PharmaSUG2011 - Paper CC05

Macros to Help You Clean Up!

Kavitha Madduri, Boehringer Ingelheim Pharmaceuticals Inc., Ridgefield, CT

ABSTRACT:

All programs leave their footprint when executed in SAS®. These programs modify the default SAS environment settings - options, formats, library references, titles, footnotes, global macro variables, and temporary datasets. This paper introduces two macros, USTART and UEND, which provide an easy and efficient solution to handle the housecleaning chores for a program and help prevent unexpected results for programs that follow.

INTRODUCTION:

Programmers have many styles for cleaning up programs. Sometimes cleanup occurs by explicitly using SAS statements and procedures, such as those for datasets as with PROC DATASETS. Here are some examples of how a programmer can explicitly cleanup the SAS session.

SAS OPTIONS:

```
/* first store the original option settings in macro variables */
%let xmprnt   = %sysfunc(getoption(mprint));
%let xmlgc    = %sysfunc(getoption(mlogic));
%let xsymblgn = %sysfunc(getoption(symbolgen));
%let xfmtr    = %sysfunc(getoption(fmterr));
%let xformc   = %sysfunc(getoption(formchar));

/* at the end of the program, reset the options */
options &xmprnt &xmlgc &xsymblgn &xfmtr ;
options formchar="&xformc";
```

FORMATS:

```
/* delete newly created formats within the program */
proc catalog catalog=work.formats;
   delete fmt1c.formatc fmt2n.format fmt3n.format fmt4n.formatc;
run;
```

LIBRARY REFERENCES:

```
/* clear the libnames */
libname lib1;
libname lib2;
libname lib3;
libname lib4;
```

TITLES AND FOOTNOTES:

```
/* clear titles */
title;

/* clear footnotes */
footnote;
```

GLOBAL MACRO VARIABLES:

```
/* delete macro variables */
%symdel xmprnt xmlgc xsymblgn xfmtr xformc;
```

TEMPORARY DATASETS:

```
/* delete temporary datasets created within the program */
proc datasets lib=work mt=data;
    delete temp1 temp2 temp3 temp4;
quit;
```

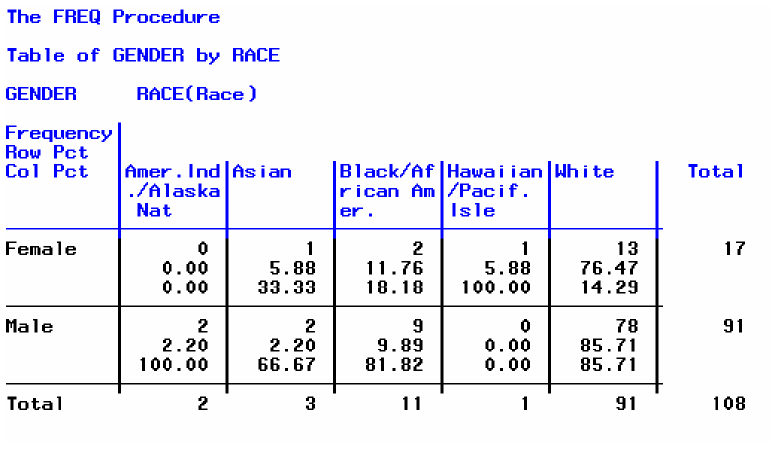
A programmer has to remember to include all of the necessary housecleaning code in each and every program written. If one program fails to clean up all the changes made by that program we will have to deal with carryover issues. When several programs are run in sequence, the environment changes from one program are inadvertently carried over to the next program(s), sometimes resulting in misleading behavior. For example, global macro variable(s) and/or format(s) changes in one program get carried over to the subsequent program(s) producing unexpected results. It is time consuming to debug issues like these when no obvious signals in the log exist.

Here is a simple example to show how a format update in one program affects another program run in sequence. In a SAS session standard formats are usually already made available. Below is one such example for a standard gender format.

```
/* standard format for gender available at the beginning of SAS session */
proc format;
    value genderf
        1 = 'Male'
        2 = 'Female';
run;
```

Program 1 is run which recreates the gender format and produces the below frequency table:

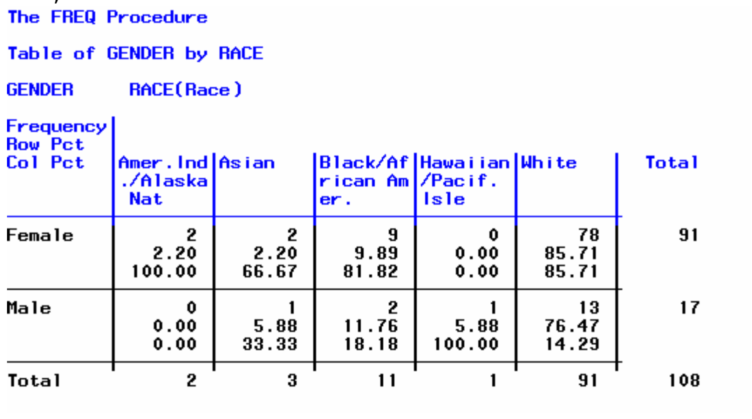
```
/* program 1*/

/* user updated format for gender */
proc format;
    value genderf
        1 = 'Female'
        2 = 'Male';
run;

proc freq data=patd;
    tables gender*race/nopct;
run;
```



```
The FREQ Procedure

Table of GENDER by RACE

GENDER     RACE(Race)

Frequency|
Row Pct  |
Col Pct  |Amer.Ind|Asian   |Black/Af|Hawaiian|White   |  Total
         |./Alaska|        |rican Am|/Pacif. |        |
         | Nat    |        |er.     |Isle    |        |
---------+--------+--------+--------+--------+--------+
Female   |      0 |      1 |      2 |      1 |     13 |     17
         |   0.00 |   5.88 |  11.76 |   5.88 |  76.47 |
         |   0.00 |  33.33 |  18.18 | 100.00 |  14.29 |
---------+--------+--------+--------+--------+--------+
Male     |      2 |      2 |      9 |      0 |     78 |     91
         |   2.20 |   2.20 |   9.89 |   0.00 |  85.71 |
         | 100.00 |  66.67 |  81.82 |   0.00 |  85.71 |
---------+--------+--------+--------+--------+--------+
Total           2        3       11        1       91      108
```

After program 1 is run, program 2 is run without knowing that the standard gender format has been updated. Program 2 produces the below frequency table:

```
/* program 2*/
proc freq data=demo;
    tables gender*race/nopct;
run;
```



```
The FREQ Procedure

Table of GENDER by RACE

GENDER      RACE(Race)

Frequency|
Row Pct  |
Col Pct  |Amer.Ind|Asian   |Black/Af|Hawaiian|White   |  Total
         |./Alaska|        |rican Am|/Pacif. |        |
         | Nat    |        |er.     | Isle   |        |
---------+--------+--------+--------+--------+--------+
Female   |      2 |      2 |      9 |      0 |     78 |     91
         |   2.20 |   2.20 |   9.89 |   0.00 |  85.71 |
         | 100.00 |  66.67 |  81.82 |   0.00 |  85.71 |
---------+--------+--------+--------+--------+--------+
Male     |      0 |      1 |      2 |      1 |     13 |     17
         |   0.00 |   5.88 |  11.76 |   5.88 |  76.47 |
         |   0.00 |  33.33 |  18.18 | 100.00 |  14.29 |
---------+--------+--------+--------+--------+--------+
Total           2        3       11        1       91      108
```

Above example clearly shows how issues like these will go undetected because of no obvious signals in the log. When the above two tables go for validation, time will be spent to investigate the reason for discrepancy in the counts. It will take some time before one can identify and trace back the issue to format update.

Proper housecleaning by each program is a key to avoiding interactions between programs. Each program should return the SAS environment to its original state when it concludes. To prevent situations as above, we have addressed the full scope of potential carryover errors into two user friendly macros - USTART and UEND. These macros must be used together. The USTART macro, called in the beginning of a program, takes a snapshot of the SAS environment, while the UEND macro, called at the end of the program, restores the environment to its original state. This eliminates the need to replicate extensive housecleaning code in each program.

USTART and UEND handle the ‘housecleaning’ chores both on the local session and the remote host. Table 1 shows how these macros use dictionary tables and SAS procedures for housecleaning.

Table 1: Description of USTART and UEND tasks

SAS environment Settings	**Dictionary tables/ Procedures used**	**Description**
Options	PROC OPTSAVE and PROC OPTLOAD	USTART stores all the SAS options into a dataset **optsave** using Proc optsave. UEND uses that dataset with Proc optload and restores the options
Formats	PROC FORMAT	USTART stores all the SAS formats into a dataset **fmtsave** using the Proc format cntlout option. UEND uses that dataset with the Proc format cntlin option and restores all formats.
Library references	SASHELP.VSLIB	USTART uses dictionary table VSLIB and stores the library references in the **libsave** dataset. UEND uses VSLIB and creates a temporary dataset. It then compares this with **libsave** and restores the library references.
Titles and Footnotes	SASHELP.VTITLE	USTART uses dictionary table VTITLE and stores all the titles and footnotes in the **ttlftsave** dataset. UEND uses VTITLE and creates a temporary dataset. It then compares this with **ttlftsave**. It deletes any newly created titles and footnotes, resets any update titles and footnotes, and creates any cleared titles and footnotes.
Global macro variables	SASHELP.VMACRO	USTART uses dictionary table VMACRO and stores only the global macro variables in the **macsave** dataset excluding the global macro variables that start with ‘SQL’ and ‘SYS’. UEND uses VMACRO and creates a temporary dataset. It then compares this with **macsave.**

		It deletes newly created global macro variables, restores any updated global macro variables, and creates any deleted global macro variables.
Temporary datasets	SASHELP.VCOLUMN	USTART uses dictionary table VCOLUMN and stores a list of datasets that are currently in WORK/RWORK to **dssave** dataset. UEND uses VCOLUMN and creates a temporary dataset that lists the datasets available in WORK/RWORK. It then compares this with the **dssave** and deletes any newly created datasets. Depending on the *flag* value it decides whether to delete or keep the temporary datasets in WORK/RWORK.

The macros are discussed here to give an idea on how to accomplish the task of housecleaning. As the SAS code for these macros is proprietary, it will not be shared. Sample calls are shown later in this paper to give an idea as to how to handle proper housecleaning.

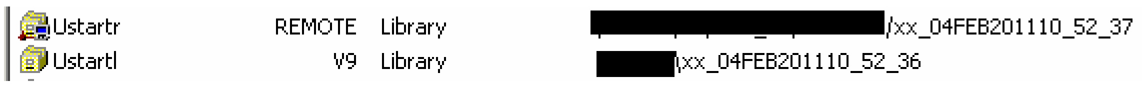
USTART

This macro takes a snapshot of SAS environment settings- options, formats, library references, titles, footnotes, global macro variables, and temporary datasets. It takes two parameters (Table 2)

Table 2: Parameters for USTART

Parameter name	**Description**	**Valid values**	**Default value**
Typ_host	Type of host – local or remote	Local, remote, both	Both
Data_Lst	List of datasets to delete before taking the snapshot	<list of datasets>, ALL	-

USTART gives the user an option to choose the host where both the USTART and UEND macros need to be executed. Whatever value for *typ_host* is given to the USTART macro is also used by UEND macro. The *data_lst* parameter gives the user an option to delete any or all of the temporary datasets from WORK/RWORK libraries. This gives the user the option to clean up and control which datasets should reside in WORK/RWORK before the program continues execution. Further discussion of the macro assumes the default value of 'both' for the *typ_host* parameter. USTART and UEND macros create temporary datasets in the process of 'housecleaning' as defined in Table 1. In order for these datasets not to interfere with any of the user created datasets, USTART creates a subfolder under WORK and RWORK to store the temporary datasets created by these macros. The first thing this macro does is to create a subfolder under WORK and RWORK, and assign libnames ***ustartl*** and ***ustartr***. The name of the subfolder is created based on the time the macro is executed. For example if the macro was run on 4th February at 10:52:36 AM, first a subfolder would be created under WORK library with the name - **xx_04FEB201110_52_36** and then a subfolder under RWORK with the name - **xx_04FEB201110_52_37**. Once these folders are created, respective libnames ***ustartl*** and ***ustartr*** are assigned.



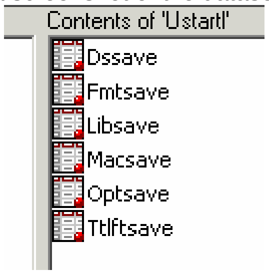
```
Ustartr      REMOTE   Library   [redacted]/xx_04FEB201110_52_37
Ustartl          V9   Library   [redacted]\xx_04FEB201110_52_36
```

Once the libnames are assigned, using SAS dictionary tables USTART saves the SAS environment information into temporary datasets as shown by below SAS code (for local session only). The same thing is done on the remote host as well. USTART does not save any of the system global macro variables that start with "SQL" or "SYS" like the SQLXOBS, SQLRC, SQLEXITCODE, SYSDBASE, and SYSDFINAL. It's a good programming practice not to use these global macro variable names in the programs. When storing the library references information, USTART macro excludes SASHELP and WORK library references as these cannot be changed by the user.

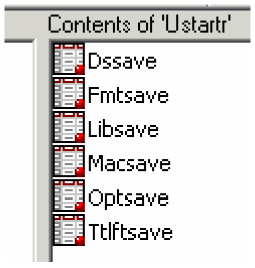
```
/* store formats */
proc format cntlout=ustartl.fmtsave;
```

```
run;


/* store global macro variables excluding some of the system and SQL ones */
data ustartl.macsave;
    set sashelp.vmacro(where=(scope eq 'GLOBAL' and
                              name not like "%%SQL%" and
                              name not like "%%SYS%")
                       );
run;


/* store options */
proc optsave data=ustartl.optsave;
run;


/* store datasets names that are currently in WORK */
proc sql;
    create table ustartl.dssave as
        select distinct libname, memname
        from sashelp.vcolumn
        where libname eq "WORK" and
              memtype eq "DATA";
quit;

/* store titles and footnotes */
data ustartl.ttlftsave;
    set sashelp.vtitle;
run;

/* store library references excluding the sashelp and work */
data ustartl.libsave;
    set sashelp.vslib(where=(libname not in ('SASHELP', 'WORK'))
                     );
run
```

Here is a screenshot of the datasets in the ***ustartl*** and ***ustartr*** libraries.



Contents of 'Ustartl'
- Dssave
- Fmtsave
- Libsave
- Macsave
- Optsave
- Ttlftsave



Contents of 'Ustartr'
- Dssave
- Fmtsave
- Libsave
- Macsave
- Optsave
- Ttlftsave

UEND
UEND restores the SAS environment back to the way it was before the program was executed. All settings of SAS options, formats, libnames, titles, footnotes, global macro variables and datasets are compared against the stored datasets that USTART created. UEND will then 'UNDO' all of the detected changes that occurred during the program's execution. UEND takes two parameters (Table 3)

Table 3: Parameters for UEND

Parameter name	**Description**	**Valid values**	**Default value**
Flag	Flag whether to keep or drop the datasets	Keep, drop	keep
Data_Lst	List of datasets to keep or drop	<list of datasets>	-

With the UEND macro the programmer has a choice to decide which datasets to keep or drop after UEND is executed. Sometimes a program creates datasets that may be used by the subsequent program(s). In this case we do not want UEND to clean up all the temporary datasets. In this situation you would give the name(s) of the dataset to *data_lst* parameter and the value for *flag* would be 'keep'. If you want to keep most of the datasets created but just delete few of them, then you would give the list of datasets to delete to *data_lst* parameter and the value for *flag* would be 'drop'. UEND compares the SAS environment settings after a program is run with the ones stored by USTART. After each comparison, the respective settings are restored to how they were before USTART was executed. Once all the SAS environment settings are restored, UEND deletes the subfolders created under WORK/RWORK.

SAMPLE CALLS

Call1

```
/* with default options */
%ustart;
%uend;
```

Call2:

```
/* local session without deleting any datasets */
%ustart(typ_host=local);

/* keep only the POPUL dataset in the WORK folder and delete all other datasets */
%uend(data_lst=popul);
```

Call3:

```
/* remote session without deleting any datasets */
%ustart(typ_host=remote);

/* delete AE and ADM datasets from RWORK folder.Do not delete any other datasets */
%uend(flag=drop, data_lst=ae adm);
```

Call4:

```
/* local session to delete only selected datasets */
%ustart(typ_host=local, data_lst=patd aee adm);

/* delete only the lab dataset in the WORK folder. Do not delete any other datasets */
%uend(flag=drop, data_lst=lab);
```

Call5:

```
/* remote session to delete only selected datasets */
%ustart(typ_host=remote, data_lst=ct cd medh);

/* keep only HIV and VLOAD datasets in the RWORK folder and delete all other datasets */
%uend(flag=keep, data_lst=hiv vload);
```

LIMITATIONS:

SAS catalogs and graphic options settings are not supported by these macros at this time. They will eventually be included in the later versions of the macros. Another current limitation is that changes to preexisting datasets (prior to USTART) are not detected. The macro will be updated later as well to overcome this limitation.

CONCLUSION:

All programs should clean up after themselves in order to avoid unintended interactions with other programs that follow. The code for performing such complete housecleaning can be quite lengthy. So it would not be desirable to repeat this code in each program. The macros discussed in this paper, however, simplify and centralize the process and are therefore recommended to be used in every program.

ACKNOWLEDGMENTS:

The author would like to thank John Adams for his technical guidance while developing these macros.

CONTACT INFORMATION:

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

Kavitha Madduri
Boehringer Ingelheim Pharmaceuticals Inc.
900 Ridgebury Road
Ridgefield, CT, 06877-0368
Work Phone: 203-791-6208
Fax: 203-837-4409
Email: Kavitha.Madduri@Boehringer-Ingelheim.com
Kavitha.Madduri@gmail.com