficiency, as we have mentioned throughout the paper. With %Runner, the user only needs to change the values of the parameters in the macro %Runner, and gets all the benefits %Runner offers.

```
%macro SetInOut;
%let rc=%sysfunc(libref(datapath));
%if  &rc  %then %do;
libname datapath 's:\study\test\data';
%end;

%let rc=%sysfunc(fileref(filepath));
%if &rc>0 %then %do;
filename filepath 's:\study\test\draft.rtf';
%end;
%mend;

%SetInOut;
```

If %Runner has user-defined datapath or/and filepath, the programs will accept them, otherwise, %Runner uses the default information contained in the programs, i.e. the test environment.

Here is an example of how to use the macro %Runner & direct the data source, output/log locations to the production area:

```
%Runner (_sasloc  =s:\study\prod\sas,
         _lstloc  =,
         _logloc  =,
         datapath =s:\study\prod\data,
         csvfile  =s:\stusy\prod\SasOut.csv,
         csvonly  =);
```

CONCLUSION

When SAS programs are being developed and tested, data library is assigned to test data. Programmers would like to see the output and log in Windows for easy and instant checking. However, to run these programs in production environment, the users should change the libname and filename statements or ODS statements in the programs. This paper provides an application to deal with the data source, file management dynamically. The macro %Runner makes it possible for the program to run in production without any code changing.

TRADEMARK INFORMATION

CONTACT INFORMATION:

Zaizai Lu
AstraZeneca Pharmaceutical
Wilmington, Delaware
Zz_lu@hotmail.com

David Shen
ClinForce Consulting
Philadelphia, PA
Shenda168@hotmail.com