Diagnosing Error Messages in Version 6 of the SAS System: But What Does It Really Mean?
Sally Painter, SAS Institute Inc., Cary, NC

ABSTRACT
The SAS Guide to Problem Solving and Error Messages, Version 5 Edition has proven to be an excellent source of information for users as they diagnose problems with SAS code. This manual allows you to find an error and then follow the guidelines given for correcting the error in your program. However, with the introduction of Version 6, a new error numbering scheme was born. Instead of listing a single error number, you may see more than one listed together on your SAS log. At first glance, it may seem more difficult to diagnose your own error messages.

This paper addresses error-message handling in Version 6 of the SAS System. The difference between compile time and execution time errors will be discussed, as well as how users can utilize this information to diagnose their own problems. In addition, you will learn how to take the new information given in a Version 6 error message and use it to your benefit as you correct programming problems in your own programs or those of your users. You will also be introduced to the kinds of information that you must have available when calling the Technical Support Division for help.

INTRODUCTION
As you program in the SAS language, at some point you will encounter an error. Writing an error-free program may be a hard goal to achieve. However, being able to understand errors and debug your SAS programs can be a fairly attainable goal.

The first step to effective debugging is knowing the kinds of errors that you may receive. Once this is clear, you will need to understand the Version 6 error numbering scheme and how error messages are generated. After learning this background information, debugging your programs should be a fairly painless process.

TWO TYPES OF ERRORS
In order to diagnose error messages, you must first understand how errors are reported. There are basically two types of errors:
- compilation
- execution

In the SAS System, each DATA or PROC step is compiled and executed as a separate piece. During the compile phase, the source code is checked for syntax errors. If one or more are encountered, the system issues a compile time error notifying you of the problem. This information is printed on your SAS log. For example, submit the following code:
```sas
data one;
input x y;
cards;
2 3
put x y;
rn;
```

The SAS log shows the following:
```sas
16 data one;
17 input x y;
18 cards;
ERROR 180_l22, Statement is not valid or it is used out of proper order.
19 run;
```

Notice the error underlining the PUT statement. Since no SAS programming statements can follow instream data, the PUT statement is flagged as being in error. This error was recognized at compile time.

Execution time errors are recognized after the program has successfully compiled. This type of error is usually harder to find in your program since it sometimes involves tracing the logic of your SAS program. For example, submit the following code:
```sas
data one;
input a b;
c=a/b;
cards;
1 2
2 0
run;
```

Notice that this code is syntactically correct. During the compile phase, everything goes as planned. However, once the step begins execution, an error of division by zero is detected. The SAS log shows the results:
```sas
12 data one;
13 input a b;
14 cards;
ERROR: Division by zero detected at line 1 column 6.
15 c=a/b;
16 cards;
ERROR: Mathematical operations could not be performed at the following places. The results of the operations have been set to missing values each place is given by: (Number of times) at (Line/Column).
15 at 14.
16 ERROR: The DATA set WORK.ONE has 2 observations and 3 variables.
17 run;
```

1481
Now that you understand the two types of errors, let's look at how to decipher error messages and debug your SAS code.

VERSION 6 ERROR NUMBERING

Beginning with Release 6.08, you may see some error messages of the following form:

```
SAS$ERROR 180-322: Statement is not valid or it is used out of proper order.
```

At first glance, this scheme seems very different from the error messages of past releases of SAS software. In reality, it is not so different. The first number identifies the component of the software where the error occurred (that is, did the error occur in a PROC PRINT step, or the DATA step, and so on). The second number just identifies the error number within the component. This information is most helpful to employees at the Institute who are responsible for finding software problems and fixing them. If you call Technical Support for help in understanding an error message, always have the exact numbers and text.

In most cases, it is relatively easy to determine which part of the SAS code is causing the error. Beginning with Version 6, SAS Institute adopted a layered approach to the SAS System. A majority of the SAS source code (approximately 70%) is portable, which means that it is exactly the same on all platforms. A smaller portable portion, the core (approximately 20%), is specific to the internal routines of the SAS System. For instance, the core section of the SAS System is responsible for handling I/O. The smallest part, approximately 10%, is host specific. This part contains the direct interfaces to the operating system and differs depending on the platform that you are using.

Because of this layered approach, known as MultiVendor Architecture™ (MVA™), a majority of error messages come from the portable section of the SAS System. These are generally easy to determine. However, when you start to look at system specific problems, the task of determining the cause of the error can be a little trickier. To ease this task, use the following information to determine where to begin for these obscure errors. Note that each error number listed corresponds to the component number that would be reflected on the SAS log.

- **host messages**: Errors ranging from 1-32 are reserved for host errors.
- **core messages**: Errors ranging from 33-60 are reserved for core messages.
- **SAS/ACCESS* messages**: Errors ranging from 61-99 are reserved for SAS/ACCESS messages.
- **application messages**: Errors ranging from 128-482 are reserved for product and procedure messages.

### RETURN CODES

In addition to error messages, Version 6 of the SAS System also provides more information via return codes for completed SAS programs. In previous releases of the SAS System under MVS, if the program run was successful it ended with a return code of 0. If the program ran unsuccessfully, it usually ended with a return code of 12.

In Version 6, this information has changed. Depending on your operating system, more information is available about the completion of your SAS programs. The following information is summarized by several of the major operating systems.

### MVS Operating System

- **Return Code of 0**: successful completion of the SAS program
- **Return Code of 4**: warning message(s) were issued during the program but processing continued
- **Return Code of 8**: error message(s) were issued during the program but processing continued
- **Return Code of 12**: error message(s) were issued during the program and processing ceased
- **Return Code of 16**: the ABORT statement caused termination of the program and no return code was specified on the ABORT statement
- **Return Code of 20**: the ABORT RETURN statement caused termination of the program and no return code was specified on the ABORT statement

### VMS Operating System

The SAS System running under the VMS operating system uses three symbols to reflect the status of the completed SAS program:

- **$SEVERITY**
- **$STATUS**
- **SAS$STATUS**

$SSTATUS shows information that you can use to decipher the final state of the entire SAS program. $SEVERITY and $STATUS show the most severe status of any step in the SAS program.

- **SAS$STATUS equals 0**: normal termination of the SAS program
- **SAS$STATUS equals 12**: an ABORT or an ABORT RETURN statement caused termination of the SAS program and no return code was specified on the ABORT statement
- **SAS$STATUS equals 999**: an ABORT ABEND statement caused termination of the SAS program and no return code was specified on the ABORT statement or a fatal error was encountered
- **SAS$STATUS equals 998**: a fatal error was encountered during execution of the SAS program

- **$SEVERITY equals 0**: warning message(s) were issued during the SAS program but processing continued
- **$SEVERITY equals 1**: normal termination of the program
- **$SEVERITY equals 2**: error message(s) were issued during the SAS program but processing continued
- **$SEVERITY equals 3**: informational message(s) were issued during the SAS program and processing continued


$SEVERITY$ equals 4
fatal error message(s) were issued during the SAS program

For more information concerning these symbols and the values returned to them from the SAS System, consult SAS Companion for the VMS Environment, Version 6, First Edition.

OS/2 Operating System

The OS/2 batch variable, ERRORLEVEL, reflects the status of a completed SAS program. The following information can be used to determine the state of the program at completion.

ERRORLEVEL equals 0
normal termination of the SAS program

ERRORLEVEL equals 4
warning message(s) were issued during the SAS program and processing continued

ERRORLEVEL equals 5
error message(s) were issued during the SAS program and processing continued

ERRORLEVEL equals 6
an ABORT statement was issued in the DATA step and no return code was specified

ERRORLEVEL equals 7
an ABORT RETURN statement was issued in the DATA step and no return code was specified. Processing of the program does not continue.

ERRORLEVEL equals 8
an ABORT ABEND statement was issued in the DATA step and no return code was specified. Processing of the program does not continue.

undefined value of ERRORLEVEL
job abnormally terminated

For more information concerning the ERRORLEVEL variable and the values returned to it from the SAS System, consult SAS Companion for the OS/2 Environment, Version 6, First Edition.

Automatic Macro Variables

In addition to the above information, there are several automatic macro variables available in Version 6 of the SAS System that can give information regarding the status of a terminated SAS job. They are SYSRC, SYSERR, SYSLIBRC, and SYSINFO.

SYSLIBRC contains the return code of a completed LIBNAME statement to allocate a SAS data library.

SYSRC provides the last return code generated by an operating system command during the SAS program. This command can be issued by one of the following statements: X command, %SYSEXEC, %TSO, or %CMS.

SYSERR provides the return code set by a SAS procedure. The following values are used:

SYSERR=0 procedure completed successfully
SYSERR=1 procedure was terminated by the user with a RUN CANCEL statement
SYSERR=2 procedure was terminated by the user with an ATTN or BREAK command

SYSERR=4 procedure completed successfully but warning message(s) were issued
SYSERR>4 an error occurred during procedure execution.

Another automatic macro variable, SYSINFO, contains return code information by some procedures. Information on the values of this variable are documented with the procedures that use it.

In addition, there are two new automatic macro variables and an updated autocall macro available in Release 6.07:

SYSFILRC automatic macro indicates the status of execution of the last completed FILENAME statement

SYSLCKRC automatic macro variable that indicates the status of the execution of the last completed LOCK statement with SAS/SHARE software

SYSRC autocall macro that allows you to test for a specific I/O condition

For more information on these new features, consult SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07.

DETERMINING THE CAUSE OF COMPILATION ERRORS

In most situations, it will be relatively easy to determine the cause of a compilation error. These will most likely be related to syntax of your SAS program. In most cases, the SAS System will flag the statement on the SAS log that it has determined caused the error. While this location is not always the exact location of the problem, it is generally close to the problem area. Once you have found the problem, use one of the SAS reference guides to determine correct usage of the statement in error.

DETERMINING THE CAUSE OF EXECUTION ERRORS

Debugging execution errors can be much harder. If the problem occurred in the DATA step, an invaluable tool is the PUT statement. Place this statement in your DATA step to write out the value of variables, or just write a text string to determine the flow of the DATA step. Let's look at the PUT statement in more detail.

PUT ...ALL...

Sometimes it is helpful to see the values of all the variables that are currently in the Program Data Vector (PDV). You can use the statement:

put paths...

to quickly write the values of all SAS variables on the SAS log. The form of the information is variable=value. In addition to dumping user defined variables, it also dumps the values of SAS automatic variables such as _ERROR_ and _N_. These two variables are extremely helpful because they relay information about the status of the SAS program. _ERROR_ has an initial value of 0 and is set to 1 if an error is encountered. _N_ is a counter of the number of passes through the DATA step.
PUT Variables

To see the value of selected variables on the SAS log, it is much easier to use the following statement:

```
put var1 var2 var3;
```

This statement will write the value of VAR1 followed by VAR2=VALUE and then the value of VAR3. Suppose you are writing a program that will generate information on all employees with the first name of 'Sally.' You might use the following program:

```
data employee;
  length first $5 last $10;
  input first $ name $10 age;
  cards;
  Randy Patterson 58
  Paige Viola 29
  Sally Painter 29
  Lynn Anderson 31
  Ken Wallace 45
  Alex Yeager 23
  Sally Baker 25
;
proc print data=employee;
  title 'All Current Employees';
```

... some data appears...

```
data subset;
  set employee;
  if first='Sally';
  proc print data=subset;
  title 'Employees with a First Name of Sally';
run;
```

Your program completes with no errors or warning messages issued but your output only contains information on one employee, Sally Painter, and you know that you should have information on three employees. What could have happened? Let's look at the SAS log for any clues:

```
22 data employees;
  length first $ 5 last $ 10;
  input first $ 5 last $ 8 age;
  cards;
  Randy Patterson 28
  Paige Viola 28
  Sally Painter 29
  Lynn Anderson 31
  Ken Wallace 24
  Alex Yeager 23
  Sally Baker 25
;
25 proc print data=employees;
  title 'All Current Employees';
```

... some data appears...

```
data subset;
  set employee;
  if first='Sally';
```

... some data appears...

```
data subset;
  set employee;
  if upcase(first)='SALLY';
```

... some data appears...

```
data subset;
  set employee;
  if upcase(first)='SALLY';
```

The output from the SAS program is shown in Output 1.

```
All Current Employees
FIRST  LAST   AGE
1   Randy Patterson 28
2   Paige Viola 29
3   Sally Painter 29
4   Lynn Anderson 31
5  Sally Smith 29
6   Ken Wallace 45
7   Alex Yeager 23
8   Sally Baker 25
```

```
Employees with a First Name of Sally
FIRST  LAST   AGE
1   Sally Painter 29
2   Sally Viola 28
3   Sally Baker 25
```

Output 1

Having used the diagnostic PUT statement, we can see the mistake: the case of the variable FIRST. In order to correct the problem, add the UPCASE function on the subsetting IF statement.

```
if upcase(first)='SALLY';
```

The PUT text

Use the following PUT statement to follow execution of the DATA step:

```
put 'in the DO loop';
```

Suppose you are writing a report to display monthly sales figures by department. During the week of February 10, 1992, all departments within the company worked together. You would like for the report to reflect this. You do not want to change the value of the variable, you just want to write something different on the report for this one week. Your SAS program runs without any errors or warnings but too many department codes on the final report have been changed. You are sure that this output is not correct, so what is the next step? This would be a good time to use a PUT statement.
Let's look at the program:

```
data depart;
  length manager $20;
  format date date7.;
  input code $3. manager date: date7. sales;
cards;
TSU Sally Painter 03feb92 1234.56
TXQ Larry Simpson 03feb92 759.00
TTR Jason Lions 03feb92 1212.12
TPO Kary Queen 03feb92 450.00
TTT Debbie Martin 03feb92 5555.50
TXX Alu Rabhan 03feb92 1999.75
TUU Forest Batten 03feb92 1212.12
TSU Sally Painter 03feb92 1234.56
TXQ Larry Simpson 03feb92 759.00
TTR Jason Lions 03feb92 1212.12
TPO Kary Queen 03feb92 450.00
TTT Debbie Martin 03feb92 5555.50
TXX Alu Rabhan 03feb92 1999.75
TUU Forest Batten 03feb92 1212.12
TSU Sally Painter 03feb92 1234.56
TXQ Larry Simpson 03feb92 759.00
TTR Jason Lions 03feb92 1212.12
TPO Kary Queen 03feb92 450.00
TTT Debbie Martin 03feb92 5555.50
TXX Alu Rabhan 03feb92 1999.75
TUU Forest Batten 03feb92 1212.12
```
By adding a PUT statement in your conditional check of the variable DATE, you should see that the conditional check is incorrect. You could use the following:

```plaintext
if date > '10FEB92'd then do;
  put 'IN THE IF CHECK';
  put date;
end;
```

You only need to check for date values equal to 10FEB92, not greater than or equal to 10FEB92.

In most cases, you will use a combination of options on the PUT statement to determine problems in your DATA steps.

**WHEN TO CALL FOR HELP**

Given the above information and additional information documented in the SAS Companion, you should be able to determine the cause of most of your SAS messages. You can also consult the SAS Notes for more information regarding specific procedures and the return codes they produce at termination. However, sometimes it is difficult to determine what caused the problem. If you receive an error message that needs to be clarified, or just need some pointers on where to get started to debug your SAS code, call the Technical Support Division at SAS Institute.

To get the most efficient service from Technical Support, call one of our direct telephone numbers. The direct numbers for Technical Support are

- 919-677-8008
- 919-677-8118

To reach a Technical Support switchboard operator, use:

919-677-8008

Call this number with new questions or problems that have already been tracked. If you are calling with a new question that involves diagnosing an error, tell the operator which component of the SAS System you were using when you received the error. Possible choices are

- a SAS software product, such as SAS/GRAPH® software
- a SAS procedure, such as PROC REPORT
- the DATA step
- system specific, such as an installation question.

If you are calling on a previously reported problem, tell the operator the tracking number for the problem and he or she will direct your call to the consultant who is handling your question.

To reach a consultant without going through the Technical Support switchboard, use our Automated Call Processing (ACP) facility. You can reach this facility by dialing

919-677-8118

For more information on using ACP, consult SAS Technical Report U-117, SAS Software Consultant’s Guide for the SAS System. You should not call this number if your problem has already been assigned a tracking number.
INFORMATION YOU WILL NEED

Once you reach a consultant, you will need to give him or her detailed information about the problem. Initially, you will probably need the following information:

- SAS log that contains the error message
- Details about exactly what was happening in the SAS program
- SAS output (if this appears to be incorrect)
- Release of SAS software
- Name and release of your operating system
- Amount of memory available to your SAS program (if the problem seems memory related).


CONCLUSION

As stated earlier, it would be very hard to write error-free programs. However, armed with the information discussed in this paper, you should find it easier to take errors and turn them around. You also have documentation at your fingertips that can make diagnosing SAS errors (and warnings and notes) a fairly painless process.

SAS, SAS/ACCESS, SAS/GRAPH, SAS/SHARE, MultiVendor Architecture, and MVA are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® Indicates USA registration. OS/2 is a registered trademark or trademark of International Business Machines Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.