SESUG Paper 102-2025

Selective SAS Option Restoration with PROC COMPARE and PROC OPTLOAD

John King, Ouachita Clinical Data Services

ABSTRACT

Managing system options in SAS macros is often error-prone, especially when global settings are altered temporarily. This paper presents a macro `%u_optstore` that allows for the safe saving and selective restoration of only the SAS options that have changed. By combining PROC OPTSAVE, PROC COMPARE, and PROC OPTLOAD, this approach minimizes risk, supports nested macros, and provides debugging transparency. It also avoids known issues associated with full option reloads, such as unwanted resets to session defaults.

INTRODUCTION

SAS provides PROC OPTSAVE and PROC OPTLOAD to manage system options, but these tools apply changes indiscriminately. Reloading all system options can cause unexpected changes to the SAS environment, especially in nested or shared contexts.

This paper introduces a macro-based method to selectively restore only those system options that have been modified. The `%u_optstore` macro is designed for safe use in reusable code and supports macro nesting through use of the macro call stack depth.

BACKGROUND

PROC OPTSAVE allows the user to write all current SAS system option values to a dataset, and PROC OPTLOAD can later reload those settings. However, these procedures offer no built-in mechanism to restore only changed values.

PROC COMPARE is widely used to compare datasets but is underutilized in the context of comparing option state snapshots. In this macro, it becomes the centerpiece of an elegant filtering strategy.

MACRO DESIGN

Parameters

- arg - Accepts SAVE, LOAD, or blank (interpreted as SAVE for backward compatibility).
- debug - Optional. When set to 1, prints a list of restored options to the LST.

Key Logic

1. **SAVE (or blank):**
    - Saves the current system options to `__optbase_&depth.__`.
    - Skips save if dataset already exists, issuing a warning.
2. **LOAD:**
    - Saves the current system options to `__optcomp_&depth.__`.

- Uses PROC COMPARE to find differences from the saved baseline.
- Restores only changed options using PROC OPTLOAD.
- Reports restored values to the SAS log.
- Deletes temporary datasets.

Use of PROC COMPARE for Selective Restoration

To determine which SAS options have changed since the original snapshot, the macro executes:

```
proc compare base=__optbase_&depth.__ comp=__optcomp_&depth.__
             out=__optload_&depth.__ outbase outnoequal noprint;
   run;
```

The combination of OUTBASE and OUTNOEQUAL options is critical:

- OUTNOEQUAL filters the output to include only observations where values differ between the base and compare datasets.
- OUTBASE ensures the output dataset retains the structure and variable names that PROC OPTLOAD expects, using values from the baseline (i.e., the saved state we want to restore).

This precisely tailored dataset is then fed into PROC OPTLOAD:

```
proc optload data=__optload_&depth.__;
   run;
```

As a result, only the changed options are reloaded, minimizing disruption and preventing unexpected resets.

Nested Macro Support

By leveraging `&sysmexecdepth`, the macro safely tracks its own level in the macro call stack, allowing multiple calls within nested macros without clashing over temporary datasets.

USAGE EXAMPLES

```
%u_optstore();                /* Save options */
options mprint mlogic;        /* Temporary changes */
%u_optstore(LOAD);            /* Restore only changed */
```

Nested Example:

```
%u_optstore(SAVE);
options compress=yes;

%macro inner;
   %u_optstore(SAVE);
   options mprint;
   %u_optstore(LOAD);
   %mend;

%inner;
%u_optstore(LOAD);
```

BENEFITS

- **Efficiency:** Avoids unnecessary reapplication of unchanged settings.
- **Safety:** Prevents unintended overwrites of session- or site-default options.
- **Debugging Transparency:** Outputs restored settings to the SASLOG and if requested to the output window using PROC PRINT to print the output dataset from PROC COMPARE.
- **Clean Resource Use:** Deletes temporary datasets after use.

RISKS OF FULL OPTLOAD

Reloading options without filtering—i.e., directly feeding the full OPTSAVE dataset to OPTLOAD—can produce unintended results.

A key example is documented in a SAS Communities discussion:
"Why does Proc OPTLOAD change the PAGESIZE to 55?"
https://communities.sas.com/t5/SAS-Programming/Why-does-Proc-OPTLOAD-change-the-PAGESIZE-to-55/m-p/799713#M314471

In that thread, a user observed that PROC OPTLOAD restored all ~300 options, including PAGESIZE, which was reset to 55—even though it had not been explicitly changed. This behavior demonstrates that full reloads ignore intent and can break assumptions about environment consistency.

By using PROC COMPARE to detect and isolate only the modified options, `%u_optstore` avoids these issues entirely.

COMPARISON TO PRIOR WORK

No published SESUG, SUGI, or SAS Global Forum paper (as of mid-2025) appears to describe the use of PROC COMPARE in conjunction with PROC OPTLOAD for selective option restoration. Most published methods rely on either full reload or manual option-by-option resets.

This macro appears to be the first implementation that:

- Automates selective restoration,
- Supports macro nesting safely,
- Avoids risk-prone full reloads.

TRADITIONAL METHOD OF SAVING AND RESTORING SAS SYSTEM OPTIONS

In contrast to the selective restoration provided by the `%u_optstore` macro, the traditional method of saving and restoring SAS system options involves manually capturing the current option values and restoring them after modifications. This is typically done using the GETOPTION function to retrieve the current value of system options and the OPTIONS statement to restore them. While this approach can be effective, it requires manual tracking of each option and lacks the ability to selectively restore only those that have changed.

Example: Saving and Restoring Options

Consider the following SAS code, which demonstrates the traditional method of saving and restoring system options:

```
%let dkricond = %sysfunc(getoption(dkricond, keyword));
%put NOTE: &=dkricond;
NOTE: DKRICOND=DKRICOND=ERROR

%let fmterr = %sysfunc(getoption(fmterr));
%put NOTE: &=fmterr;
NOTE: FMTERR=NOFMTERR

* Change options;
options dkricond=NOWARN;
options fmterr=NOFMTERR;

* Reset options to original values;
options &dkricond &fmterr;
```

In this example:

1. **Save Current Option Values:**
   The %sysfunc(getoption(option-name, keyword)) function is used to capture the current value of specific system options. The KEYWORD argument ensures that the option's value is retrieved as a keyword-value pair.
2. **Modify Options:**
   The options statement is used to change the values of dkricond and fmterr options.
3. **Restore Original Values:**
   After modifications, the original values stored in macro variables are reapplied using the options statement.

While this traditional method is effective, it can be cumbersome when managing multiple options, especially in environments with nested macros or when dealing with large numbers of options that need to be tracked and restored.

Advantages and Limitations

- **Advantages:**
  - Simple and intuitive.
  - Works well for small, straightforward SAS workflows.
- **Limitations:**
  - Requires manual tracking of each option.
  - Does not support selective restoration—if not carefully managed, it can lead to unintended resets of options that were not modified.
  - It is error-prone in complex environments, especially when nested macros or procedures are involved.

In contrast, the `%u_optstore` macro streamlines this process by automating the detection of changed options, restoring only those that have been modified, and avoiding the overhead and risks of full option restores.

CONCLUSION

The `%u_optstore` macro provides a novel, safe, and efficient mechanism for managing SAS system options within reusable code. Its ability to restore only changed options makes it a strong alternative to traditional OPTSAVE/OPTLOAD workflows.

This approach prevents side effects in macro-heavy or production SAS environments, enhances clarity in debugging, and resolves known issues with full-option restores.

REFERENCES

1. SAS Institute Inc. *SAS System Options: Reference*. Cary, NC: SAS Institute Inc.
2. SAS Communities (2022). "Why does Proc OPTLOAD change the PAGESIZE to 55?" https://communities.sas.com/t5/SAS-Programming/Why-does-Proc-OPTLOAD-change-the-PAGESIZE-to-55/m-p/799713#M314471
3. SESUG, SUGI, and SASGF proceedings archive (2010–2024) — no prior similar implementation found.

CONTACT INFORMATION

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

John King
Ouachita Clinical Data Services
501-351-0432
datanull@gmail.com

APPENDIX A: %U_OPTSTORE MACRO SOURCE

```
%macro
   u_optstore
      (
         arg,        /* SAVE or LOAD */
         debug = 0
      );

   %put NOTE: This macro (&sysmacroname) is being called from (%sysmexecname(%sysmexecdepth - 1));
   %local /readonly depth=%sysmexecdepth;

   %put NOTE: (&sysmacroname): &=arg;
   %put NOTE: (&sysmacroname): &=depth;

   %if %qupcase(&arg) eq SAVE or %sysevalf(%superq(arg)=,boolean) %then %do;
      %if %sysfunc(exist(__optbase_&depth.__)) %then %do;
         %put %str(WARN)ING: (&sysmacroname): Data Set __OPTBASE_&depth.__ already exists which is unexpected and implies you have
already called %nrstr(%%)&sysmacroname with the SAVE option.;
         %put %str(WARN)ING: (&sysmacroname): Macro will end with no action taken.;
         %return;
         %end;
      proc optsave out=__optbase_&depth.__; /*Save current system option settings*/
         run;
      %end;
   %else %do;
      %if not %sysfunc(exist(__optbase_&depth.__)) %then %do;
         %put %str(WARN)ING: (&sysmacroname): Data Set __OPTBASE_&depth.__ does not exists which is unexpected and implies you
have NOT called %nrstr(%%)&sysmacroname previously with the SAVE option.;
         %put %str(WARN)ING: (&sysmacroname): Macro will end with no action taken.;
         %return;
         %end;
      proc optsave out=__optcomp_&depth.__; /*Save current potentially changed system option settings to compare with options
previously saved with SAVE option*/
        run;
      proc compare base=__optbase_&depth.__ comp=__optcomp_&depth.__ out=__optload_&depth.__ outbase outnoequal noprint;
         run;
      proc optload data=__optload_&depth.__; /*Load previously saved options that were changed.*/
         run;
      data _null_;
         if not eof then put "NOTE: (&sysmacroname): " 'The following options have been restored to previous settings.';
         do while(not eof);
```

```
         set __optload_&depth.__ end=eof;
         put "NOTE: (&sysmacroname): " (OPTNAME  OPTVALUE) (=);
         end;
      stop;
      run;
   %if &debug eq 1 %then %do;
      proc print;
         run;
      %end;
   proc delete data=__optbase_&depth.__ __optcomp_&depth.__ __optload_&depth.__;
      run;
   %end;

%mend u_optstore;
```