Flexibility by Design: A Look at New and Updated System Options in SAS® 9.4

Jan Squillace, SAS Technical Support, Cary, NC

ABSTRACT

Base SAS® software's many, customizable options make it an extremely flexible programming language. For example, you can customize the options to make the software suitable for different machine sizes, data file sizes, data types, memory sizes, languages, and locales. Each SAS release adds more flexibility with new options and updated default values for existing options.

This paper discusses new and updated SAS® 9.4 system options that are most likely to affect Base SAS programmers. The discussion covers options that enable you to create very large data sets, accelerate program execution time, ensure the correct date and time in SAS output, prevent truncation of input and output external files and SAS programs, read two-digit years correctly, and save your SAS environment for use in later SAS sessions.

INTRODUCTION

Various types of options are part of the infrastructure that gives the SAS® System such great power and flexibility. Each subsequent release of SAS introduced various types of options that allow the user to specify new capabilities, to specify former behavior, that continue to make SAS more robust and flexible.

SAS has several categories of options, including data set options, LIBNAME statement options, system options, and other categories of options as well. You can generate a list of all SAS options and their default settings by submitting the following OPTIONS procedure:

```
proc options;
run;
```

This paper specifically discusses the SAS 9.4 system options, which modify the behavior of your entire SAS session.

This paper introduces new system options that extend SAS features and improve performance, discusses changed default option values, and presents a few powerful, existing options from earlier releases of SAS that people tend to overlook. Some of the system options also have corresponding data set or LIBNAME statement options of the same name. These corresponding options are noted in the details of the appropriate sections.

Written for advanced programmers and SAS administrators, the paper discusses the following topics and related options:

- creating very large SAS data sets (the EXTENDOBCOUNTER=YES system option)
- accelerating program-execution time by passing the operating system's file cache (the BUFNO=, UBUFNO=, UBUFSIZE=, BUFSIZE=, and VBUFSIZE= system options)
- obtaining the correct date and time in SAS output (the DTRESET | NODTRESET system option)
- preventing the truncation of input and output external files and SAS output (the LRECL= and DMSOUTSIZE= system options)
- reading two-digit years correctly (the YEARCUTOFF= system option)
- saving your SAS environment for use in later SAS sessions (the PRESENV= system option and the PRESENV procedure)

Note: All SAS code samples presented in this paper are run using the following global OPTIONS statement in order to generate helpful messages:

```
options msglevel=I;
```

Flexibility by Design: A Look at New and Updated System Options in SAS® 9.4

CREATING VERY LARGE SAS DATA SETS

As the SAS® System has grown over the years, so has the size of SAS data sets. A data set of several million observations was considered large at the time. Who could possibly imagine the need for data sets larger than that? But now, in the era of big data, data sets have become considerably larger and more complex. As a result, the floating-point, double-precision binary number (a 64-bit number) that is used to hold the number of observations in the data set is inadequate. When the number of observations is not accurate, features of SAS data sets (for example, the POINT= option in the SET statement), indexing and random access) are not available.

SAS 9.3 introduced a new SAS data set structure for 32-bit SAS. This new structure creates room for a longer observation counter of 128 bits, and it depends on the setting of the data set option EXTENDOBSCOUNTER= (alias: EOC=). In SAS 9.3, the default setting is EOC=NO. The structure of the data set is considered SAS®9. However, this structure can no longer be read by SAS® 9.2 or earlier releases.

For SAS 9.4, EXTENDOBSCOUNTER= falls into three SAS option categories: data set options, LIBNAME statement options, and system options. In SAS 9.4, the default setting is EOC=YES. When you need to create a SAS library or SAS data set that is compatible with earlier releases, you can specify EOC=NO in either the LIBNAME statement or as an output data set option.

This option has no effect on 64-bit SAS of any release. This option is important only when you want SAS data sets that are created in 64-bit SAS 9.4 to be read by 32-bit SAS 9.2.

Table 1 summarizes the effects of using the EOC= system option for input and output data sets. For example, a data set that is written by SAS 9..3 on a 32-bit machine and that uses EOC=YES is not readable by SAS 9.2, even when SAS 9.3 and SAS 9.2 reside on identical machines. As another example, consider a data set that is written by SAS 9.4 with the default setting EOC=YES. This data set can be read by SAS 9.3 on a 32-bit machine, but it cannot be read by SAS 9.2 at all.

EXTENDOBSCOUNTER= Option						
Output → / **Input ↓**	**SAS 9.3 32-bit Data Set Option**		**SAS 9.3 64-bit Data Set Option**		**SAS 9.4 64-bit Data Set, LIBNAME, and System Option**	
	EOC = NO	EOC = YES	EOC = NO	EOC = YES	EOC = NO	EOC = YES
SAS 9.2 32-bit	Read access	No access	Read access	No access	Read access	No access
SAS 9.3 32-bit	Read access	Read access	Read access	Read access	Read access	Read access
SAS 9.4 64-bit	Read access	Read access	not applicable	not applicable	not applicable	not applicable

Table 1. Summary of the EOC= System Option's Values and Effects

ACCELERATING PROGRAM EXECUTION TIME

Historically, memory has been one of the most expensive resources on a computer. In the past, programmers prided themselves in writing programs that used the minimum amount of memory to get the job done. One of the ways to reduce memory use was to keep the number of I/O buffers to a minimum. While this method made the program's input and output (I/O) a bit slower, you could use the extra memory for more program instructions.

Fast-forward to the 21st century, and memory is much less expensive and available in larger amounts. Machines have gigabytes (rather than kilobytes) of memory. Now, the emphasis is on minimum elapsed time rather than minimum use of resources. While disk I/O is much faster than it used to be, it is frequently still the bottleneck for faster processing. Enter file caching!

File Caching

In general, file caching improves the throughput of disk I/O. Some form of file caching exists in all Microsoft Windows and UNIX or LINUX operating environments on which SAS runs. The *file cache* is an area in memory where previously read or written data is stored. When an additional request for I/O is detected, the file cache is examined by the operating system to see if the data already exists there. For small data sets (less than 2 GB), this technique prevents unneeded disk I/O and increases the amount of work that is done by the system. The number of buffers does not materially affect the execution time when file caching is used.

When large data sets (more than 2GB) exceed the size of the file cache, the movement of data into and out of the file cache becomes a bottleneck and degrades the performance of the system.

SAS Systems Options for Improving Performance in Specific Operating Environments

When you have a bottleneck that degrades your program's performance, how do you improve that performance? Essentially, you need to bypass file caching for SAS data sets that are larger than 2 GB. The following sections explain how to bypass the file cache for various operating environments in which SAS runs.

Microsoft Windows operating environments: Use the SGIO option (which indicates the scatter-read/gather-write feature or scatter/gather) to bypass file caching. The conditions for scatter/gather include an 8 KB, multiple page size for 64-bit systems and sequential processing. For the complete requirements, see the section "SGIO System Option: Windows" in the *SAS® Companion for Windows*. (**support.sas.com/documentation/cdl/en/hostwin/64812/PDF/default/hostwin.pdf**) When a data set does not meet the requirements for SGIO processing, the SGIO option is ignored.

The following code illustrates the use of the SGIO option to bypass file caching:

```
%macro mymac;
data tmp (sgio=yes);
   array arnum (*) var1-var50;
   do I = 1 to &obsmax;   /* Create observations. */
      x=I;
      y=i*1.5;
      z=i*.5;
      do j = 1 to hbound(arnum);
         arnum(j) = x + y;
      end;
     output;
   end;
run;

data newtmp;
   set tmp (sgio=yes);
run;

%mend;

*options bufsize=9k;  /* Changing the buffer size disables SGIO processing. */
options msglevel=I;
%let obsmax=10**3;
%mymac;
run;
```

When SGIO is enabled, the following informational message appears in the output:

```
NOTE: SGIO processing active for file WORK.TMP.DATA.
```

When SGIO is not active, this message is generated:

```
NOTE: SGIO processing disabled for file WORK.TMP.DATA since the page size does
not meet the criteria. See the Windows Companion for more details.
```

UNIX and LINUX operating environments: The LIBNAME engine option ENABLEDIRECTIO bypasses file caching for an entire SAS library. You can also use the USEDIRECTIO= option to bypass file caching for individual data sets after file-cache bypassing is set for the entire library. This technique is particularly useful when both small and large SAS data sets reside in the same SAS library, as shown in the following example:

```
   /* Enable direct I/O for the entire library. */
libname mylib '/u/user/library-directory-name' enabledirectio;

   /* Set direct I/O for individual data sets within the library. */
data mylib.newdataset (usedirectio=yes);

   set mylib.olddataset (usedirectio=yes);

run;
```

Note: When you have a library in which all of the data sets are large, you specify USEDIRECTIO in the LIBNAME statement and omit the data set option, as shown below:

```
libname newlib '.' enabledirectio usedirectio=yes;

data newlib.ds1;
   set sasuser.class;
run;
```

IBM z/OS operating environment: The z/OS operating system does not use file caching for SAS bound libraries. (For details about SAS bound libraries, see the *SAS® 9.4 Companion for z/OS*. (**support.sas.com/documentation/cdl/en/hosto390/64786/PDF/default/hosto390.pdf**) There are no SAS options to work with. All input and output is direct-to-disk I/O.

SAS Data Set Buffering for All Operating Environments

When you are not using file caching, you can use excess memory to speed up I/O performance. Because determining the optimum value takes several iterations of running the SAS code and evaluating the results, tuning with these options is most beneficial for programs that will be run many times on similar data volumes.

Note: The techniques described in this section apply to all of the operating systems (Window, UNIX, Linux, and z/OS), but they are particularly useful for z/OS because z/OS does not have file caching.

The following sections discuss techniques for speeding up I/O performance using the BUFNO=, BUFSIZE=, VBUFSIZE=, UBUFNO-, and UBUFSIZE= options. These system options give you more control over the number and size of buffers for data sets, views, and utility files.

Using the BUFNO= and BUFSIZE= System Options to Control Buffer Number and Size for SAS Data Sets

The BUFNO= and BUFSIZE= options are existing systems options that specify the number of buffers and buffer size for data sets in Base SAS® software.

You can run a SAS program that shows the effect on elapsed time and CPU time when you increase the number of buffers. The following program shows this effect for 1, 10, 100, and 1000 buffers for a SAS data set of 10 million observations:

```
%macro mymac;
data tmp (sgio=yes);
   array arnum (*) var1-var50;
   do i=1 to &obsmax;     /* Create observations. */
      x=i;
      y=i*1.5;
      z=i*.5;
      do j=1 to hbound(arnum);
         arnum(j) = x + y;
      end;
      output;
   end;
run;

data newtmp;
   set tmp (sgio=yes);
run;
%mend;

options msglevel=I;
%let obsmax=10**7;
options bufno=1;
%mymac;
options bufno=10;
%mymac;
options bufno=100;
%mymac;
options bufno=1000;
%mymac;
run;
```

Table 2 shows the output that is generated by the previous code.

10 Million Observations		BUFNO=1	BUFNO=10	BUFNO=100	BUFNO=1000
write	real time	72.21	70.83	35.06	20.22
	CPU time	16.08	12.75	12.32	12.16
read	real time	98.13	76.02	50.75	51.61
	CPU time	10.35	8.22	8.42	8.72

Table 2. Effect on Elapsed Time and CPU Time When You Use More Buffers for Large SAS Data Sets without File Caching

The default setting is BUFNO=1 because file caching is the default value. When file caching is disabled, increasing the number of buffers improves performance until you run out of memory and start paging the least recently used pages of memory to disk. When paging occurs, performance worsens dramatically as a result of extra disk I/O. In Table 2, above, you can see that increasing the number of buffers improves the elapsed time until the number of buffers reaches 1000. The optimal number of buffers for this particular workload is 900.

Using the VBUFSIZE= System Option to Control the Size of Buffers for DATA Step Views

In earlier releases of SAS, you can control the number of observations that are presented by a DATA step view by using the OBSBUF= data set option. You have to specify this option each time the view is used. The default value for the OBSBUF= option is the number of observations that fit into 32K of memory. You can specify a larger value when enough real memory is available.

However, in SAS 9.4, the new VBUFSIZE= option gives you the ability to set the default size of the view buffer for an entire session rather than each time the view is used.. The default value for this option is 65536 (64K).

NOTE: The VBUFSIZE= option does not apply to SQL views.

When a DATA step view is processed, two SAS tasks are started, one task for the procedure or DATA step using the view (Process 1 in Table 3) and the other task for resolving the view (Process 2 in Table 3). Both of these tasks share the view buffer that is defined by the VBUFSIZE= option. First, the buffer is filled with as many observations as it can hold. Then task switching (back to the calling task) occurs, where the observations are used until the view buffer is empty. When the view buffer is small, frequent task switching can cause unnecessary overhead. The best size for the view buffer is one that is large enough to hold as many observations as possible, but not so large that it causes an out-of-memory condition. The number of observations that the buffer can hold is dependent on the width of the observation as well as the size of the view buffer.

The following code uses the VBUFSIZE= option to generate a table that shows the correlation between various buffer sizes and the time elapsed (real time) for processing:

```
%macro mymac (vbufsize=);
options vbufsize=&vbufsize;
run;

proc options option=vbufsize value;
run;

data new;
   set newtmp;
run;

%mend;
options msglevel=I;
%let obsmax=10**7;   /* No more disk space. */

data tmp;
   array arnum (*) var1-var50;
   do I=1 to &obsmax;    /* Create observations. */
      x=I;
      y=i*1.5;
      z=i*.5;
```

(code continued)

```
      do j=1 to hbound(arnum);
         arnum(j) = x + y;
      end;
      output;
   end;
run;

data newtmp/view=newtmp;
  set tmp ;
  if mod(x,2) = 0;
run;

%mymac(vbufsize=128K);
%mymac(vbufsize=256K);
%mymac(vbufsize=512K);
%mymac(vbufsize=1024K);
run;
```

The code above generates Table 3:

ProcessingTime		VBUFSIZE=128K	VBUFSIZE=256K	VBUFSIZE=512K	VBUFSIZE=1024K
Process 1	Real time	99.53	72.36	64.67	61.25
	CPU	8.61	8.43	8.73	8.48
Process 2	Real time	99.69	72.50	64.76	61.38
	CPU	8.62	8.47	8.74	8.50

Table 3. Correlation between View Buffer Size and Processing Time (Real and CPU Time)

Using the UBUFNO=, UBUFSIZE=, and DATAPAGESIZE= System Options to Control Buffer Number and Size for Utility Files

In releases earlier than SAS 9.4, you have no control over the number and size of utility buffers. In those earlier releases, SAS uses an algorithm to determine these values.

In SAS 9.4, you can use the UBUFNO= and UBUFSIZE= options specify different numbers of buffers and buffer size for sequentially accessed utility files that are used in the SAS SORT procedure.

The following code uses the UBUFNO= option to generate a table that provides real (elapsed) time values for various numbers of utility buffers. This code also uses the SGIO=YES option to turn off file caching.

```
%macro mymac (ubufno=0);
options ubufno=&ubufno;
run;

proc options option=ubufno value;
run;

proc sort data=tmp(sgio=yes)
   out=tmp1(sgio=yes);
   by x;
run;
%mend;

options msglevel=I;
%let max=7;
%let obsmax=10**&max;

data tmp;
   do I=1 to &obsmax;   /* Create observations. */
      x=int(ranuni(1234)*10**&max);
      y=i*1.5;
      z=i*.5;
      output;
   end;
run;

proc contents data=work.newtmp;
run;

%mymac(ubufno=1);
%mymac(ubufno=5);
%mymac(ubufno=10);
%mymac(ubufno=15);
%mymac(ubufno=20);
run;
```

Table 4, generated by the previous code, illustrates the correlation between then number of buffers and processing time. The CPU processing time is roughly the same. However, the real processing time for PROC SORT is reduced as you increase the number of utility buffers.

Processing Time	UBUFNO=1	UBUFNO=5	UBUFNO=10	UBUFNO=15	UBUFNO=20
Real time	23.10	22.09	20.62	19.74	19.25
CPU	11.61	11.72	11.36	11.46	11.46

Table 4. Correlation between the Number of Buffers and Processing Time (Real and CPU Time)

The following code uses the UBUFSIZE= option to generate a table that provides real (elapsed) time values for various utility buffer sizes. This code also uses the SGIO=YES option to turn off file caching.

```
%macro mymac (ubufsize=);
options ubufsize=&ubufsize;
run;

proc options option=ubufsize value;
run;

proc sort data=tmp(sgio=yes)
          out=tmp1(sgio=yes);
   by x;
run;
%mend;
%let max=7;

options msglevel=I;

%let obsmax=10**&max;
data tmp;
   do I=1 to &obsmax;   /* Create observations. */
      x=int(ranuni(1234)*10**&max);
      y=i*1.5;
      z=i*.5;
      output;
   end;
run;

%mymac(ubufsize=128K);
%mymac(ubufsize=256K);
%mymac(ubufsize=512K);
%mymac(ubufsize=1024K);
run;
```

Table 5, generated by the previous code, illustrates the correlation between buffer size and processing time. The CPU processing time is roughly the same. However, the real processing time for PROC SORT is reduced as you increase the size of the utility buffers.

Processing Time	UBUFSIZE=128K	UBUFSIZE=256K	UBUFSIZE=512K	UBUFSIZE=1024K
Real time	22.74	20.46	19.97	19.21
CPU	11.45	11.09	11.67	10.95

Table 5. Correlation between Buffer Size and Processing Time (Real and CPU Time)

SAS 9.4 also includes the DATAPAGESIZE= option, which controls the algorithm that determines optimal buffer size when the UBUFSIZE= or BUFSIZE= options are set to **0**. Valid values for the DATAPAGESIZE= option are COMP93 and CURRENT. The CURRENT value specifies that the buffer size should be determined by optimization processes for the current SAS release, while COMPAT93 specifies that the buffer size should be determined by SAS 9.3 optimization processes.

For example, supposed that the best buffer size for SAS 9.4 (DATAPAGESIZE=CURRENT) is too large for the hardware. In that case, you can use the option DATAPAGESIZE=COMPAT93 to force the use of the algorithm that is used for SAS 9.3 calculations.

OBTAINING THE CORRECT DATE AND TIME IN SAS OUTPUT

When a SAS session is open overnight, the output does not update the date and time from when the session started to the current date and time. So a common question that SAS Technical Support receives is how to change that date to the current date and time. The solution is a simple option reset using the DTRESET | NODTRESET system option, as shown in the following OPTIONS statement:

```
options dtreset;  /* Current date and time are printed from this */
                  /* point forward in your session.              */
```

In this case, DTRESET updates the date and time for each procedure where the date and time are shown. The default value is NODTRESET, which maintains the initial date and time throughout the SAS execution.

PREVENTING TRUNCATION OF INPUT AND OUTPUT EXTERNAL FILES AND SAS OUTPUT

In releases of SAS prior to 9.4, extremely large volumes of output can mean that you experience truncation of that output, all based on the size of the SAS buffer and the number of lines that the output window can hold. In releases prior to 9.4, SAS has a small buffer (256 bytes) that is used for external-file input and output. In SAS 9.4, the new default value for the size of the external-file I/O buffer is 32 KB. This size is specified in the LRECL= system option.

With regard to how many lines the output window can hold, the default size is 999999 lines in releases earlier than SAS 9.4. In SAS 9.4, the new default size, which is specified in the DMSOUTSIZE= system option, has increased to 2147483647 lines of output.

The following sections provide more details about the system options (LRECL= and DMSOUTSIZE=) that you can use to avoid truncation.

Using the LRECL= System Option to Avoid Truncation of Input or Output

In releases of SAS earlier than 9.4, programmers might see the following message in their SAS Log:

```
One or more lines were truncated.
```

That message indicates that SAS either did not read the entire input line or that your output file lost some of the data on the right side. But what really happens is that the default buffer size for the external file is too small to hold the entire row, so some of the data is dropped. However, this problem is eliminated in SAS 9.4 with the change to the default LRECL= value.

Flexibility by Design: A Look at New and Updated System Options in SAS® 9.4

Previous releases of SAS use a default input and output row size of 256 bytes. This value works when most rows are 80 bytes or less. But the amount of data that users want to transfer has grown over time. As a result, SAS Technical Support receives many calls from customers who are experiencing truncated input or output files.

In prior releases, this issue is solved by adding an LRECL= system option that specifies a longer maximum logical-record length (32 KB) in either the INFILE or the FILE statement.

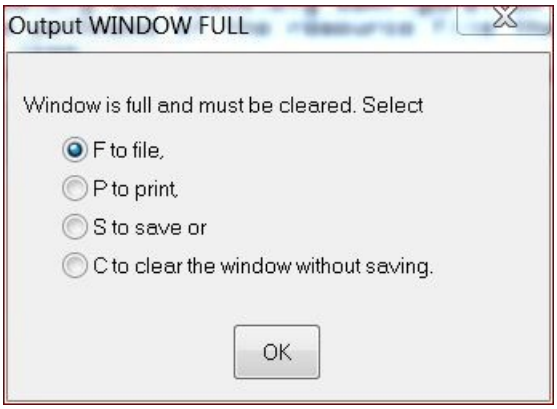
```
options lrecl=32767;  /* This LRECL= value applies to input and output. */

infile 'your-external-input-file' lrecl=32767;  /* This LRECL= value        */
                                                /* applies only to the file */
                                                /* is in this statement.    */
```

In SAS 9.4, you no longer need to add the LRECL= option to the FILE or INFILE statement because the default value is larger.

Using the DMSOUTSIZE System Option to Expand the Space in the SAS Display Manager Window Queue

You write and run code in the SAS windowing environment. When a large amount of SAS output is generated and the SAS Output window full, SAS displays a dialog box similar to the following that asks what you want to do with the window contents.



Sometimes large programs take a long time to run, and you might step away to do something else while it is running. So it can be frustrating to return, only to see this dialog box and know that you still have more time to wait because you were not there to make a selection and click **OK**.

However, now you avoid that situation by using the DMSOUTSIZE system option to enable the output window to hold many more lines. This option affects only the output window, and it specifies the number of data rows that the output window can display. As mentioned earlier, previous releases of SAS can only hold a maximum of 999999 rows. In SAS 9.4, the default maximum value is increased to 2147483647 rows.

For example, the following SAS command uses the DMSOUTSIZE option to specify a size of 1024 KB:

```
sas -dmsoutsize 1024K;    /* Command line for Windows or UNIX    */
                          /* that specifies a desired value.   */
```

As an alternative, you can turn off the SAS Output window and only use the SAS Results window to view your output. To do this:

1. Select **Tools ► Options ► Preferences**.
2. Click the **Results** tab.
3. Clear the **Create listing** check box.
4. Click **OK**.

Note: The corresponding system option for the SAS Log window is the DMSLOGSIZE option. The default value is unchanged from SAS 9.3.

Flexibility by Design: A Look at New and Updated System Options in SAS® 9.4

HANDLING UNINITIALIZED VARIABLES

Previous releases of SAS issue a note when an uninitialized variable is used on the right side of an assignment statement or is written to an output data set. The only way to eliminate a note that results from use of uninitialized variables is to correct the program. The most common reason for uninitialized variables is a misspelling in a variable name. An uninitialized variable can also occur when a variable is dropped from a data set option in a SET or MERGE statement and then that variable is used in a calculation.

In SAS 9.4, you can either change the importance of uninitialized variables (by specifying that the messages should be warnings or errors) or suppress the note altogether.

Consider the following program and output.

```
1       /*uninit.sas */
2
3    data _NULL_;
4       x1=2;
5       y=xl;
6       put _all_;
7    run;

NOTE: Variable xl is uninitialized.
x1=2 y=. xl=.  _ERROR_=0 _N_=1
NOTE: DATA statement used (Total Process time):
      real time           13.61 seconds
      cpu time            0.15 seconds
```

Can you find the problem in the DATA step? Why does variable Y have a missing value? You would assume that Y has a value of 2 based on the program.

In this case, the variable X1 is defined with a value of 2. However, the variable Y is assigned the value of Xl (X plus the lowercase version of the letter L), so the variable name that is assigned to Y is misnamed. The font that is used for the log text makes it very difficult to see this difference. What gives it away is that two variables (X1 and Xl) are shown side by side in the log output and in the NOTE text.

SAS 9.4 provides a new option, VARINITCHK=, that you can use to change the status of the message from NOTE to WARNING or ERROR. While the message remains the same, changing the status of the message causes SAS to issue a return code of either 4 (for a warning) or 8 (for an error) so you have more information about the problem.

To specify a warning, use the following OPTIONS statement:

```
options varinitchk=warn;
```

To specify an error, use this OPTIONS statement:

```
options varinitchk=error;
```

One other option is to completely suppress a message, as shown in this statement with the value NONOTE:

```
options varinitchk=nonote;
```

Caution: Use the NONOTE setting carefully because you will not know that you have an uninitialized variable if the message is suppressed.

READING TWO-DIGIT YEARS CORRECTLY

In the past, two-digit years were common in data processing to save space in data sets because disk space was expensive. As a result, four-digit years were commonly dropped to the last two digits because people always knew what century it was. Fast forward to the 21st century. Disk space is much less expensive and is available in larger quantity than ever before. More available disk space means that you can now store the full four-digits of a year in character format.

Flexibility by Design: A Look at New and Updated System Options in SAS® 9.4

SAS date values are stored as the number of days since 01JAN1960. The underlying numeric value always includes the century. The YEARCUTOFF= option is useful to establish the context when you are reading input that contains two-digit years. That is, the option helps determine the first two digits (century) in a four-digit year.

But reading the year number with only the last two digits can present a problem in knowing what the correct century (first two digits) is. For example, suppose you have data for people who are eligible for Medicaid. If a person's birth date is 10/09/10, then you undoubtedly know that the year is 1910. Or, if you are working with data about children under 12, and you see that same date (10/09/10), you know that the year is 2010. It is up to the programmer to determine the correct century. In SAS 9.4, you can use the YEARCUTOFF= system option to help you make that determination.

The following code uses the YEARCUTOFF= system option to generate a table that shows the effect of the option value when you are reading two-digit years:

```
data test;
   infile datalines truncover;
   input indt $char10.;
   datalines;
1/20/10
1/20/20
1/20/25
1/20/28
1/15/35
1/10/55
1/5/60
1/30/75
1/31/90
1/12/95
;

%macro cutoff (yr4);
options yearcutoff=&yr4 nodate;

data out&yr4;
   set test;
   outdt = input(indt,mmddyy10.);
   format outdt mmddyy10.;
   yrcut= getoption('YEARCUTOFF');
run;
proc print data=out&yr4 noobs;
   title "YEARCUTOFF demonstration";
   var yrcut indt outdt;
run;

%mend;
ods pdf file='c:\temp\cutoffeffect.pdf' columns=4 contents=no;
```

(code continued)

```
%cutoff(1900);
run;
%cutoff(1920);
run;
%cutoff(1926);
run;
%cutoff(1955);
run;
ods pdf close;
```

Table 6 demonstrates how the value of the YEARCUTOFF= option affects the result when you are reading two-digit years. The first column is the date that is being read with various years. For releases earlier than SAS 9.4, the default setting is YEARCUTOFF=1920. In SAS 9.4, the default setting is YEARCUTOFF=1926. The last two columns demonstrate higher values for the YEARCUTOFF= option.

Input Date	YEARCUTOFF= System Option			
Two-Digit Year	**1900**	**1920**	**1926**	**1955**
1/20/10	01/20/1910	01/20/2010	01/20/2010	01/20/2010
1/20/20	01/20/1920	01/20/1920	01/20/2020	01/20/2020
1/20/25	01/20/1925	01/20/1925	01/20/2025	01/20/2025
1/20/28	01/20/1928	01/20/1928	01/20/1928	01/20/2028
1/15/35	01/15/1935	01/15/1935	01/15/1935	01/15/2035
1/10/55	01/10/1955	01/10/1955	01/10/1955	01/10/1955
1/5/60	01/05/1960	01/05/1960	01/05/1960	01/05/1960
1/30/75	01/30/1975	01/30/1975	01/30/1975	01/30/1975
1/31/90	01/31/1990	01/31/1990	01/31/1990	01/31/1990
1/12/95	01/12/1995	01/12/1995	01/12/1995	01/12/1995

Table 6: The Effect of the YEARCUTOFF= System Option on Two-digit Years

SAVING THE SAS® ENVIRONMENT FOR USE IN LATER SAS® SESSIONS

The hardest part of many projects is assembling all the data that you need to solve the problem. Each time that you write a section, you find another SAS library that needs to be made available to your program. The LIBNAME statements that are used to define these libraries are scattered through an entire SAS session, so they sometimes difficult to locate and collect.

SAS 9.4 can help by recording all of the SAS library assignments, along with global macro variable definitions, current option settings, and macros that you have defined in the session.

SAS 9.4 provides the PRESENV= system option as a way to preserve your SAS environment for later use, and the PRESENV procedure stores the saved SAS environment.

To save and store your SAS environment for later use, follow these steps:

1. Submit the following OPTIONS statement to activate the PRESENV= option so that SAS starts saving environment:

```
options presenv;
```

This OPTIONS statement saves the following items:

- global SAS statements (for example, the FILENAME, LIBNAME, TITLE, and FOOTNOTE statements)
- compiled macros in the SAS Work library
- temporary formats in the SAS Work library
- option settings in your session

2. At the completion of the session that you want to preserve, submit the following PRESENV procedure:

```
libname outdata 'c:\temp\outdata';          /* Use your SAS library. */
filename outcode 'c:\temp\outpgm\outpgm.sas'; /* Use your text file. */
proc presenv permdir=outdata
     sascode=outcode
     show_comments;
run;
```

In this code:

- The SHOW_COMMENTS argument, which is optional, causes all of the generated SAS program statements to appear in your log. This feature is helpful when you want to see what information is saved and where that information is located.
- The PERMDIR= option is assigned a libref that points to the SAS library where you want the contents of your Work library to be written. The data set that is referenced by the libref is emptied (using the DATASETS procedure) before the WORK library contents are copied into the library.

3. When you want to start a new SAS session, submit the following code to restore the former environment:

```
options nopresenv;
   %include 'c:\temp\outpgm\outpgm.sas';
run;
```

Caution: Although the entire contents of the Work library are copied to the data library referenced in the PERMDIR= option, only the SAS data sets (including their indexes) are restored back to the Work library in the refreshed session. SAS views **are NOT restored** to the WORK library. Instead, they remain available in the library that is referenced by the PERMDIR= option.

Note: PROC PRESENV saves any global resource (libraries, macros, and so on) that are allocated with SAS statements. However, if you allocate a SAS library by selecting **File ► New ► Library Specification window**, you need to select the check box labeled **Enable at startup** in order for the library to be available in successive sessions because PROC PRESENV does not preserve that library definition.

CONCLUSION

As the output of PROC OPTIONS shows, many SAS 9.4 system options are available to SAS users. You can use the flexibility of a number of these options to customize the behavior of your SAS data and programs. This discussion covered a number of examples of using SAS 9.4 system option to customize behavior:

- changing the number of buffers in SAS data sets, views, and utility files by using the BUFNO=, UBUFNO=, and VBUFNO= system options
- changing the size of the buffers with the BUFSIZE=, UBUFSIZE=, and VBUFIZE= system options
- modifying the layout of a SAS data set by using the EXTENDOBSCOUNTER= system option

(list continued)

- changing the volume of output that is permitted for your SAS session by using the DMSOUTSIZE= and, DMSLOGSIZE= system options
- updating the date and time of your SAS output by using the DTRESET= system option

You also learned how to use the PRESENV= system option, PROC PRESENV, and the %INCLUDE statement to save and restore the contents of an established SAS session for later use.

Now that you have seen the flexibility and power that is available just through the several options discussed in this paper, SAS Technical Support encourages you to explore the full range of SAS system options. In doing that, you might just find a new way to use SAS more efficiently, making your job easier and your time more productive!

REFERENCES

Ihnen, Leigh and Michael R. Jones. 2009. "Improving SAS® I/O Throughput by Avoiding the Operating System File Cache." *Proceedings of the SAS Global Forum 2009 Conference*, Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings09/327-2009.pdf**.

SAS Institute Inc. 2008. "Achieving Better I/O Throughput Using SGIO in the Microsoft Windows Environment." Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/IOthruSGIO.pdf**.

RECOMMENDED READING

SAS Institute Inc. 2013. *SAS® 9.4 Companion for Windows*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/hostwin/64812/PDF/default/hostwin.pdf**.

SAS Institute Inc. 2013. *SAS® 9.4 Companion for UNIX Environments*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/hostunx/64815/PDF/default/hostunx.pdf**.

SAS Institute Inc. 2013. *SAS® 9.4 System Options: Reference*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/lesysoptsref/64789/PDF/default/lesysoptsref.pdf**.

SAS Institute Inc. 2013. "Techniques for Optimizing I/O" in *SAS® 9.4 Language Reference: Concepts*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/lrcon/64801/PDF/default/lrcon.pdf**.

ACKNOWLEDGMENTS

Thank you to Dan Squillace.

CONTACT INFORMATION

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

Jan Squillace
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: **support@sas.com**
Web: **support.sas.com**