

## A Poor/Rich SAS® User's Proc Export

Arthur S. Tabachneck, Ph.D., myQNA, Inc., Thornhill, Ontario Canada

Tom Abernathy, Pfizer, Inc., New York, NY

Randy Herbison, Westat, Rockville, MD

Matthew Kastin, I-Behavior, Inc., Penn Valley, PA

### ABSTRACT

Have you ever wished that with one click you could copy any SAS® dataset, including variable names, so that you could paste the text into a Word file, powerpoint or spreadsheet? You can and, with just base SAS, there are some little known but easy to use methods that are available for automating many of your (or your users') common tasks.

### BACKGROUND

With the introductions of both 64-bit computers, and XLXS versions of Excel Workbooks, the task of exporting SAS® datasets to Excel has become increasingly difficult and often requires the purchase of additional products. Interestingly, a little known SAS Explorer feature provides a free alternative. When one right clicks on a dataset name in SAS Explorer, one of the menu options is *Copy Contents to Clipboard*. Clicking on that option writes an HTML version of the dataset to one's clipboard, thus allowing the data to be pasted into Excel or any other program that allows pasting from one's clipboard.

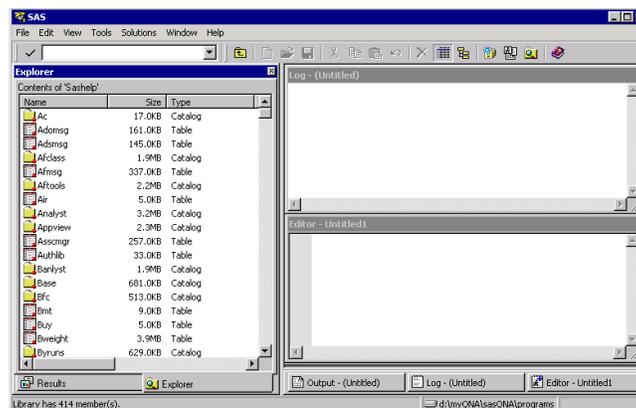
The present paper goes a couple of steps further. In addition to the actions you see when right clicking on a dataset, SAS Explorer provides a way for users to add their own actions to the menu. This paper provides step-by-step instructions on how to add *Copy to Clipboard* and *Export to Excel* actions designed to your own specifications.

### SAS EXPLORER

Many SAS users may not know about the *Copy Contents to Clipboard* because it is only briefly mentioned in the documentation. We discovered it while writing this paper which, originally, was going to offer a PROC EXPORT alternative using the CLIPBOARD Access Method. In the documentation's description of the CLIPBOARD Access Method, the following statement can be found: *You can also copy data to the clipboard by using the Explorer pop-up menu item **Copy Contents to Clipboard**.*

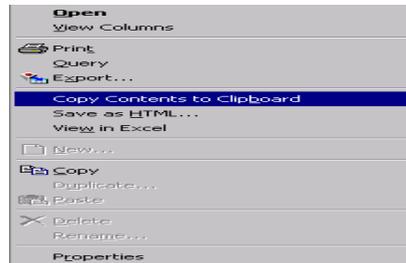
Any attempt to explain features of the SAS Windowing environment is complicated by the fact that users can alter the placement and presence of the various windows that will appear on their monitors when they open an interactive SAS session. A screenshot of one of the author's typical SAS Windowing environment is shown, below, in Figure 1.

Figure 1  
One Way a SAS Windowing Environment May Appear



The SAS Explorer window, shown on the lower left side of the configuration shown in Figure 1, is the window from which one can view all active SAS libraries and their contents. The screenshot shown in Figure 1, specifically, is the way the screen would appear if one has traversed through the SAS Explorer menu to show all of the tables (i.e., SAS datasets) and catalogs that exist in the Sashelp library. Right clicking on the *name* of any table shown in that window will cause the pop-up menu, like the one shown in Figure 2, to appear:

**Figure 2**  
**An Explorer Pop-up Menu**



Moving one's mouse up and down through the pop-up menu makes the various options active. In our example, *Copy Contents to Clipboard* is *active* and, if one left-clicks while an option is active, they will cause the action to occur. This particular action creates an HTML version of the dataset and places that version within one's clipboard so that it can be pasted into an Excel workbook or any other program that includes a paste feature.

## ADDING YOUR OWN MENU ACTIONS

As mentioned earlier, we discovered the *Copy Contents to Clipboard* action during our review of the documentation pertaining to the Clipboard Access Method, as our original intent was to write and present code that could be used to provide such a capability. We could have stopped as soon as we discovered the already available feature, but we didn't as the feature didn't quite meet our specifications. Specifically, the HTML file included a couple of undesired header records and highlighted borders around all of the cells.

The current paper's authors were already familiar with adding SAS Explorer menu options as three of us had collaborated on writing a paper, in 2010, which used the method to give users a way to copy and paste the variable names from any SAS dataset. However, other than our own paper, and some excellent papers by Richard DeVenezia and Art Carpenter, only one paragraph in the SAS documentation even mentions the capability. However, creating a pop-up action is fairly straight forward:

1. From the Explorer window, select Tools ➔ Options ➔ Explorer
2. Left click on Members
3. Double left click on Table
4. Left click on Add.
5. Enter a name for the action (this is the action that will appear on the pop-up menu for the item). If you include an ampersand (i.e. an & symbol) before one of the letters, the letter will serve as a shortcut key that can be used to select the action
6. Enter the command you want run whenever the action is selected
7. Select OK to exit the Add Action screen.
8. Select OK to exit the Table Options screen

You can submit any SAS code as long as the code uses 255 or fewer characters, uses two % signs whenever one is normally needed and, if quotation marks are needed, either uses masking functions or very carefully balanced quotation marks.

## WORKING WITHIN THE 255 CHARACTER LIMITATION

Of course, writing code given such limitations can easily look like code that Art Carpenter has described as "Programming for Job Security". The following code, entered as the action command described in step 6, above, uses the gsubmit command to submit a program that:

1. Uses PROC TRANSPOSE to create a file containing a SAS dataset's variable names and position
2. Obtains the libname and filename using the %8b and %s that are passed by SAS Explorer
3. Uses PROC SQL to create a macro variable containing an ordered delimited list of the variable names
4. Uses the &syslast system macro variable to select the data file that was created by proc transpose
5. Uses a drop table statement to delete the file that was created by proc transpose
6. Uses a filename statement to declare a file using the Clipboard Access method
7. Uses a data \_null\_ step to write a tab-delimited file to one's clipboard, first writing a line that displays the variable names, and then one line for each record, displaying each record's values for each variable
8. Uses a filename statement to clear the fileref created in step 5

```
gsub 'proc transpose data=%8b.%s(obs=0);var _all_;proc sql noprint;select
_name_ into :_ separated by " " from &syslast;drop table
&syslast;filename c clipbrd;data _null_;file c dsd dlm=" "; set %8b.%s;if
_n_=1 then put "&_";put(_all_)(:);run;filename c;';
```

While we couldn't show it, above, the action command is actually 255 continuous characters. To keep the code within the 255 character limitation, only the minimum number of statements, and non-descriptive one character file names were used. Also, using "09"x in either the separated by clause or dlm option would have caused us to exceed limitation, an actual tab character was used. That can be easily typed in Windows by holding the Alt key down, pressing 0 and then 9 on one's numeric keypad, and then releasing the Alt key.

## GETTING AROUND THE 255 CHARACTER LIMITATION

One can avoid the 255 character limitation by either wrapping the code in a SAS macro, or creating and wrapping a PROC FCMP function in a macro. Of course, doing either adds a layer of complexity, as the macro would have to be saved in a directory declared in one's sasautos' path (see, e.g., <http://analytics.ncsu.edu/sesug/2008/SBC-126.pdf>). As an example of simply wrapping the code in a macro, the following macro could be used in place of the code we described previously:

```
%macro copy2clipbrd(lib,mem);
  proc transpose data=&lib..&mem. (obs=0);
    var _all_;
  run;
  proc sql noprint;
    select _name_ into :_names
      separated by "09"x from &syslast;
    drop table &syslast;
  quit;
  filename clippy clipbrd;
  data _null_;
    file clippy dsd dlm="09"x;
    set &lib..&mem.;
    if _n_=1 then put "&_names.";
    put(_all_)(:);
  run;
  filename clippy clear;
%mend copy2clipbrd;
```

The action command would be: `gsubmit '%copy2clipbrd(%08b,%32b)';`

A limitation of both of the above approaches is that the clipboard access method, as used in the two calls, can only be used to copy and paste up to 256 characters per record. That limitation can be eliminated by creating and wrapping a PROC FCMP function in a macro. The following code, which would be submitted using the same action command (i.e., `gsubmit '%copy2clipbrd(%08b,%32b)';`): (1) creates a macro that contains the needed PROC FCMP code; temporarily modifies the CMPLIB option to allow use of the newly created function

outside of PROC FCMP; (3) uses SAS file I/O functions to access the clipboard and perform all of the necessary operations; and (4) upon exiting closes all I/O streams to ensure a clean programming environment.

```

%macro copy2clipbrd(lib,mem);
  proc fcmp outlib=work.func.util;
    function c2cb(lib $,mem $);
      rc=filename('clippy',' ','clipbrd');
      if rc ne 0 then return(1);
      fid=fopen('clippy','o',0,'v');
      if fid eq 0 then do;
        rc = filename( 'clippy' );
        return(2);
      end;
      dsid=open(catx('.',lib,mem));
      if dsid eq 0 then do;
        rc=fclose(fid);
        rc=filename('clippy');
        return(3);
      end;
      rc=attrn(dsid,'any');
      if rc ne 1 then do;
        rc=fclose(fid);
        rc=close(dsid);
        rc=filename('clippy');
        return(4);
      end;
      nvar=attrn(dsid,'nvar');
      array v[1] /nosymbols;
      call dynamic_array(v,nvar);
      do i = 1 to nvar;
        if i gt 1 then rc=fput(fid,'09'x);
        rc=fput(fid,varname(dsid,i));
        v[i]=ifn( vartype(dsid,i)='C',1,2);
      end;
      rc=fwrite(fid);
      do i=1 to attrn(dsid,'nlobs');
        rc=fetchobs(dsid,i);
        do j=1 to nvar;
          if j gt 1 then rc=fput(fid,'09'x);
          if (v[j] eq 1) then rc=fput(fid, getvarc(dsid,j));
          else do;
            fmt=varfmt(dsid,j) ;
            if missing(fmt) then fmt='best12.';
            rc=fput (fid,putc(putn(getvarn(dsid,j) ),fmt ),'$char12.'));
          end;
        end;
        rc=fwrite(fid);
      end;
      rc=fclose(fid);
      rc=close(dsid);
      rc=filename('clippy');
      return(0);
    endsub;
  quit;
  %local cmplib;
  %let cmplib=%sysfunc(getoption(cmplib));
  options cmplib=(work.func);
  %put %sysfunc(c2cb(&lib,&mem)) ;
  options cmplib=(&cmplib);
%mend copy2clipbrd;

```

## AUTOMATING PROC EXPORT

If you can already use PROC EXPORT to create Excel files you still might be interested in adding the task as a SAS Explorer menu action. For example, clicking on an action that used the following code as its action command would create an Excel file with the same name as the SAS dataset and in the same directory where the dataset resides:

```
gsubmit '%%let filepath=%sysfunc(pathname(%8b)); %%let filenm=%s;
PROC EXPORT DATA=%8b.%s OUTFILE="%&filepath.\&filenm..xlsx" DBMS=XLSX
REPLACE;RUN;';
```

## WHERE TO GET THE MACRO

We did our best to only include carefully written and tested code, but the code may have to be updated from time to time to correct for errors or enhancements that we or others might discover. Additionally, while a copy of the macro is included in this paper, copying and pasting from a pdf file often introduces stylish looking quotation marks which aren't correctly recognized by SAS. As such, we created a page for the paper on [sasCommunity.org](http://www.sascommunity.org/wiki/A_Poor/Rich_SAS_Users_Proc_Export). The page includes copies of the source code and updated versions of this paper. The page can be found at: [http://www.sascommunity.org/wiki/A\\_Poor/Rich\\_SAS\\_Users\\_Proc\\_Export](http://www.sascommunity.org/wiki/A_Poor/Rich_SAS_Users_Proc_Export)

## SUMMARY AND CONCLUSION

The purpose of the present paper was to create and describe alternative menu driven methods one could use for exporting SAS datasets to Excel. Four methods were provided, each using SAS Explorer's built-in functionality for adding pop-up menu actions. The methods are generalizable to almost any task that has to be repeated. Thus, one could use the methods to build menu items for running the INSIGHT software, PROC CONTENTS, PROC MEANS, PROC UNIVARIATE, or any proc or custom macro one might typically use to gain an understanding of any dataset, all with just one click.

## DISCLAIMER

The content of this paper is the work of the authors and does not necessarily represent the opinions, practices or recommendations of the authors' organizations.

## CONTACT INFORMATION

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

Arthur Tabachneck, Ph.D., President  
myQNA, Inc. Thornhill, ON Canada  
E-mail: [atabachneck@gmail.com](mailto:atabachneck@gmail.com)

Tom Abernathy  
Pfizer, Inc.  
235 E. 42nd Street  
New York, NY 1001  
E-mail: [tom.abernathy@pfizer.com](mailto:tom.abernathy@pfizer.com)

Randy Herbison, Senior Systems Analyst  
Westat  
1650 Research Boulevard  
Rockville, MD  
E-mail: [RandyHerbison@westat.com](mailto:RandyHerbison@westat.com)

Matthew Kastin  
I-Behavior, Inc.  
Penn Valley, PA  
E-mail: [matthew.kastin@gmail.com](mailto:matthew.kastin@gmail.com)

## REFERENCES

*Automagically Copying and Pasting Variable Names*, Tabachneck, A., Herbison, R., Clapson, A., King, J., DeAngelis, R. and Abernathy, T., SGF 2010 Proceedings, <http://support.sas.com/resources/papers/proceedings10/046-2010.pdf>  
*Copy and Paste Almost Anything*, Tabachneck, A., Herbison, R., King, J., DeVenezia, R., Derby, N., and Powell, B., SGF 2012 Proceedings, <http://support.sas.com/resources/papers/proceedings12/238-2012.pdf>

*Doing More with the SAS® Display Manager: From Editor to ViewTable - Options and Tools You Should Know*, Carpenter, A., SGF 2012 Proceedings, <http://support.sas.com/resources/papers/proceedings12/151-2012.pdf>

*FILENAME, CLIPBOARD Access Method*, SAS 9.2 Documentation, SAS Institute 2012, <http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002571877.htm>

*Programming for Job Security: Maximize Your Indispensability - Become a Specialist*, Carpenter, A., Presented as a luncheon keynote address at the SAS Global Forum 2008 Conference in San Antonio, Texas  
<http://www.caloxy.com/papers/80JobSecuritySpecialist.pdf>

*SAS® Explorer: Use and Customization*, DeVenezia, R., NESUG 2005, <http://www.nesug.org/proceedings/nesug05/ap/ap6.pdf>

*Step-by-Step Programming with Base SAS® Software, Customizing the SAS Widowing Environment*  
<http://support.sas.com/documentation/cdl/en/basess/58133/HTML/default/viewer.htm#a001115650.htm>

*Which SASAUTOS Macros Are Available to My SAS® Session?*, Droogendyk, H, SESUG 2008, <http://analytics.ncsu.edu/sesug/2008/SBC-126.pdf>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.