Paper 81-27

Macro for Restoring SAS® Transport Files

Yefim Gershteyn, Takeda Pharmaceuticals North America, Inc., Lincolnshire, IL

ABSTRACT

SAS provides two basic ways to create a SAS transport file: using either the CPORT, or COPY procedures. Accordingly, PROC CIMPORT and PROC COPY are used to restore the file on the target host. Without prior knowledge of which procedure was used to create the transport file, the programmer may apply the wrong procedure to restore it, which will result in an error message and the necessity of re-writing the program. Also, if PROC COPY was used, format catalogs can only be transported in the form of SAS data sets. In this case, an additional piece of code is needed to create a respective format catalog on the target host.

A macro is presented, which restores a SAS transport file without prior knowledge of how this file was created. If the macro detects that PROC COPY was used, it attempts to create a format catalog, if a data set with format descriptions exists in the restored data files. The presentation concerns SAS base under Windows NT or UNIX and is intended for intermediate and advanced SAS users.

INTRODUCTION

SAS provides three ways to create a SAS transport file – with PROC CPORT, PROC COPY, or in the DATA step using XPORT engine in the LIBNAME statement. The third way is limited in scope, while the first two are the most common. If PROC CPORT was used to create the transport file, then PROC CIMPORT needs to be applied to restore the file on the target host. If PROC COPY created the file, then the same procedure PROC COPY is used to restore it. While PROC CPORT/PROC CIMPORT procedures can handle both SAS data sets and format catalogs, PROC COPY cannot deal with format catalogs. In this case, in order to include format specifications in the transport file, a SAS data set is created containing the information on the formats in the catalog.

Without prior knowledge of which procedure was used to create the transport file, the programmer may apply the wrong procedure to restore it, which will result in an error message (e.g. “ERROR: File is probably a cport file. XPORT engine unable to read file created by proc cport. Please use proc cimport to convert this file to native format.”) and the necessity of re-writing the program. Although not that time-consuming, substituting the code can be irritating. The problem becomes more serious if the program runs in the batch mode and its output is used in the consequent programs. Obviously, some precautions can be made in case of an error (e.g. triggering the other procedure if the error was detected). However, addressing the possible lack of a-priori knowledge about the procedure which was used to create the transport file is a better solution.

Also, if PROC COPY was used, there is a chance that the transport file contains a data set(s) with the information about formats, created by PROC FORMAT. In this case, an additional piece of code is needed to create a respective format catalog on the target host. It is appealing to automate this task as well.

A description of a macro that performs the above mentioned tasks follows.

%IMPORT_XPT MACRO: SUMMARY

The %IMPORT_XPT macro restores a SAS transport file without prior knowledge of how this file was created. If the macro detects that PROC COPY was used, it attempts to create a format catalog, provided that a data set with format descriptions exists in the restored data files.

The macro works in the following sequence:

- Detects which procedure was used to create the transport file. This is done by reading in the first line of the transport file
- If it detects that PROC COPY was used, it
  - Restores all SAS data sets using PROC COPY
  - Searches for a data set containing the string “FORMAT” in its name
    - If it finds such a data set, it checks whether the data set holds variables FMTNAME, START, LABEL, which are the required variables in a data set describing formats
      - If these variables are present, it creates a format catalog from the data set
- If it detects that PROC CPORT was used, it restores the data sets and catalogs in the transport file.

%IMPORT_XPT MACRO: DETAILED DESCRIPTION AND SOURCE CODE

```
**********************************************
*********
MACRO %IMPORT_XPT RESTORES FILES IN
TRANSPORT FILES INTO SAS DATA SETS AND FORMAT
CATALOGS.

Macro Parameters:
   FILEIN - physical address of transport file, e.g.
   J:\RECEIVE_DIRECTORY\MYXPTFILE.XPT
   LIBOUT - physical location of restored data sets and
   catalogs, e.g. J:\OUTPUT_DIRECTORY
NOTE: If PROC COPY was used, there is a possibility of
      having a  data set with formats among the restored  data
      sets. The macro checks if the data set(s) named
      FORMATXXX exists. If it exists, the macro checks
      further if the three required variables are in the data set.
      If so, the macro creates a format catalog.
**********************************************
**********;
```

```
%MACRO import_xpt(filein=, libout=);
  ❶%global whichproc;
  ❷
*********************************************;
**Check which procedure was used to create the transport file.
According to SAS Institute, if the XPORT engine created the
transport file, the beginning of the file contains this text:
HEADER RECORD*******LIBRARY HEADER
RECORD!!!!!!!00
If  PROC CPORT created the transport file, the beginning of
that file contains this text:
**COMPRESSED** **COMPRESSED****COMPRESSED**
**COM
If NOCOMPRESS option in PROC CPORT was set,
compression is suppressed, which prevents the display of the
preceding text. In this case the file begins with the text:
LIB CONTROL                                                ****;
*************************************************;
**SET A FILEREF TO READ IN THE FIRST LINE OF
**THE TRANSPORT FILE **;
❸filename findhow "&filein";
  data _null_;
    length line $1;
    infile findhow obs=1;
         input line;
         if upcase(line) ='H' then call symput('whichproc',
                          'FROMCOPY');
         else call symput('whichproc', 'FROMCPORT');
   run;

   filename findhow clear;

❹**IF XPORT ENGINE (PROC COPY) WAS USED**;
  %if &whichproc=FROMCOPY %then %do;
****SET UP LIBREFS  *;
     libname indata xport "&filein";
     libname library  "&libout";
****CREATE FILES *;
      proc copy in=indata out =library;
      run;
❺**CREATING A FORMAT CATALOG FROM A DATA
      SET(S) NAMED FORMATxxx  **;
****CHECK WHETHER DATASET(S) NAMED
       FORMATxxx EXIST**;
  ❻ proc sql;
     create table names as
         select memname
         from dictionary.members
         where libname='LIBRARY';
     quit;

❼    data names;
        set names;
         if index(upcase(memname), "FORMAT");
         counter=_n_;
         varname = "mem"||left(put(counter, 3.));
        call symput(varname, memname);
      run;
❽**CHECK WHETHER THE DATA SET IS NOT EMPTY
     AND DEPENDING ON THAT PROCEED **;
****HOW MANY DATA SETS **;
      %let dsid=%sysfunc(open(names));
            %if &dsid %then %do;
                  %let nobs=%sysfunc(attrn(&dsid,NOBS));
                  %let rc = %sysfunc(close(&dsid));
            %end;
❾**CHECK IF REQUIRED VARIABLE NAMES EXIST **;
  **REQUIRED VARIABLE NAMES ARE FMTNAME,
     START, LABEL **;
        %if &nobs %then %do;
          %do i=1 %to &nobs;
               proc contents data=library.&&mem&i
                     noprint out=&&mem&i;

               data &&mem&i(drop=memname name);
                  set &&mem&i(keep=name memname) end=eof;
                   retain datvar 0;
                   if upcase(name) in ('FMTNAME', 'START',
                      'LABEL') then  datvar + 1;
                   if eof then do;
                         if datvar ge 3 then
                            call symput("formats&i", "YES");
                         else call symput("formats&i", "NO");
                   end;
                run;
❿          %if formats&i = "YES" %then %do;
                     proc format library=library
                                  cntlin=&&mem&i;
           %put NOTE: FORMAT CATALOG WAS CREATED
                    FROM DATA SET &&mem&i;
            %end;
            %if formats&i = "NO" %then %do;
              %put NOTE: FORMAT CATALOG CANNOT BE
                 CREATED FROM DATA SET &&mem&i;
              %put AS THE DATA SET DOES NOT CONTAIN
                 REQUIRED VARIABLE NAMES;
               %put FMTNAME, START, LABEL. PLEASE
                 CHECK THE DATA SET;
            %end;
        %end;
     %end;

     %else %do;
           %put NOTE: NO DATA SETS FOUND NAMED
                 FORMAT OR SIMILAR;
      %end;
          libname indata clear;
          libname library clear;
          proc datasets;
            delete names;
          quit;
          run;
  %end;
❶❶**ASSUMING PROC CPORT WAS USED **;
  %if &whichproc=FROMCPORT %then %do;
****SET UP LIBREFS  **;
          filename indata  "&filein";
           libname outdata  "&libout";
****CREATE FILES **;
           proc cimport infile=indata library =outdata;
           run;
                  filename indata clear;
           libname outdata clear;
           run;
  %end;
*********************************************;
%mend import_xpt;
```

Notes:

❶ Define global macro variable &WHICHPROC holding the name of the SAS procedure that created the transport file.

❷ Identify which SAS procedure created the transport file:

❸ read in the first character in the first line of the transport file. If the XPORT engine created the transport file, the beginning of the file contains this text: *HEADER RECORD*******LIBRARY HEADER RECORD!!!!!!!00*, and the first character should be 'H'. Otherwise, the transport file was created by PROC CPORT.

❹ If it is identified that the XPORT engine was used (most frequently, with PROC COPY), then use PROC COPY to restore the file. The XPORT engine can also be used in a DATA step to create a transport file containing only one data set. In this case, the macro should work as well, since the transport file still starts with the letter 'H'.

❺ Since a transport file created by PROC COPY cannot contain catalogs, but could contain a data set(s) with format descriptions, the macro attempts to find such a data set in the restored files and create a format catalog from this data set.

❻ obtain data set names in the library.

❼ subset and count those names having the string "FORMAT" - in an assumption that if there is a data set(s) with format descriptions, it would contain at least this string in its name. Also create macro variables holding name(s) of such data sets.

❽ check if the data set NAMES has any observations, e.g. if there are any restored data sets with names containing the string "FORMAT", and if there are observations, pass the number of observations to the macro variable &NOBS.

❾ for each of the data sets having the string "FORMAT" in their names, check if it contains three variables that are required in order to create a format catalog from the data set. The required variables are FMTNAME, START, LABEL.

❿ for each data set having all three variables, create a format catalog.

❶❶ If it is identified that PROC CPORT was used to create the transport file, invoke PROC CIMPORT to restore the files.

CONCLUSIONS

Using SAS specifications for transport files, the process of restoring these files on the target hosts can be automated so no prior knowledge of how the transport file was created is needed. In restoring the files, it is also possible to create format catalogs from respective data sets, if it is found that the PROC COPY procedure was used to create the transport file. The macro performing these tasks can save time and helps to avoid error messages. It is especially useful when restoring of the transport files is done in the batch mode.

CONTACT INFORMATION

Yefim 'Fima' Gershteyn, Ph.D.
Takeda Pharmaceuticals North America, Inc.
475 Half Day Rd., Suite 500
Lincolnshire, IL 60069
Phone: (847) 383-3425
E-mail: fgershteyn@takedapharm.com

TRADEMARKS