Paper 308

Year 2000 Issues for SAS Users

Mike Kalt, SAS Institute Inc., Cary, NC
Rick Langston, SAS Institute Inc., Cary, NC

ABSTRACT

This paper addresses the most frequently asked questions about Year 2000 compliance for the SAS® System. Topics include:

- Is the SAS System Year 2000 compliant?
- What should I look for when testing my SAS applications for Year 2000 compliance?
- What resources are available to test my SAS applications for compliance?
- Do I need special licensing in order to do Year 2000 testing?
- How do I correct problems in my data and SAS applications?

We assume that you have a basic knowledge of the SAS system, and are familiar with general aspects of the "Year 2000 problem”.

INTRODUCTION: SHOULD I PANIC?

If you are responsible for maintaining SAS programs and applications, you may be wondering whether you can safely book that New Year’s Eve cruise on December 31, 1999 or whether you will need to be at your desk with a large bottle of antacid as the clock approaches midnight. Your location on New Year’s Eve largely depends on the answers to the following questions:

- Is SAS software compliant?
- Are your SAS applications compliant?
- Are your data compliant?

In the course of this paper we will address all three issues. As we will see, your major concerns should be the last two of these items.

IS SAS SOFTWARE YEAR 2000 COMPLIANT?

Current versions of SAS software on most operating systems are Year 2000 compliant. For detailed and current information on the compliance status of specific SAS software releases and SAS system products, refer to our Year 2000 compliance statement at http://www.sas.com/techsup/download/technote/y2kcomp.html.

ARE YOUR SAS APPLICATIONS COMPLIANT?

Because most SAS applications are built using such components as the DATA step, Screen Control Language (SCL), or Interactive Matrix Language (IML), the fact that SAS software itself is compliant does not mean that user applications developed using the software are necessarily compliant. Although SAS software provides the tools for handling date values to meet all of the compliance standards above, it is up to the user to ensure that those tools are correctly used in the applications they develop.

OVERVIEW OF SAS DATE HANDLING

To evaluate your SAS code for Year 2000 compliance (and to understand why SAS software itself is compliant), it is helpful to understand how SAS software processes date values.

The SAS System stores date values as an offset in days from January 1, 1960, and datetime values (such as 23FEB1998:16:30) as an offset in seconds from the first second of January 1, 1960. SAS users can convert external data to or from SAS date values and datetime values by the use of various informats, formats, and functions (IFFs). These IFFs all accept two-digit years as well as four-digit years. For example, if your input file contains a date value of '07/27/1998', you can read it using the MMDDYY10. informat, while a field containing '07/27/98' can be read using the MMDDYY8. informat. Similarly, dates with four- and two-digit years can be written to external files using the MMDDYY10. and MMDDYY8. formats respectively.

Starting with Release 6.06 of the SAS System, the YEARCUTOFF= option can be used to specify the century to be associated with years containing only two digits. For example, YEARCUTOFF=1950 means that two-digit years 50 through 99 correspond to 1950 through 1999, while two-digit years 00 through 49 correspond to 2000 through 2049. The user can specify whatever YEARCUTOFF= value is appropriate for the application, and the site can decide what the default value for YEARCUTOFF= should be.

Note that the default value of the YEARCUTOFF= option in Version 6 is 1900, which means that all 2-digit years are assumed to be in the 1900’s. If you have 2-digit year dates in your data greater than 2000, you should specify a YEARCUTOFF= value whose 100-year range will include the latest date in your data. The default value for the YEARCUTOFF= option in Version 7 is 1920.

For more information on the YEARCUTOFF= option, refer to *SAS Language: Reference, Version 6, First Edition,* pp. 128-131. Also, a Technical Tip on the YEARCUTOFF= option can be obtained at http://www.sas.com/techsup/download/technote/ts597.html

LIMITATIONS OF SAS DATE HANDLING

Although the SAS System's date handling capabilities are designed to address Year 2000 compatibility, they are only effective if they are actually used. If your application reads in a date value with a standard numeric or character format instead of a date format, Year 2000-related errors can still result. For example, if your input data contains the value 99360 to represent the 360th day of 1999 and 00220 to represent the 220th day of 2000, and you read them in with a standard numeric informat, the 360th day of 1999 will be assumed to be after the 220th day of 2000 when the dates are compared. However, if you read them in using the JULIAN5. date informat (and an appropriate value for the YEARCUTOFF= option), they are each stored as the number of days since January 1, 1960, and any calculations, comparisons, or sorts using the dates will be accurate.

Similarly, if you read in a date field such as 95/05/01 or 950501 as a character variable, and use the variable as a sort key (a not uncommon occurrence), observations will be sorted in incorrect order if the dates cross the Year 2000 boundary. However using a date informat and the YEARCUTOFF= option will result in correct processing.

Even if you are not reading in complete date values, you can minimize compliance problems by converting any date-related data to SAS date values. For example, suppose your input data contains a two digit year field, with no month or day information.

If you attempt to compare values of 98 and 02 (where the latter represents the year 2002), you will obtain incorrect results. However, you can convert the year to a complete date using the following code:

```
data;
   input yr 2.;
   date=mdy(1,1,yr);
   .
   .
   .
```

The variable DATE will be stored as a SAS date value of January 1 of the corresponding year, and any subsequent manipulation or comparison of dates will return correct results.
Whether or not you are using SAS date values, there are some circumstances where you must take additional steps to ensure proper date handling. These include:

- If the range of date values for a variable cover more than a 100-year span, using the YEARCUTOFF= option will not ensure unambiguous interpretation of a two digit year. (For example, if your input years can run from 1880 through 2010, you cannot specify a single YEARCUTOFF= value that would interpret the date 24JUN05 correctly. For data covering more than a 100 year span, you must either use 4 digit years in the data, or use conditional logic in a DATA step to interpret them appropriately.

- The YEARCUTOFF= value is used for all date variables in the step for which the option is in effect. If you are processing multiple date variables in a single step and each of them requires a different YEARCUTOFF= value, you need to split the processing across multiple DATA steps, with each DATA step using a different YEARCUTOFF= value.

- If you are reading data from an external database using a SAS/ACCESS® view descriptor or PROC SQL, the YEARCUTOFF= option has no effect if the external database represents date values as an offset from a certain date. For example, suppose you enter the date '02/17/05' into a Microsoft Excel spreadsheet, and then use SAS/ACCESS software to convert the spreadsheet into a SAS data set. Since Excel stores the date as an offset from January 1, 1900, SAS/ACCESS software takes that value and converts it to the corresponding SAS date value. If Excel interpreted the date as being in 1905, it would be stored as February 17, 1905 in the SAS data set. If Excel interpreted the date as being in 2005, it would be stored as February 17, 2005 in the SAS data set, regardless of the value of the YEARCUTOFF= option.

  Note that this does not apply to many mainframe databases such as IMS or ADABAS, which store date representations as character strings or simple numeric values (such as 980204). SAS/ACCESS interfaces for these databases and PROC SQL apply date informats when reading in date values, and the YEARCUTOFF= option does have an effect in this case.

- Output to external files or written reports may not be interpreted correctly if 2 digit years are used in output date formats. For example, if your SAS program reads in a field containing 26JUL2000 with a DATE9. informat and then writes the data to an external file using a DATE7. format (26JUL02), the output may be interpreted incorrectly by subsequent programs that read it.

Note also that the YEARCUTOFF= option has no effect on date values already stored in a SAS data set. For example, suppose you specify YEARCUTOFF=1900 and read in the field 24AUG01 using a DATE7. informat. The date is stored internally as August 24, 1901. If you subsequently specify YEARCUTOFF=1920 and display the date using the DATE9. format, it will still display as 01AUG1901.

YEAR 2000 TEST CRITERIA

There are several different aspects of Year 2000 compliance for which you need to test your SAS applications. They are:

Will my application run and correctly interpret the system date in the Year 2000 and after? This should not be a problem with most SAS applications, since the SAS System will run and return a correct system date with a system clock set on or after January 1, 2000. (Note that the SAS System must be licensed for the period in which you are running--see "Testing With a Future Date" below for more information on testing SAS applications with system dates in the future.)

Will my application correctly process data containing date values? Some specific aspects of compliance that fall into this category include:

- All calculations involving date values (such as duration, comparisons of dates, determination of days of the week) produce correct results.
- Input data containing either 2-digit or 4-digit years are interpreted correctly.
- Dates that might be ambiguously interpreted with respect to century are written to external files using 4 digit years (or a century indicator)

.
Are date values that my application stores in SAS data sets represented in a format that ensures compliance? To meet this requirement, all date-related values used in your SAS applications should be stored as SAS date or datetime values, not as character variables or ordinary numeric variables.

Are non-SAS components of my application compliant? Many SAS applications are part of larger integrated systems containing applications written in other languages. Even if your SAS application writes out all dates using 4-digit years, your overall application may still be non-compliant if the application that reads in the file created by SAS only reads in the two-digit portion of the year.

TESTING YOUR SAS APPLICATIONS

The following sections describe some specific techniques for testing your SAS programs and data for Year 2000 compliance. ***Remember that there is no "silver bullet" or single tool that can be used to test your SAS applications for compliance***. To be thorough, you must employ several different types of tools and approaches, each designed to detect different types of non-compliance. Some of the approaches are:

- Testing with a future date
- Testing with future data
- Static Testing
  - Visual code review
  - Automated search tools
- "Run-time" checking

TESTING WITH A FUTURE DATE

One approach to Year 2000 testing is to set up a test system with the system clock to a date after December 31, 1999. Typically, the system clock is set to certain key dates that may be potentially troublesome. These include:

- September 9, 1999, to determine whether '9/9/99' is being used to represent missing date values.

- Just before midnight of January 1, 2000 to ensure that the application will run correctly if it is being executed when the clock changes to 2000.
- January 1, 2000, to ensure that dates in the Year 2000 are correctly processed.
- February 29 and March 1, 2000, to ensure that the application correctly recognizes 2000 as a leap year.

This approach can be used to determine if the application will run during and after the Year 2000 and whether the application correctly processes the current system date. By itself, this type of testing does not test for correct processing of dates in the input data (except for data that are dependent on the system clock).

Note that you only need to set your system time to dates within the range of dates that you will actually be running your SAS applications. For example, there is no need to test with your system date set for the year 2027 unless you intend to actually be running your SAS application in 2027. This is true even if your data contains dates that far in the future. If you expect to be processing date values greater than 2027, for example, you will need to test with *data* containing those values.

Obviously, resetting the system clock is not practical in multi-user environments. On MVS, it is possible to set up a special partition (LPAR) that has its clock set to a different date than the system clock. In addition, several vendors have developed tools that can be used on certain operating systems to simulate resetting the system clock. A number of these are available under MVS, and are referred to as SVC 11 screener products, because many IBM® mainframe applications determine the system date by calling the SVC 11 system routine. An SVC 11 screener product intercepts the call, and returns a date that you specify, rather than the date from the system clock. You can use an SVC 11 screener to make an application "think" that the system date is in the 21st century, enabling you to test the application in a simulated 21st century environment. Some common SVC11 screening utilities include HourGlass 2000™, HOTDATE™ 2000/SIMULATE, and TICTOC™. On HP-UX systems, a utility called Time Machine™ from SolutionSoft can be used to return a specific time or date whenever an application requests the system time.

SAS Licensing Considerations for Testing with a Future Date

Testing with your system clock set to a future date (or using one of the utilities described above) presents a special challenge for SAS software users. SAS software is licensed on a yearly basis, and the SAS System will not run if the system date is outside the licensing period. Thus, if you set your system clock to a date after the year 2000, and your SAS product authorization (setinit) does not cover the system date, you will receive a fatal initialization error when attempting to invoke the SAS System. Methods for circumventing this problem depend on the operating system you are using.

If you are running under MVS: If you are running under MVS and using an SVC 11 screener product, you can avoid this problem by using the SAS system option SVC11SCREEN. The SVC11SCREEN option (which must be specified in your system configuration file or added to the options parameter on the EXEC card in your JCL) causes all SAS date and time requests (except for license validation) to be done using SVC 11 calls rather than accessing the system clock. For example, if actual testing is done on March 1, 1998 and an SVC 11 screen utility is used to have SVC 11 return a system date of January 15, 2000, then SAS date and time handling routines see the system date as January 15, 2000, while the license validation routines see the system date as March 1, 1998. By having SAS license validation use the system clock, there is no need for a special setinit for Year 2000 testing.

In order to use the SVC11SCREEN option, you must be running SAS Release 6.09E TS455 or later. For complete details on the SVC11SCREEN option, see SAS Technical note TS-568**.** If you do not have web access, you can contact SAS Institute's Technical Support Division and request Technical Note TS-568.

If you are using a separate dedicated test machine or LPAR with the clock set to a future date, the SVC11SCREEN option cannot be used.[*] In this environment, you can use a new procedure (PROC DATECHG) to enable testing of SAS applications. In addition to the procedure modules, you will need special code to enable PROC DATECHG; this code must be obtained from Technical Support. PROC DATECHG is available only for the most recent release (6.09E for MVS) of the SAS system. For MVS, PROC DATECHG is available with SAS release 6.09E TS460 and later.

If you are running on a system other than MVS: On non-MVS systems, PROC DATECHG (see previous paragraph) must be used to enable testing of SAS applications with a future system date. For CMS, OpenVMS VAX, Unix, Windows, and OS/2, PROC DATECHG is available in the current maintenance for each system (6.09E TS460 or later for CMS and OpenVMS VAX; 6.12 TS050 or later for Unix, Windows and OS/2). Note that PROC DATECHG is available only for the current release of SAS software on each operating system.

TESTING WITH FUTURE DATA

In order to ensure that your SAS applications can correctly process dates with values after December 31, 1999, you need to run your applications using data containing dates of January 1, 2000 or higher. As with testing with the system time set to the future, there are several key dates that you should include in your data to check for correct processing. These include:

- September 9, 1999
- December 31, 1999
- January 1, 2000
- February 29, 2000
- March 1, 2000
- February 29, 2004
- March 1, 2004
- January 1, 2027

Remember that in order to test with future data, it is not necessary for your system clock to be set for these dates, nor is it necessary for SAS to be licensed for the dates.

STATIC TESTING

One of the most important things you can do to assess the compliance of your SAS applications is to look at the code itself. There are two approaches you can use—visual review and automated searching (using a program to search for certain keywords in your code). **Neither approach is sufficient by itself**—to effectively analyze your SAS applications, you should use a combination of the two.

What Files Need to be Looked at?

Obviously, static testing should start with any SAS source

[*] The SVC11SCREEN option causes SAS date-handling routines to obtain the date from the SVC 11 routine (which returns a date after 2000 if an SVC 11 screener is being used), while the license validation checking uses the system clock (which returns the current date). If a dedicated test machine or LPAR is being used, the system clock is set to a date after 2000, and the licensing routines cannot determine the current date.

programs that are part of your application. However, there are a number of less obvious files that also need to be evaluated. These include:

- Code that is included with %INCLUDE statements
- AUTOEXEC files
- Code included with the INITSTMT= option
- Configuration files
- Macros (including those in autocall macro libraries)

SAS catalogs also need to be checked for compliance. Some specific catalog entry types that may process date values include:

- SOURCE entries
- PROGRAM entries
- SCL entries
- Frames
- Letters
- Formulas

What Should You Look For?

When doing static testing of SAS source code, you should look for the following types of statements that are most likely to contain date processing:

- PROC or global statements that accept date values. Some examples of these include:
  - PROC CALENDAR
  - PROC CITIBASE
  - PROC PLOT or PROC GPLOT (dates in axis specifications)
  - AXIS statements

- Date and datetime informats and formats. Note that these are not limited to INPUT, PUT, INFORMAT, and FORMAT statements. For example, the following code processes two-digit years:

  ```
  cdate='26jul97';
  date=input(cdate,date7.);
  ```

- Functions (such as MDY) that manipulate date and datetime values.

- Date constants or literals (such as '26jul98'd).

- Variable names containing date-related strings, such as 'YEAR', 'YR', 'DATE', 'BEGIN', 'BIRTH', etc.

- Statements that add 1900 to a value or concatenate '19' to a text string.

- Use of the SYSDATE macro variable. (Because &SYSDATE returns the date in DATE7. format, any applications that use &SYSDATE should be checked particularly closely. In Version 7 of the SAS System, the SYSDATE9 macro variable can be used to return the date in DATE9. format)

- Use of 99, 9/9/99, or 99365 as a missing value.

- Use of the YEARCUTOFF= option

- Two-digit years in date entry fields

Many of the above examples can be detected by code scanning tools. However, there are a number of situations where date-related variables can be used in a more subtle fashion. For example:

- Any date information that is not stored as a SAS date value, for example, CLASS='1999' or INIT=98075, or YEAR=98.

- Dates created by concatenating variables, for example NEWDATE=mon||day||yr;.

- Use of dates in macros

- Use of variables that have permanent date formats associated with them. For example, suppose the variable BEGIN has been defined with a permanent format of DATE7. If your code contains the statement:

  ```
  put begin;
  ```
  there is no way to tell from the PUT statement that the variable is being written out with a two-digit year.

- Date information that is written to external files with user-written formats. For example, the following code would cause SAS date values to be written to an external file using two-digit years:

  ```
  proc format;
     value myfmt
     '01DEC1999'd-'31DEC1999'd='DEC99';
     '01JAN2000'd-'31JAN2000'd='JAN00';
      run;
  data;
     date='07JAN2000'd;
     file 'c:\filename.ext';
     put date myfmt.;
     run;
  ```

- Applications that process date stamps from SAS catalogs or catalog entries. Catalog update and creation dates are displayed by PROC CATALOG, PROC GREPLAY, the VIEWTABLE command and other components using a DATE7. format. If your application uses PROC CATALOG to write output to an external file and you subsequently read in the update or creation dates for catalogs or catalog entries using the DATE7. format, you should ensure that the YEARCUTOFF= option is set appropriately.

- SAS/ACCESS interfaces to databases that store date information as character strings or simple numerics. While most database products for PCs, VMS, and Unix systems store date information as an offset from a particular date (which are handled transparently by SAS/ACCESS), some older mainframe-based databases store dates as character strings or simple numerics, and require the specification of a DBCONTENT value (date format) in the descriptor.

  Unfortunately, it is often particularly difficult to identify use of SAS/ACCESS interfaces to databases because of the near-transparency of the access process. For example, you can use a SET, UPDATE, or MERGE statement to reference a view of an external data base, rather than a SAS data set.

- Applications that use SAS/ACCESS software to display date fields from external data bases. By default, the SAS/ACCESS view descriptors use a DATE7. format when displaying date values. To ensure that 4-digit years are used when displaying dates, a DATE9. format (or any other format that uses 4-digit years) should be specified for date fields.

Automated Search Tools

As mentioned earlier (but worth reiterating), there is no "silver bullet" application that will scan your applications for compliance. SAS Institute does not provide any packaged scanning applications. Some third-party vendors market such applications, but they have been neither evaluated nor tested by SAS Institute.

The most powerful "do it yourself" code-scanning tool is the SAS DATA Step. You can easily write a DATA step that will search code for use of date formats and informats, date functions, date constants, date-related variable names, and key strings or values, such as 1900 or '19'.

For applications that use SCL, the SCL Static Analyzer (available in Release 6.12 of SAS/AF® software) can be used to search for strings that indicate the use of date processing. Information on the SCL static analyzer can be found in the SAS Help system. You can also use the PRINT option in PROC BUILD to write SCL or PROGRAM entry code to an external file and scan the file using DATA step code.

Limitations of Code Scanning

Several limitations of scanning programs have already been mentioned, including problems detecting the use of dates in macros, autoexec files, and statements processed with %INCLUDE statements. In addition, code scanning is not "run time", which means that it does not detect cases where two-digit years exist in the data, but are not reflected in the code itself. For example, a SAS program may be reading in date values using a DATE9. informat, which would imply compliance. However, the DATE9. informat can also read raw data that contains two-digit years. Thus, your raw data could contain the string 12FEB01, which could be read in by the DATE9. informat as February 12, 1901 if the default YEARCUTOFF value of 1900 is used. Thus, use of compliant formats does not ensure compliance of the incoming data.

Additionally, there are date functions (such as the MDY function) that can accept either a 4-digit year or a 2-digit year. Simply knowing that the function is being used in an application does not tell you whether a 2-digit or 4-digit year is being used as an argument.

Finally, code scanning cannot be used with SAS programs that are compiled and stored using the SAS Stored Program Facility, unless you have saved the source code for the program.

RUN TIME CHECKING

Because of the limitations of code scanning, we recommend the use of run-time checking as a supplemental testing tool. With run time checking, any instance of two digit years being used with SAS date informats, functions, or date constants can be noted in the SAS log. The key difference between run time checking and code scanning is that run time checking uses using actual data and can uncover situations where two-digit years occur in the data. Code scanning, on the other hand, does not process any data.

A special module and SAS option are required in order to do a run time check. The module is a special version of the SASXDTU module (which is used internally by SAS date processing routines) and is available for download from our World Wide Web site.

After installing the replacement SASXDTU module, you can check for the use of two-digit years by using the following statement:

```
options debug='year2000check=nnn';
```

where *'nnn'* is the upper limit on the number of two-digit year occurrences that should be listed in the SAS log.

The following output illustrates use of the SASXDTU module and YEAR2000CHECK parameter to display diagnostic information in the SAS log:

```
     options debug='year2000check=100';
     data _null_;
        *-----MMDDYY informat-----*;
        x=input('030497',mmddyy6.);

        *-----MDY function-----*;
        x=mdy(2,2,98);

        *-----DATEJUL function-----*;

        x=datejul(97001);

        run;

WARNING:  TWO  DIGIT  YEAR  CHECK  (MDY):  MONTH=3
DAY=4 YEAR=97
WARNING:  TWO  DIGIT  YEAR  CHECK  (MDY):  MONTH=2
DAY=2 YEAR=98
WARNING: TWO DIGIT YEAR CHECK (Julian): 97001
```

Note that there are several limitations to using the SASXDTU module and the YEAR2000CHECK debug parameter for run time scanning. They include:

- The SASXDTU module processes informats, date functions, and date constants, but does not process output date formats.
- SASXDTU will not detect date processing that does not use SAS date values. If your application treats date information as character or integer values, the processing of these values will not be detected.
- Run time checking requires that the SAS program actually be executed using data. We recommend that you use SASXDTU with test data. If you need to test with production data, note that the SASXDTU module is experimental and has not been tested by SAS Institute Quality Assurance.
- You do not need to set your system clock to a future date when testing with SASXDTU, since the routine checks for use of two-digit years, regardless of actual values. If you do test in a future environment, you should make sure that you do not test your programs with actual production data, as permanent data sets and catalogs may be "stamped" with future dates and be unusable when you set your system date back.

The SASXDTU module is available for the following releases of the SAS system:

Operating System	**SAS Release**
MVS, CMS, VAX/VMS	6.07 and later
UNIX, Windows, OS/2®	6.11 and later
MacOS	6.12

Complete documentation on downloading, installing, and using the SASXDTU module is available on our World Wide Web site at: http://www.sas.com/y2k.

ARE YOUR DATA YEAR 2000 COMPLIANT?

To be truly Year 2000 compliant, the data that you are processing with your SAS application must be unambiguous as to the century. To ensure this, you should check both your input and output data (both raw data and SAS data sets) for compliance.

Note that you can still use 2-digit years in raw input and output data, so long as the century is interpreted correctly. You can use the YEARCUTOFF= option when reading in raw data to determine what century to associate with specific two digit years. While writing out or displaying dates with 4-digit years is strongly recommended, 2-digit years can be used if the interpretation of those years is unambiguous.

SAS Data Sets

If reading from or writing to a SAS data set, any data stored as a SAS date value is Year 2000 compliant, because all dates are stored as an offset from January 1, 1960 and there is no century dependence. However, if you are storing date information as a simple numeric or character variable, compliance is not ensured. To check for compliance of your SAS data, you should check each data set to make sure that all date-related variables are stored as SAS date values and have compliant formats. Some tools that you can use to do this include PROC CONTENTS, the VIEWTABLE window , or PROC SQL (which can be used to examine tables such as SASHELP.VCOLUMN which contain data set information). For example, the following PROC SQL statements display a list of all variables in available libraries that have associated date formats or informats:

```
proc sql;
select libname, memname, memtype, name,
format, informat
     from sashelp.vcolumn
      where (informat contains ‘YY’
              or informat contains ‘DATE’
              or format contains ‘YY’
              or format contains ‘DATE’);
```

Identifying variables that contain date information but are not stored as date values is more complicated. You can use PROC SQL statements similar to those above to list variable names or labels containing date-related strings, such as DATE, BIRTH, YEAR, and so on.

Data From External Sources

Most compliance problems with data being read from or written to external sources can be identified by examining the SAS programs that read and write the data. Any SAS informats or formats that use two-digit years (such as MMDDYY6.) or statements that add 1900 to a year value should be interpreted as signals that the fields being read or written may not be compliant.

In most cases, it is easier to correct the problem by modifying SAS code rather than reformatting the raw data. Suppose you are reading in a two-digit field as a variable called YEAR. Rather than expanding the data field to 4 digits, you can simply convert the variable to a SAS date, and (if desired) extract the year portion of the date. For example:

```
options yearcutoff=1920;
data;
input year 2.;
date=mdy(1,1,year);
year=year(date);
```

In this example, assuming that all YEAR values in the data fall between 1920 and 2019, two-digit years in the input file will be interpreted correctly.

Dates that are read from external data bases using SAS/ACCESS view descriptors are interpreted correctly if the dates are represented in the external database as the number of days since a certain date. SAS/ACCESS interfaces simply adjust the offset to the January 1, 1960 starting point used by SAS. However, some older mainframe data bases store date representations as character strings or simple numerics, and SAS/ACCESS view descriptors apply date informats via DBCONTENT to convert the date representation to a SAS date value. If the database you are accessing requires the use of date informats and contains dates with two-digit years, you can use the YEARCUTOFF= option to ensure that two-digit years are interpreted appropriately.

Even if you are reading in dates from a database that stores dates as an offset, you should check applications that use SAS/ACCESS interfaces, because the SAS/ACCESS view descriptors use a default DATE7. format when displaying date values. You can ensure that 4-digit years are used when displaying dates by specifying a DATE9. format (or any other format that uses 4-digit years) for date fields.

Of course, any application that combines SAS programs with other applications should be checked to make sure that the other applications are correctly processing date values. For example, suppose you have an Oracle application that reads in a raw data field of 12/31/02 as December 31, 1902 when you meant to have it interpreted as December 31, 2002. Accessing the Oracle field using a SAS/ACCESS interface will cause SAS to interpret the date as December 31, 1902.

CORRECTING PROBLEMS

Once you have identified actual or potential compliance issues in your SAS applications, you need to address them. Not all potential problems need to be corrected. If your SAS program writes out date values with two-digit years, you may not need to modify the program if the year can be correctly inferred from the context. For example, a printed report listing expiration dates for current drivers licenses can use 2-digit years. However, if the same program writes out the dates to a file and a subsequent application reads the two digit year and subtracts it from another two digit year, then the program(s) should be changed.

Here are some (but by no means all) steps you can take to correct potential compliance problems.

- Make sure that an appropriate value for the YEARCUTOFF= option is specified in your application. The specific value to use will obviously depend on the range of dates your applications use. Note that if all of your applications use the same general range, you can set the YEARCUTOFF= option at the system level so that all applications use the same default. Note also that you can specify the option at any point in your program to adjust for different date ranges in different input files. You can even create a “sliding window”, by making the YEARCUTOFF= value dependent on the current date. For example:

```
%macro slide(buffer);
   %local date year;
   %let date = %sysfunc(TODAY());
   %let year = %sysfunc(YEAR(&date));
   %let year = %eval(&year+&buffer-100);
   OPTIONS YEARCUTOFF = &year;
%mend slide;
```

  In the above example, the parameter BUFFER is the number of years between the current date and the end of the century window.

- Make sure all date information is stored as SAS date variables. Various functions are available to convert date information in other formats to SAS date values. This may also require modifying subsequent statements in your program to process the values as date values, but doing this will save you trouble in the long run.

- Change instances of 2-digit year date formats (such as DATE7.) to 4-digit years (such as DATE9.). This may not always be practical, particularly when writing output files, since the format of the file (and any applications that process it) may need to be modified to adjust for the extra digits.

- Modify applications that input date information through screens or forms to require 4-digit years.

- If variables in SAS data sets are assigned permanent formats, use PROC DATASETS or a DATA step to change the formats to ones using 4-digit years.

- Assign appropriate formats to date fields that are imported through SAS/ACCESS interfaces.

SUMMARY

Although the latest releases of SAS software are Year 2000 compliant, compliance of the software should not be your major concern—compliance of your SAS programs and data should be.

Unfortunately, there is no single or simple solution for making sure that your SAS applications are compliant. Your strategy should incorporate several types of evaluation including:

- Testing with a future date
- Testing with future data
- Static Testing (including visual code review and use of automated search tools
- "Run-time" checking using the YEAR2000CHECK parameter

Although by no means a total solution, the two most important things you can do to help ensure that your SAS applications are compliant are to make sure that all date information used in the application are represented as SAS date values, and to make judicious use of the YEARCUTOFF= option.

Finally, you should monitor SAS Institute's Year 2000 compliance page at http://www.sas.com/y2k for the latest information on product compliance and testing resources.

CONTACT INFORMATION

Mike Kalt
SAS Institute Inc.
Box 8000
Cary, NC 27513 USA
e-mail: sasmfk@wnt.sas.com

Rick Langston
SAS Institute Inc.
Box 8000
Cary, NC 27513 USA
e-mail: sasrdl@unx.sas.com

THIS IS A YEAR 2000 READINESS DISCLOSURE