

An Easy, Concise Way to Summarize Multiple PROC COMPAREs Using the SYSINFO Macro Variable

Lex Fennell, PPD, Inc, Wilmington, NC

ABSTRACT

Every time the COMPARE procedure is executed in the SAS® System, the procedure assigns a code to the SYSINFO automatic macro variable. This code represents the result of the PROC COMPARE (ie, whether there were no unequal values, differing attributes, etc.). With the one-time construction of a SAS macro, the SYSINFO codes from multiple compares can be captured and placed in a common data set. This data set, which can contain information such as the timestamp of execution, compare identifier, and SYSINFO code, can be placed in a common location for all compare results. Finally, another SAS program can be created that will use this data set as input, interpret the SYSINFO codes, and output a summary report that is concise and easy to interpret. This report can be kept as a paper-saving alternative for audit documentation.

INTRODUCTION

Ever get tired of looking through pages and pages of PROC COMPARE output to check the results? All that output looks jumbled up anyway. How about all those stacks of paper needed to be kept for audit documentation? Well, you can stop all the nonsense – there is an easy way out. With a little help from the SYSINFO automatic macro variable you can set up a way to view the validation status of all PROC COMPARE output in one clear, concise report. This report can also be used as a paper-saving alternative for audit documentation.

SYSINFO CODES EXPLAINED

The SYSINFO automatic macro variable has a wide range of uses. Even though it is used by many SAS procedures, we are only concerned with its application associated with PROC COMPARE. Every time a PROC COMPARE is executed, SAS assigns a value to SYSINFO. I will refer to this as a SYSINFO code. This SYSINFO code provides information about the result of the comparison as a numeric value having a bitwise nature. The table below shows all the possible results from PROC COMPARE and their representative codes. If more than one result is applicable, the codes from each are added together. Notice the bitwise approach to the SYSINFO codes (more explanation to come).

Bit	Condition	Code	Description
1	DSLABEL	1	Data set labels differ
2	DSTYPE	2	Data set types differ
3	INFORMAT	4	Variable has different informat
4	FORMAT	8	Variable has different format
5	LENGTH	16	Variable has different length
6	LABEL	32	Variable has different label
7	BASEOBS	64	Base data set has observation not in comparison
8	COMPOBS	128	Comparison data set has observation not in base
9	BASEBY	256	Base data set has BY group not in comparison
10	COMPBY	512	Comparison data set has BY group not in base
11	BASEVAR	1024	Base data set has variable not in comparison
12	COMPVAR	2048	Comparison data set has variable not in base
13	VALUE	4096	A value comparison was unequal
14	TYPE	8192	Conflicting variable types
15	BYVAR	16384	BY variables do not match
16	ERROR	32768	Fatal error: comparison not done

COLLECTING ALL SYSINFO CODES IN A COMMON LOCATION

One step needed to facilitate this technique is the one-time construction of a SAS macro that will capture a SYSINFO code and place it in a common SAS data set. This data set can be constructed to contain numerous variables, but perhaps the three most important variables are a compare identifier, an execution timestamp, and the SYSINFO code. The compare identifier should be unique for each PROC COMPARE and should be descriptive. Below is an excerpt from an example program that uses this approach. The three parameters used in this macro are:

libname (a libname pointing to the location of the common SAS data set), dataset (the name of the common SAS data set), and comp_ID (the compare identifier).

```
proc compare base = baseds compare = compds;
run;

%get_sysinfo(libname = common,
              dataset = validation,
              comp_id = ae_dataset
            );
```

Please note that you must remember the `run;` statement to end the COMPARE procedure. SAS will not update the SYSINFO value until the procedure is executed, either by `run;` or by the start of another procedure or data step. If the previous example did not contain `run;` before the macro call, then the macro would begin resolving before the PROC COMPARE was executed. Also, the macro call must immediately follow the PROC COMPARE because other SAS procedures and data steps will reset the SYSINFO code, typically setting the value to zero.

USING THE COMMON DATA SET TO CREATE A SUMMARY REPORT

The last step of leveraging the SYSINFO code from PROC COMPARE is the one-time construction of a SAS program implemented as a stand-alone program or as a SAS macro. This program simply reads the data set containing the compare results, interprets the SYSINFO codes for each row (or each compare identifier), and outputs the results in a clear, concise report. Again, this report can contain many different pieces of data, but the three variables that can be a good starting point include: a compare identifier, a time stamp of execution, and the compare results. Not only is this report easy to read and interpret, but this can replace the pages and pages of PROC COMPARE output that may have to be saved for audit purposes.

The interpretation of the SYSINFO code is really where the power of its bitwise approach is shown. It allows you to specify certain conditions for which to search. If you want to only check to make sure there are no unequal observations, no by groups in one data set but not the other, and no observations in one data set but not the other, you can use the statement:

```
if sysinfo ^= '...0..0000.....'b then error = 'YES';
```

Or, if you only want to know if there are unequal values, you can check with the statement:

```
if sysinfo = '...1.....'b then error = 'YES';
```

And lastly, if you don't want to check for differing attributes, you can use the statement:

```
if sysinfo >= 64 then error = 'YES';
```

CONCLUSION

The SYSINFO code can be a powerful tool if used properly. The technique described in this paper is just one approach that can be adapted to fit your need. Moreover, if you spend a little time developing the appropriate SAS macros and/or programs, the benefits will far outweigh the time spent.

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

Lex Fennell
PPD, Inc.
3151 South 17th St.
Wilmington, NC 28412
Work Phone: 910-772-7043
Fax: 919-654-7890
Email: Lexter.Fennell@wilm.ppd.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.