



Business Analytics



PROPER HOUSEKEEPING – DEVELOPING THE PERFECT “MAID” TO CLEAN YOUR SAS® ENVIRONMENT

CHUCK BININGER, COMSYS, KALAMAZOO, MI

Do business. **We're IT.**

ABSTRACT

As programs run in any SAS® environment they often leave messy residual artifacts lying around. It is this mess that can often create unexpected results when running a SAS® program. Sometimes Titles and Footers need to be cleared, options reset to their original values, or macro variables put in the trash. We try to pick up after ourselves by using good programming practices but often we fall short and find ourselves in need of a good scrubbing.

This paper will not only show the readers where to look for all dirt that may affect your environment but also supply the reader with the tools needed to keep your environment clean.

INTRODUCTION

As an application developer I find myself in situations where I need to run a series of queries and/or reports in series in the same SAS® session. The most desirable approach is to run each SAS® job in its own SAS® session. Sometimes this is not possible. In these situations you must ensure that a proceeding program does not alter or influence how successive programs run.

The following items need to be addressed:

1. OPTIONS
2. GOPTIONS
3. System macro variables
4. Temporary data sets and formats
5. Libraries
6. Filenames
7. Titles and Footnotes
8. Macros
9. Macro variables

Ignoring any of these items can cause your SAS® program to have unintended results.

OPTIONS

SAS® options are usually either defined by the programmer or established by a department as guidelines. In either case it is necessary to take a snap shot of the SAS® options in order to return the environment to that state. The following code saves SAS® options as a SAS® data set in the SASUSER library.

```
proc optsave out=SASUser.MyOptions;
run;
```

In order to reset options back to the original state execute the following.

```
proc optload data=SASUser.MyOptions;
run;
```

GOPTIONS

Resetting GOPTIONS to its original state is easy but extremely important. It is common for graphs to have unexpected formatting based on changes to the GOPTION values. It is actually good programming practice to reset the GOPTIONS prior to running your code. The following code solves the problem.

```
goptions reset=all;
```

SYSTEM MACRO VARIABLES

Resetting system macro variables is probably the most important thing that needs to be done. If any of these macro variables are set to a non-zero value SAS® may or may not execute code. It is also important to make sure that the OBS option be reset to MAX. The same error that triggers the following system macro variables can also set OBS=0. The macro variables and OBS option can be reset as follows.

```
%let syscc = 0; ** Operating environment condition
code **;
%let sysrc = 0; ** Operating system condition
code **;
%let syslibrc = 0; ** Libname statement condition
code **;
%let sysfilrc = 0; ** Filename statement condition
code **;
%let syslockrc = 0; ** SAS® Share lock statement
condition code **;
%let syslast = ; ** Contains last created data set
**;
options obs = max; ** Resets the number of
observations to process **;
```

TEMPORARY DATA SETS AND FORMATS

It is always good practice to delete any unneeded data sets during the program execution. This can be done by executing the following.

```
proc datasets
library = work
delete <name of data set>;
quit;
```

However if you have neglected to clean along the way the following code will wipe away all data sets in the work library. Notice the kill statement. This statement deletes all the data sets and the format catalog in the work library. The format catalog is extremely important to delete because depending on the value of the FMTSEARCH option unexpected results could take place.

```
proc datasets
library = work
kill;
quit;
```

LIBRARIES

The first step to un-assign all the defined libnames is find out what libnames are assigned. This can be done by looking in the dictionary table DICTIONARY.LIBNAME. The trick here is to create one macro variable and use the SEPARATED BY statement. This alleviates the need to create multiple macro variables and loop through them.

```
proc sql noprint;
  select unique libname into :mylibs separated by
  \ clear; libname \
  from dictionary.libnames
  where libname not in ('MAPS','SASHELP','SASUSER',
  'WORK');
quit;
libname &mylibs clear;
```

FILENAMES

Filenames can be handled exactly the same way as libraries. You will need to query DICTIONARY.EXTFILES to determine

```
proc sql noprint;
  select unique fileref into :myfiles separated by
  \; filename \
  from dictionary.extfiles
  where substr(fileref,1,1) ^= '#';
quit;

filename &myfiles;
```

TITLES AND FOOTNOTES

Titles and footnotes are easily cleared by submitting the following statements.

```
title;
footnote;
```

MACROS

Macros are particularly important to clean up because depending on the value of option SASAUTOS it is possible that if there is two macros with the same name the wrong would could be executed. The following code will delete all macros defined to the work.sasmacr catalog.

```
proc catalog
  catalog = work.sasmacr kill force;
run;
```

MACRO VARIABLES

It is typically only necessary to cleanup global macro variables. Local macro variables are cleaned up by the SAS® system once the macro is executed. Global macros can be determined by querying the dictionary table DICTIONARY.MACROS.

```
proc sql noprint;
  select name into :mymacrovars separated by ' '
  from dictionary.macros
  where scope = 'GLOBAL';
quit;
%symdel &mymacrovars mymacrovars;
```

Notice that the macro variable MYMACROVARS is also added to the %SYMDEL statement.

IMPLEMENTATION

Now that we know how to handle each SAS® component that can affect other SAS® processes within the same SAS® session, we need to create a process that we can use to “SCRUB” the SAS® environment. One solution that I find extremely handy is creating a macro in a macro library that incorporates the above code into a clean up utility. By permanently storing the macro it does not become the clutter that we are trying to clean up.

CONCLUSION

When writing code, we all have intentions of cleaning up unnecessary data sets, macro variables and other artifacts left over from previous code. This can cause unwanted behavior in later programs or modules. By following the above guide you can reduce the risk of contamination from previously run programs.

ACKNOWLEDGMENTS

Special thanks to Brian Varney for providing the idea of creating a process to clean up a SAS® session.

CONTACT INFORMATION

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

Chuck Bininger
COMSYS
5220 Lovers Lane
Portage, MI 49002
269.553.5111
cbininger@comsys.com
sas.comsys.com

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.

Other brand and product names are trademarks of their respective companies.