ABSTRACT

Occasionally, SAS professionals are asked to supply our customers with Excel versions of one or more of the datasets. Typically, users have no, nor should be expected to have, knowledge of the underlying data’s structure such as what a variable’s short name means with respect to the contents of a column.

Though there are a number of SAS tools to provide a user community direct access to these datasets, like SAS®’s Add-In for Microsoft Office, one tried and true method is to export these datasets to Excel using SAS’s Proc Export.

A few limitations of this method are that this can be a fairly manual process (generally one dataset at a time) and, maybe more importantly, Proc Export uses variable names as column headers.

If, as this author encountered, when there are dozens to scores of datasets with, cumulatively, hundreds of variables this task can become daunting. Combine that with the need to export a number of data libraries to Excel and the task screams for a more productive approach.

This paper presents an all-purpose macro that: dynamically reads a data library, grabs each dataset in turn (including the variable’s labels) and inserts each dataset into its own worksheet in a Microsoft Excel spreadsheet using the descriptive label rather than the variable’s name.

BACKGROUND

A hurried question: “Do you have a quick way to export a three or four libraries to Excel so that non-technical people in…. medical affairs, legal, …, ______ (you fill in the blank) can review all (50, 60, ???) of the datasets? Also, I’m going to need to do this on 3 or 4 directories so, doing this manually is not an option!”

My first response was, “What about SAS® Add-In for Microsoft Office?” and, judging by the look I got, I immediately concluded that that didn’t exist at this site. My next response was, “Well, what about PROC EXPORT?” Predictably, the response was that “They need a description of the column’s content not some obscure SAS® name… any other ideas? And, oh yeah, I just can’t send them 60 different Excel spreadsheets so, we really need combine these into one spreadsheet!”

So, labels in the column headings… each library has a separate spreadsheet… each dataset in a separate (named) tabbed worksheet and we have 3-4 hours to do this (that is, it must be easy to do because it has to happen REALLY FAST!)

Piece of cake, right?

Actually yes! But, not quite as light (easy) as an Angel’s Food Cake but not as heavy, so-to-speak, as a Fruitcake; we came close to meeting our goal of exporting these directories to spreadsheets but we encountered interesting challenges that made for an interesting solution.

UNDERSTANDING THE TASK and CHALLENGES

THE TOOLS

Even the simplest of programming tasks requires knowledge of the tools involved, their functionality and limitations, as well as the user’s skills and knowledge. So, a brief discussion of what went on both between the requestor and the developer and within the developer’s head as this development rapidly progressed.

SAS® stores a library of datasets in discrete files in a file location and Excel stores a ‘dataset’ in a separate worksheet within a workbook. SAS® allows a very large number of: datasets, records and variables compared to Microsoft Excel’s limitations of 64K records for rows (records) and 256 columns (variables.) The size of the SAS file is limited only by the file system used; Excel on the other hand uses memory to display a workbook and memory size affects processing and viewing speed.

Though information is sketchy on this, it is reasonable to expect that there is a limit on the number of worksheets (read datasets) that can be placed in any workbook but experience shows that a workbook can handle over 70 worksheets at a time so the maximum number of worksheets is probably (at a minimum) 256, though this has not been tested.

Variable names in SAS have restrictions that column labels in Excel don’t share and, generally, SAS variable names are not meaningful to end-users, though the variable labels are closer to the mark. The same applies to the SAS dataset names; useful to the technical personnel but, not so much to the general user community yet the SAS dataset labels are more useful to the
non-technical personnel. Additionally, SAS® variable labels can be longer than the 32 character limits of the variable names so, care must be taken when using variable labels rather than names or the export process must eliminate SAS’s 32 character variable name limit.

**THE TIME**

Typically, only a few programmers can be assigned a task of this sort, oftentimes only one, so there can not be a lot of manual: file inclusions, cutting and pasting, editing of worksheets, etc. so the process must be efficient with a minimum of manual work.

With only a few datasets to work with a manual process might work but, since we’re talking about a large number of libraries, datasets and variables the manual process quickly becomes unwieldy and was rejected out of hand by the programmer.

**EXPLORING SAS® EXPORT TOOLS**

SAS® has a number of method to perform these export tasks, they include:

1) Proc Export to Excel
2) Excel Libname Engine
3) SAS® Add-In to Microsoft Office

Each has advantages, and limitations, and the first two are methods are discussed in the table below1.

<table>
<thead>
<tr>
<th>Idea</th>
<th>Method</th>
<th>Base Code</th>
<th>Limitations</th>
<th>Additional Macro Needs</th>
</tr>
</thead>
</table>
| 1    | Proc Export (to Excel) | PROC EXPORT  
data=<dataset_name#1>  
outfile=<out_file_name>  
DBMS=EXCEL  
REPLACE  
sheet="<dataset_name#1>" ;  
run;  
... more export statements...  
PROC EXPORT  
data=<dataset_name#2>  
outfile=<out_file_name>  
DBMS=EXCEL  
REPLACE  
sheet="<dataset_name#2>" ;  
run; | 1) Does NOT export variable labels  
2) Dataset names not descriptive enough  
3) Requires enumeration of each dataset  
4) Variable labels are not valid variable names  
5) SAS’s 32 character variable naming convention will not allow any LONG labels to be used as variable names. | 1) Method to loop through data list  
2) Method to use dataset’s label rather than dataset’s name  
3) SAS’s options (validvarname=any) must be used  
4) Method to use variable’s label instead of variable’s name |
| 2    | Excel Libname Engine (Proc COPY) | libname xPort  "<My.Dir>" ;  
libname myxls  "<myfile.xls>";  
proc datasets lib=xPort  
mt=data ;  
copy out=myxls ;  
exclude <DSs_2_Exclude> ;  
(include < DSs_2_Include> ;)  
run ;  
quit ;  
libname myxls clear; | 6) Does NOT export variable labels  
7) Dataset names not descriptive enough | 5) Method to loop through data list  
6) Method to use dataset label rather than dataset name  
7) Method to use variable’s label instead of variable’s name |
| 3    | Excel Libname Engine (Data step) | libname xPort  "<My.Dir>" ;  
libname myxls  "<myfile.xls>";  
data myxls.firstShoes  
dblabel=yes;  
set sashelp.shoes;  
run;  
... more dataset statements...  
data myxls.LastShoes  
dblabel=yes;  
set sashelp.shoes;  
run;  
libname myxls clear; | 8) Must use the <DBLABEL> option – a small price!  
9) Dataset names not descriptive enough  
10) Requires enumeration of each dataset | 8) Method to loop through data list  
9) Method to use dataset’s label instead of dataset’s name |

To summarize the advantages and limitations in the above table -- All three methods of exporting datasets or entire SAS® library contents to Microsoft Excel will encounter SOME undesirable processing. Additionally, all three methods must deal with the differences in the numbers of variables and records allowed between SAS® and MS-Excel.

---

1 Number 3, SAS Add-In to Microsoft Office, is not discussed due to the generally limited availability of this product in most user sites. Another alternative, export to CSV, is also not discussed because it is in the same general class as the first two as well as the limitations associated with importing a large number of these files into MS-Excel.
MACRO SPECIFICATIONS

INPUTS: Library name (previously defined in code)
OUTPUTS: 1) Named, Excel spreadsheet placed in the same directory location as the datasets. (The spreadsheet’s name will be passed into the macro at macro invocation.)
2) Each dataset in a worksheet labeled with the dataset: 1) label or 2) name (if no dataset label is present)
3) Each column in the worksheet will use the variable’s label (if available) and, if not, the variable’s name

RESOURCES: Each conversion will require only 10 minutes or less to be executed, spreadsheet formatting will be limited to formatting column headings using cross-tabbed, row-selected formatting.

COMMON CONSTRAINTS: The macro must prevent SILENT losses of: datasets, variables and records during the export processing

ALTERNATIVE ANALYSIS

IDEA 1: Proc Export (to Excel)
Status: Implemented

Issue # 1: Method to loop through data list – Good and… may be able to use later!
# 2: Method to use dataset label rather than dataset name

Resolution: Develop a LOOP CONTROL DATASET

```sas
%macro LOOPER ( LOI = WORK );
   proc contents data = &LOI.._all_ out = dsList (keep=memname memlabel name label nobs varnum) noprint ;
   run ;
   proc summary data = dslist noprint nway ;
   var nobs varnum ;
   class memname memlabel ;
   output out = looper ( label = "Control Dataset for Looping Through List of Members"
     drop = _freq_ _type_ )
     max(nobs varnum)=numobs numvars ;
   run ;
   proc sql noprint ;
   select count(distinct(memname)) into :nDatasets from looper ;
   select max(numObs) into :maxNumObs from looper ;
   select max(numvars) into :maxNumVars from looper ;
   quit ;
   / * Determine if the number of SAS records exceed Excel's limit of 65,636 * /
   %if &maxNumObs. Gt 65636 or &maxNumVars. Gt 256 %then %goto %doSomethingNice ;
   /* Start looping if run constraints have been met */
   %do i=1 %to &nDatasets. ;
   %global dsn dsnLbl numobs numvars ;
   data _null_ ;
   set looper ;
   if _n_=&i. ;
   call symput('dsn',strip(memname)) ;
   call symput('dsnLbl',strip(memlabel)) ;
   call symput('numobs',strip(put(numobs,9.))) ;
   call symput('numvars',strip(put(numvars,9.))) ;
   run ;
   < put_your_code_here > ;
   %end ;
   %doSomethingNice:
   %mend LOOPER ;
```
Issue #3: SAS’s options (validvarname=any) must be used
# 4: Method to use variable’s label instead of variable’s name

Resolution: In the <put_your_code_here> section… for each dataset, dynamically create a RENAME statement
something like this:

```sas
/* Create a file segment for inclusion in later steps to rename variables using the name function */
data _null_
    options validvarname=any ;
    set dslist(where=(memname="&dsn.") end = lastone)
    destFile= "sysfunc(pathname(&LOI.))\varRenames_&dsn..seg"
    file dummy filevar=destFile
    if label eq "" then label=propcase(name)
    if _n_=1 then put /*2 " rename " 8 ;
    sPos=20-length(name)
    put @sPos name = "label +(-1) "'n" ;
    if lastOne then put @15 " ;" ;
run ;
```

The code produces output like this:

```
File name: <pathname>\varRenames_NEWSHOES.seg
rename Inventory = 'Total Inventory'n
Product = 'Product'n
Region = 'Region'n
Returns = 'Total Returns'n
Stores = 'Number of Stores'n
Subsidiary = 'Subsidiary'n

NOTE: This only works because of the VALIDVARNAME=ANY option
```

Utilizing the rename code segments in a data step to create temporary datasets the code below finally implements the importation design.

```sas
data &dsn.
    set &LOI.&dsn.
    %inc "sysfunc(pathname(&LOI.))\varRenames_&dsn..seg";
run ;
```

The final, exported Excel Workbook shows (after formatting the headers) the following:

```
Recap -- IDEA #1:
All four of the issues CAN be addressed and the final core PROC EXPORT code show above is relatively clean and detailed programming has been avoided.

NOTE: The REPLACE option shown above allows repeated updates to the worksheet without any need to remove it first!
```
Excel Libname Engine (Proc COPY)

IDEA 2: Excel Libname Engine (Proc COPY)
Resolution: Use the LOOP CONTROL DATASET developed above and controls both the dataset label and the column headers

```sas
/* Create a file segment for inclusion in later steps */
/* to rename variables using the name function */
libname xPort "<My_Dir>";
libname myxls "<myfile.xls>";
proc datasets lib=xPort
  mt=data;
    copy out=myxls;
    run;
quit;
libname myxls clear;
```

Out of the box, this procedure, combined with the Excel libname engine is pretty powerful giving you the following results.

NOTE: Variable NAMES are used rather than LABELS and dataset NAMES are used instead of dataset LABELS but, other than that, the code is a simple and powerful method for the export of all datasets in a SAS® library to a single MS Excel spreadsheet for dissemination to people with a need to access this data.

Excel Libname Engine (Data step)

IDEA 3: Excel Libname Engine (Data step)
Resolution: Use the LOOP CONTROL DATASET developed above and control the dataset label, and let the DBLABEL dataset option control the placement of the variable’s label in the Excel worksheet
Advantage: Long variable labels will be directly placed in the target worksheet with no truncation or special handling.

```sas
/* This data step runs inside of the LOOP CONTROL UTILITY */
libname myxls "%sysfunc(pathname(&LOI.))\&XLSfileName2Use..xls";
data myxls.&dsnLbl.
  (dblabel=yes);
set &LOI.\&dsn.
run;
libname myxls clear; * Clear the libname to release the spreadsheet ;
```

NOTE: If a worksheet of the same name exists in the XLS file, the EXCEL libname will not update the workbook. It is safer to delete the spreadsheet and re-execute the macro.

RUN NOTE: The following NOTE will be, erroneously, written to the log

“NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.”
SAS Technical support has provided the following response:
“I’ve requested that this Note be altered or removed when the DBLABEL data set option is specified.”
The above table is what was desired; with respect to column and tab labels and we are now done!

### SUMMARY

Exporting from SAS to MS Excel can be relatively painless with the use of a few simple programming techniques but, you have to be careful to protect yourself from unnoticed errors.

### Contact Information

The author can be contacted at:  Joe Perry  
Perry & Associates Consulting  
342 Spring Canyon Way  
Oceanside, CA 92057  
Phone: 760-822-7959  
e-mail: PerryAJoE@cox.net

### Acknowledgements:

The author wishes to thank Sandy Owens of SAS Technical Support for her assistance in preparing this paper.
Appendix 1: Macro -- XPORTCIB2XL.SAS

Program: xPortLib2XL.SAS
Purpose: Automatically create a tabbed, column labeled XLS spreadsheet from a single SAS library, using dataset and variable LABELs rather than names.
Type: Utility

Developed by Joe Perry, Perry & Associates Consulting -- 23 MARCH 2008

Special thanks to Sandy Owens of SAS Technical Support for help in identifying different techniques for performing this export process.

Presented at PharmaSUG 2008

BEGIN - xPortLib2XL Macro

%macro xPortLib2XL ( LOI = work /* Source library of export datasets */
   , exportMethod = XLSENGINE /* Method used to export DSs|| DEF=XLSENGINE, Oth=EXPORT */
   , XLSfileName2Use = LIBDATA /* Pass in the XLS file name|| DEF=LIBDATA */
   , xCelVarThrsh = 256 /* Maximum number of VARIABLES to export to EXCEL */
   , xCelRecThrsh = 65636 /* Maximum number of RECORDS to export to EXCEL */);

   %let noCanRunWithoutErrors = 0 ; * Set the error flag to FALSE ;

   %let exportMethod = %upcase(&exportMethod.) ;
   %if &exportMethod. NE XLSENGINE or &exportMethod. NE EXPORT %then
   %let exportMethod = XLSENGINE ;

   proc contents data = &LOI._all out = dsList (keep=memname memlabel name label nobs varnum) noprint ;
   run ;
   proc summary data = dslist noprint nway missing ;
   var nobs varnum ;
   class memname memlabel ;
   output out = xPortLib2XL ( label = "Control Dataset for Looping Through List of Library Members"
      , drop = _freq_ _type_)
      max(nobs varnum)=numobs numvars ;
   run ;

   proc sql noprint ;
   select count(distinct(memname)) into :nDatasets
   from xPortLib2XL ;
   select max(numObs) into :maxNumObs
   from xPortLib2XL ;
   select max(numvars) into :maxNumVars
   from xPortLib2XL ;
   quit ;

   %if &maxNumObs. Gt &xCelRecThrsh. or &maxNumVars. Gt &xCelVarThrsh. %then %do ;
   %let noCanRunWithoutErrors = 1 ;
   %goto %doSomethingNice ;
   %end ;

END - xPortLib2XL Macro

*========================================================================*
* Do NOT let the user mess this up so, protect against poor control! *
*========================================================================*

* BUILD CONTROL DATA *
*========================================================================*

* Grab the directories contents to control dynamic data builds and error checking. Since we are going to (maybe) use both the dataset and var labels, grab all of that information now! *
*========================================================================*

* STATE INFORMATION *
*========================================================================*

* NDATASETS == Number of tabs to populate *
* MAXNUMOBS, MAXNUMVARS provide error protection *
*========================================================================*

* ERROR TRAPPING *
*========================================================================*

* 1) Determine if the number of records exceed Excels limit of 65,636 *
* 2) Determine if the number of variables exceed Excels limit of 256 *
* 3) Not sure about worksheets so, no trap for it! *
*========================================================================*

%if &maxNumObs. Gt &xCelRecThrsh. or &maxNumVars. Gt &xCelVarThrsh. %then %do ;
   %let noCanRunWithoutErrors = 1 ;
   %goto %doSomethingNice ;
%end ;
* DATASET LOOPING *
* =============== *
* Start looping if run constraints have been met *
* Grab the key information to control the writing of tabs, their names *
* and such. *
* Protect against missing and too long of dataset labels...

```sas
%do i=1 %to &nDatasets. ;
%global dsn dsnLbl numobs numvars ;
data _null_;
set xPortLib2XL ;
if n =&i. ;
if memlabel ne '' and length(memlabel) le 32 then dsnLbl=strip(memlabel) ;
else dsnLbl=memname ;
call symput( 'dsn' , strip(memname) ) ;
call symput( 'dsnLbl' , strip(dsnLbl) ) ;
call symput( 'numobs' , strip(put(numobs, 9.)) ) ;
call symput( 'numvars' , strip(put(numvars, 9.)) ) ;
run ;
```

* --- EXPORT ROUTINE --- *
* Create a file segment for inclusion in later steps to rename variables *
* using the name function *
* We need to create a rename function using LONG name but, with some *
* protection for really LONG labels so, perform this operation via SEG *
* files (meaning code SEGMNTS, not whole programs) to dynamically rename*
* the variables. *
* Write these files to work so they will disappear when SAS terminates! *

```sas
%if &exportMethod. = EXPORT %then %do ;
options validvarname=any ;
data _null_;
set dslist(where=(memname="&dsn.")) end = lastone ;
destFile="%sysfunc(pathname(work))\varRenames_&dsn..seg" ;
file dummy filevar=destFile ;
if label eq '' then label=propcase(name) ;
if _n=1 then put /*2 "rename " @ ;
sPos=20-length(name) ;
put @sPos name "= " label +(-1) "n" ;
if lastOne then put @15 ";" ;
run ;
data &dsn. ;
set &LOI.\&dsn. ;
%inc "%sysfunc(pathname(work))\varRenames_&dsn..seg" / source ;
run ;
proc export data = &dsn. ;
outfile="%sysfunc(pathname(&LOI.))\&XLSfileName2Use..xls" ;
DBMS=EXCEL REPLACE;
sheet="&dsnLbl." ;
RUN ;
%end ;
```

* --- XLSENGINE ROUTINE --- *
* This is a pretty simple routine now so, just let the LOOP CONTROLLER *
* cycle through and export these datasets to their respective tabs. *

```sas
%if &exportMethod. = XLSENGINE %then %do ;
libname myxls "%sysfunc(pathname(&LOI.))\&XLSfileName2Use..xls" ;
data myxls."&dsnLbl."n (dblabel=yes) ;
set &LOI.\&dsn. ;
run ;
libname myxls clear ;
%end ;
```
doSomethingNice:
   %if &noCanRunWithoutErrors. = 1 %then %do ;
   data _null_ ;
   put //
   @10 "===================================================================" /
   @10 "Warning... Warning... Warning... Warning... Warning... Warning... " /
   @10 "===================================================================" /
   @12 "You just might want to check the log! " /
   @12 "At least one dataset had too many RECORDS or VARIABLES to export " /
   @12 "Max # of Records found was &maxNumObs. > " /
   @12 "Max # of Variables found was &maxNumVars. > " /
   @12 "You just might want to check the log! " /
   @10 "===================================================================" /
   @10 "Warning... Warning... Warning... Warning... Warning... Warning... " /
   @10 "===================================================================" ;
   run ;
%end ;
%mend xPortLib2XL ;

END - xPortLib2XL Macro

Developed by Joe Perry, Perry & Associates Consulting -- 23 MARCH 2008

Presented at PharmaSUG 2008

ERROR MESSAGING

Let someone know something happened to end this process