Paper 327-2013

An Overview of Syntax Check Mode and Why it is Important

Thomas E. Billings, Union Bank, San Francisco, California

Abstract

The syntax check options direct the SAS® system, when a syntax error occurs while compiling source code, to enter a special mode to scan the remainder of the job for syntax errors after the point where the first error occurred. In this mode, only the header portion of some data sets are created, permanent data sets are not replaced, but global commands are executed (also a very few PROCs). The options controlling the mode are explained and illustrated using simple test jobs. The effects of setting and resetting the option within a job are explored, and there are some surprises along the way. The risks of running with the options enabled vs. disabled are discussed.

Introduction

If it is not suppressed by certain options, the SAS® system enters ***syntax check mode*** whenever a syntax error occurs in a DATA step or PROC that creates a data set. (The system does not enter syntax check mode when errors occur in _null_ DATA steps.) When this occurs, the SAS System sets the options OBS=0 and NOREPLACE. The SAS system continues to run, compiling statements and noting errors, and running global statements. Headers are created for new data sets, i.e., a version of the data set with zero rows, but existing permanent data sets are not overwritten. Most PROC invocations are syntax-checked but the PROC does not execute. PROC CONTENTS and PROC DATASETS are exceptions to this; the test programs below provide some insight on this point.

Syntax check mode is enabled/disabled in a job by the options:

- SYNTAXCHECK, NOSYNTAXCHECK in a batch or non-interactive environment.
- DMSSYNCHK, NODMSSYNCHK in the windowing environment.

To clarify the above, the DMSSYNCHK option works in the windowing environment, and SYNTAXCHECK should be used in all other environments, including most client-server environments. A related option, ERRORCHECK, controls whether the system enters syntax check mode for errors that occur in LIBNAME, FILENAME, %INCLUDE statements, as well as the LOCK statement of SAS/Share. Explicitly setting ERRORCHECK in your programs is also recommended so that SYNTAXCHECK and DMSSYNCHK work as expected.

Why enabling the syntax check option is important

If the syntax check option is disabled – by explicit user-selected option or by default via the autoexec file for the environment - the SAS system does not enter syntax check mode when relevant errors occur, and instead continues to run code after the point where the error occurred. This can be a high-risk situation as the result of running after errors occur can include loss of data/history files, corruption of data/files, and other unpleasant things.

The default values for the above options in your environments might not be what you expect. To identify the default values, run the code snippet below as the first code in a SAS session; it will that gives you the initial/default value for all of the options in the SAS environment. Be sure to run it in **all** SAS environments that you work in, and – if possible – in all other relevant environments available in your enterprise where your code might run.

```
proc options;
run;
```

Even if the syntax check option is enabled by default in your environment, it is not harmful – and in fact is a good idea – for your standard coding practice to explicitly set both of these options: SYNTAXCHECK, DMSSYNCHK, and also ERRORCHECK. Even if you work only in, say, SAS Enterprise Guide and don't use the windowing environment (SAS for Windows aka PC SAS), others in your enterprise might do so, so it is best to explicitly set both forms of the syntax check option in case others run your code in different environments.

While syntax check mode provides a low-risk method for SAS programs to run after an error occurs, the options for terminating a program after an error include:

- Gracefully terminate the program at or near the next step boundary (or other convenient/feasible point), i.e., don't even run the rest of the job (lowest risk method; coding may be required to accomplish this.)
- Go into syntax check mode and run the rest of the job (low risk.)
- Run normally after an error (potentially very high risk.)

The lowest risk option can be achieved by running in batch with the ERRORABEND option selected – as well as multiple other error options set to force a termination for any/all errors (e.g., ERRORBYABEND, DSNFERR, etc.). Unfortunately ERRORABEND might not work the way you want it to in the client-server environment common to many SAS Solutions (including SAS Enterprise Guide), as gracefully terminating programs in that environment is difficult. (Note that you can terminate on errors, but it can take a long time and you might not get a complete log or a (standard) log at all, depending on the SAS Solution.)

Although syntax check mode is low risk, it is not zero risk. Global commands are executed in the mode, including the X, FILENAME, LIBNAME, CATNAME statements. The X command can invoke processes that are external to the current job, and you may want to disable that via the NOXCMD system option for debug and perhaps for testing. Note that NOXCMD can disable some system features; check before using to avoid breaking your program. FILENAME, LIBNAME, CATNAME usually cause no direct harm when run.

Applying the principles for debug

You can - as a debug tool for relevant code - set the options OBS=0 and NOREPLACE to have the SAS system perform a syntax scan of your code. This places your program in the low-risk processing mode invoked during syntax check mode. However, when you do this, you actually don't enter formal syntax check mode unless errors occur and the relevant syntax check mode options are selected. After your code is working with OBS=0 and NOREPLACE, you can progress to running with OBS= set to some low number of rows, and REPLACE specified, for further debug.

Exploring the functionality of the option

Running with the syntax check option enabled

Next, we explore how the option works by reviewing the results of some simple test jobs. The jobs below were run under SAS 9.2 on a Linux server, using SAS Enterprise Guide. First, we run proc options and check the values for the options. We observe the following default/initialization values for our environment:

```
NODMSSYNCHK        Do not enable syntax check, in windowing mode, for a submitted
statement block
ERRORCHECK=NORMAL Level of special error processing to be performed
NOSYNTAXCHECK      Do not put SAS into syntaxcheck mode
```

So we need to explicitly set the options. Two macros were developed, one to print the values of options and system-set macro variables of interest, and a macro function to test for syntax check mode and a few other error conditions. The latter macro is provided to illustrate that a function-style macro may be useful in this context; it is not intended as a macro function to flag every possible error condition in every context.

```
%macro err_check;
       %* function-style macro that returns Y (for yes) if an error condition exists
         - including when obs, replace options are set as in syntax check mode
         - otherwise it returns N (for no);
       %if ((%nrstr(OBS=0) = %nrquote(%sysfunc(getoption(obs,keyword)))) AND
              (NOREPLACE = %sysfunc(getoption(replace,keyword)))) OR
              ( (&syserr ne 0) and (&syserr ne 4)) %then
              Y;
       %else N;
%mend;

%macro check_settings;
       %* print values of relevant option settings and error check macro;
       %put syntaxcheck option setting= %sysfunc(getoption(syntaxcheck,keyword));
       %put dmssynchk option setting= %sysfunc(getoption(dmssynchk,keyword));
       %put obs option setting=%sysfunc(getoption(obs,keyword));
       %put replace option setting=%sysfunc(getoption(replace,keyword));
       %put syserr= &syserr;
       %put syscc= &syscc;
       %put err_check macro= %err_check;
%mend;

options syntaxcheck dmssynchk;
run;  * <- forces step boundary;

%check_settings;
```

This yields the results:

```
syntaxcheck option setting= SYNTAXCHECK
dmssynchk option setting= DMSSYNCHK
obs option setting=OBS=9223372036854775807
replace option setting=REPLACE
syserr= 0
syscc= 0
err_check macro= N
```

So the syntax check option has been selected. Notice the large number for OBS=. This is the same number as you get for OBS=MAX, and equals $2^{64} - 1$. Next we create some test data sets followed by PROC PRINT invocations, then a DATA step with a syntax error to invoke the special syntax check mode. The edited log follows:

```
54          data test_base;
55            x=0;
56            y=1;
57            output;
58            x=1;
59            y=2;
60            output;
61            stop;
62          run;
NOTE: The data set WORK.TEST_BASE has 2 observations and 2 variables.

64          data test_2;
65            x=4;
66            y=5;
67            output;
68            stop;
69          run;
NOTE: The data set WORK.TEST_2 has 1 observations and 2 variables.

71          proc print data=test_base;
72            title "test_base file";
73          run;
NOTE: There were 2 observations read from the data set WORK.TEST_BASE.

79          data test_error;
80            x=3;
81            y=fakefunction(x);
                ____________
                68
ERROR 68-185: The function FAKEFUNCTION is unknown, or cannot be accessed.
82            output;
83          run;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: SAS set option OBS=0 and will continue to check statements. This may cause NOTE:
No observations in data set.
WARNING: The data set WORK.TEST_ERROR may be incomplete.  When this step was stopped
there were 0 observations and 2 variables.
```

Checking the options and parameters now reveals:

```
85          %check_settings;
syntaxcheck option setting= SYNTAXCHECK
dmssynchk option setting= DMSSYNCHK
obs option setting=OBS=0
replace option setting=NOREPLACE
syserr= 1012
syscc= 1012
err_check macro= Y
```

This shows that the special syntax check mode has been entered (per OBS, NOREPLACE), and both the system-set macro error indicator variables &SYSERR and &SYSCC are set to 1012, which indicates a “general error condition”. Also, the function-style error check macro successfully identifies the error state. Now that we are in syntax check mode, let’s try to do a few things, e.g., make a new data set and delete a file using PROC DATASETS:

```
87          data test_3;
88            x=5;
89            y=6;
90            output;
91            stop;
92          run;
NOTE: The data set WORK.TEST_3 has 0 observations and 2 variables.

94          proc datasets lib=WORK;
95            title "after error: try to delete test_2";
96
96        !  delete test_2;
97          run;
NOTE: PROCEDURE DATASETS used (Total process time):

99          proc datasets lib=WORK;
100           title "after error: was test_2 deleted?";
101         run;

NOTE: PROCEDURE DATASETS used (Total process time):
```

The attempt to create a new data set (test_3) yielded one with zero rows. PROC DATASETS ran but there is no message that a data set was deleted. The output of the 2 PROC DATASETS invocations is the same and the essential part is:

#	Name	Member Type	File Size	Last Modified
1	REGSTRY	ITEMSTOR	32768	11Nov12:14:22:42
2	SASGOPT	CATALOG	12288	11Nov12:14:22:47
3	SASMAC1	CATALOG	77824	11Nov12:14:22:47
4	TEST_2	DATA	16384	11Nov12:14:22:48
5	TEST_3	DATA	16384	11Nov12:14:22:48
6	TEST_BASE	DATA	16384	11Nov12:14:22:48
7	TEST_ERROR	DATA	16384	11Nov12:14:22:48
8	_PRODSAVAIL	DATA	16384	11Nov12:14:22:42

Which shows that file test_2 was not deleted by PROC DATASETS in syntax check mode. An attempt to use PROC APPEND to concatenate 2 of the files yielded similar results: the procedure ran but did not do anything/did not append the files.

Once we have entered the special syntax check mode, can we turn it off? Let’s try:

```
109         options nosyntaxcheck nodmssynchk;
110         options obs=max replace;
112         proc print data=test_base;
```

```
113          title "#1: try to print test_base";
114        run;

NOTE: There were 2 observations read from the data set WORK.TEST_BASE.
```

The above shows that setting all 4 of the highlighted options appears to turn the option and special check mode off. In early test runs, splitting the options above so that 2 options each are reset with 2 separate PROC PRINTs did **not** turn the check mode off. You need to specify all 3-4 options in one job step for the mode to be reset/turned off. Next, can we turn the option back on again? (The ability to turn the option on/off/on might be useful in a control program.) Let's try:

```
135        * try turning the options back on;
136        options syntaxcheck dmssynchk;
137        run;
138
139        %check_settings;
syntaxcheck option setting= SYNTAXCHECK
dmssynchk option setting= DMSSYNCHK
obs option setting=OBS=9223372036854775807
replace option setting=REPLACE
syserr= 0
syscc= 1012      <- 4 options are normal but &SYSCC still shows error
err_check macro= N
140
141        proc print data=test_base;
142          title "#3: try to print test_base";
143        run;
NOTE: PROCEDURE PRINT used (Total process time): <- nothing printed

145        %let syscc=0;     <- reset &SYSCC
146
147        %check_settings;
syntaxcheck option setting= SYNTAXCHECK
dmssynchk option setting= DMSSYNCHK
obs option setting=OBS=0              <- these options indicate we are back in syntax check mode
replace option setting=NOREPLACE
syserr= 3   <- SAS documentation suggests this value can indicate syntax check mode
syscc= 0
err_check macro= Y
148        run;

150        proc print data=test_base;
151          title "#4: try to print test_base";
152        run;
NOTE: PROCEDURE PRINT used (Total process time):     <- nothing printed
```

Recall that prior to the above, we went into syntax check mode, reset to get out of the mode, then tried to turn the options back on. We were successful in turning the options back on, but it appears - per the above - that the system "remembered" that it was in the special syntax check mode previously and went right back into the mode. That is, it did not continue normal processing, reentering syntax check mode again only ***after*** yet another error occurs. This has significant implications for control programs, i.e., there are constraints on going in and out of syntax check mode.

Running with the syntax check option disabled

Now let's run a version of the above program with the syntax check option disabled. Start by setting the options and checking.

```
48          * test run with syntax check mode options turned off;
49          options nosyntaxcheck nodmssynchk;
50          run;
51
52          %check_settings;
syntaxcheck option setting= NOSYNTAXCHECK
dmssynchk option setting= NODMSSYNCHK
obs option setting=OBS=9223372036854775807
replace option setting=REPLACE
syserr= 0
syscc= 0
err_check macro= N
```

Data sets test_base, test_2 were created exactly as for the first test job, and prints were done to verify the files. The DATA step test_error is again used to produce a syntax error, and we check the option settings:

```
81          data test_error;
82            x=3;
83            y=fakefunction(x);
              ____________
              68
ERROR 68-185: The function FAKEFUNCTION is unknown, or cannot be accessed.
84            output;
85          run;
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.TEST_ERROR may be incomplete.  When this step was stopped
there were 0 observations and 2 variables.

87          %check_settings;
syntaxcheck option setting= NOSYNTAXCHECK
dmssynchk option setting= NODMSSYNCHK
obs option setting=OBS=9223372036854775807
replace option setting=REPLACE
syserr= 1012
syscc= 1012
err_check macro= Y
```

OBS and REPLACE have not been reset so despite the error, we are not in the special syntax check scan mode. Now let's experiment with creating a data set and PROC DATASETS:

```
89          data test_3;
90            x=5;
91            y=6;
92            output;
93            stop;
94          run;
NOTE: The data set WORK.TEST_3 has 1 observations and 2 variables.

96          proc datasets lib=WORK;
97            title "after error: try to delete test_2";
98        !  delete test_2;
99          run;
NOTE: Deleting WORK.TEST_2 (memtype=DATA).
NOTE: PROCEDURE DATASETS used (Total process time):

101         proc datasets lib=WORK;
102           title "after error: was test_2 deleted?";
103         run;
```

Since the SAS system is running in normal mode, we can create and delete data sets. The results of the 2nd PROC DATASETS confirm that the file test_2 was deleted:

#	Name	Member Type	File Size	Last Modified
1	REGSTRY	ITEMSTOR	32768	11Nov12:14:16:45
2	SASGOPT	CATALOG	12288	11Nov12:14:16:51
3	SASMAC1	CATALOG	77824	11Nov12:14:16:51
4	TEST_3	DATA	16384	11Nov12:14:16:52
5	TEST_BASE	DATA	16384	11Nov12:14:16:52
6	TEST_ERROR	DATA	16384	11Nov12:14:16:52
7	_PRODSAVAIL	DATA	16384	11Nov12:14:16:46

Next, PROC APPEND runs but it concatenates a bad (defective) file and a good file, something you probably don't want to do. If a history file were to be corrupted (like test_error above) then appended to a good file, it would look normal – until you checked the row count or did a report and discovered the history data from previous runs had in effect "vanished".

```
105         proc append base=test_error data=test_base;
106         run;
NOTE: Appending WORK.TEST_BASE to WORK.TEST_ERROR.
NOTE: There were 2 observations read from the data set WORK.TEST_BASE.
NOTE: 2 observations added.
NOTE: The data set WORK.TEST_ERROR has 2 observations and 2 variables.
NOTE: PROCEDURE APPEND used (Total process time):
```

Can we turn the option on after an error? The answer is yes:

```
111         * try turning the options back on;
112         options syntaxcheck dmssynchk;
113         run;
115         %check_settings;
syntaxcheck option setting= SYNTAXCHECK
dmssynchk option setting= DMSSYNCHK
obs option setting=OBS=9223372036854775807
replace option setting=REPLACE
syserr= 0
syscc= 1012
err_check macro= N
116
117         proc print data=test_base;
118           title "#3: try to print test_base";
119         run;
NOTE: There were 2 observations read from the data set WORK.TEST_BASE.
```

In the above, syntax check was disabled when the syntax error occurred, so the system did not enter the syntax check scan mode. Because of this, when syntax check was enabled later in the program, the system was in a normal, non-error state.

Summary

1. Running jobs with syntax check mode disabled is very high risk, and generally should be avoided when writing to permanent files. Running with syntax check disabled can cause loss of/damage to permanent files (e.g., history files).
2. If your work is ad-hoc and outputs **only** to WORK, temporary, or expendable intermediate files, then disabling syntax check may be OK during development or for some debug work. This is the situation for some but certainly not all SAS Enterprise Guide projects.
3. &syserr = 3 should not be used to determine if a program is in the special syntax check mode. Instead use the macro/code suggested above to check the status of the OBS and REPLACE options.

4. Recommendation: explicitly turn syntax check on in your programs (OPTIONS SYNTAXCHECK DMSSYNCHK) even if those are the default for the environment you run in. This is because your code may be run in a different environment than the one you work in.
5. Running with the syntax check option set is much lower risk than running with the option disabled. However, it is not a zero risk option.
6. If the syntax check option is turned on and an error occurs, it is possible to turn the option and special syntax check mode off and continue running normally. If you then turn the option back on, the system will reenter the special syntax check mode, and this is probably not what you want. This has significant implications for error handling in control programs.
7. If the option is turned off at the start of a job, it can easily be turned on later in the job.

Epilogue

If you are not currently setting the options SYNTAXCHECK and DMSSYNCHK in your jobs, you should set the options in all production programs and any/all programs that write to permanent data sets. Additionally, run PROC OPTIONS and review the option settings in the environments you work in. You may want to explicitly set several error-handling options in your jobs, to reduce the risk of data loss on errors.

References:

SAS Institute, Inc. *SAS(R) 9.2 Language Reference: Concepts, Second Edition.* Online documentation:

- Error Processing in SAS
  http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000993436.htm

SAS Institute, Inc. *SAS(R) 9.2 Language Reference: Dictionary, Fourth Edition*. Online documentation:

- SYNTAXCHECK System Option
  http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#base-sysop-syntaxcheck.htm
- DMSSYNCHK System Option
  http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002217186.htm

SAS Institute, Inc. *SAS(R) 9.2 Macro Language: Reference*. Online documentation:

- SYSCC Automatic Macro Variable
  http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#tw3514-syscc-var.htm
- SYSERR Automatic Macro Variable
  http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000208995.htm

Paper presentation history:

Versions of this paper are being or were presented at the following SAS User Group conferences:

- SAS Global Forum 2013, San Francisco, California.

Contact Information:

Thomas E. Billings
Union Bank
Basel II - Retail Credit BTMU
400 California St.; 9th floor
MC 1-001-09
San Francisco, CA 94104

Phone: 415-765-2567
Email: tebillings@yahoo.com