

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

ABSTRACT

This paper primarily reviews the BASE SAS® SYSPARM automatic variable, what it is, how it is used, and provides examples for using it to conditionally execute SAS code. Use of SYSPARM can prove very useful in production type SAS jobs that must have the same conditional information updated with each execution by reducing the updates to the source code. SYSPARM can be updated before execution, accessed in a SAS program, and also be updated in the SAS program.

In addition, this paper briefly reviews other selected automatic variables that are created at SAS session invocation. These are SYSJOBID, SYSUID, SYSUSERID, SYSDAY, SYSDATE, SYSDATE9, SYSTIME, SYSSCP, SYSSCPL, SYSVER, and SYSVLONG. Unlike SYSPARM, these are read only variables.

INTRODUCTION – AUTOMATIC SYSTEM VARIABLES

System variables are created at the beginning of a SAS job or session. Each contains information about the SAS session that can be accessed by SAS programming statements. Unlike macro and data step variables, these variables are assigned values before any SAS code is processed. In general all the auto variables discussed in this paper are accessed as read only variables using the *'&system-variable'* format, the same as a macro variable reference.

INTRODUCTION – WHAT IS 'SYSPARM'?

SYSPARM is an automatic variable in SAS® which is put into the global Macro table at execution of every SAS job. SYSPARM is a system parameter, hence the name 'SYSPARM'. The value of SYSPARM is set to null (' ') at execution, but can be up to 254 characters long on the mainframe, and 32k on Windows. This allows information to be passed from the operating environment to SAS program steps and provides a means of accessing or using the string while a program is executing. SYSPARM can be used anywhere in a SAS program.

Since SYSPARM is a system option, it can be defined in the configuration file, the OPTIONS window, the OPTIONS statement, or at SAS invocation as an option. The value of SYSPARM can also be set within a SAS program.

The syntax is:

SYSPARM='character string' when used in mainframe SAS EXEC option statement and OPTIONS statement

-SYSPARM <">character string<"> when used in batch submissions in distributed command line based environments and configuration files

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

Assigning a value to SYSPARM is the same as specifying a value for the SYSPARM= system option. Retrieving the value of SYSPARM is the same as using the SYSPARM() SAS function in a SAS DATA step, or anywhere in the SAS program by using the automatic macro variable referenced by &SYSPARM.

SYSPARM can prove very useful in production type SAS jobs that must have the same conditional information updated with each execution. In some cases, it could be used instead of %LET and macro parms.

Why Use SYSPARM?

There are several reasons to use SYSPARM, but this paper will discuss three:

- 1) Use SYSPARM instead of %LET statements and macro parameters when update to source code is controlled and needs to be limited. If your environment has a version control process that requires checking source code in and out, the numbers of check-in/out are reduced because the internal SAS code changes are kept to a minimum. For instance, to change a %LET statement or update a macro parameter is considered a code change.
- 2) Use SYSPARM to shield the source code when someone other than the SAS programmer is executing the code. For instance, you have written a SAS program for a client that requires updating a %LET statement and for some reason the client deletes the semi-colon. If the update is made instead as a SYSPARM update either in the run or icon, the source code is never accessed for changes.
- 3) Using SYSPARM instead of or to feed %LET statements and macro parameters reduces possible errors caused by source code updates. See #1 and 2, but the main reason is that the need to update the actual source code is reduced, as well as possible areas for placing efforts toward correcting problems.

When Should SYSPARM Be Used

SYSPARM should be used when there is a need to control conditional processing within a SAS program. The same reason for using macros is also the same reason for using SYSPARM. Using macros and SYSPARM reduces maintenance and the need for source code changes. SYSPARM is used at the invocation of every SAS job or session. If you are using macros, then SYSPARM provides another tool to enhance your SAS code.

If there is a need to control source code changes (such as in SARBOX compliance) while also there is a need to provide dynamic coding, use of SYSPARM and macros will provide tools to reduce the overhead of tracking source code changes. Using %LET and macro parameters require updates to the values, and those updates must be made in the code. If your shop uses version management software (such as PVCS, CVS, or SourceSafe), the updates for %LET and macro parms are considered a source code change, even though there isn't a real change per se, then using SYSPARM reduces the incidents for check in/checkout.

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

If you have written SAS programs for non-SAS users, and they must update ANYTHING inside of the SAS code, then they have an opportunity to make an error that will cause your SAS code to fail. By using SYSPARM options as part of the execute statement or as the command line in the desktop icon, the changes to the source code are shielded from changes. This would mean that there may be several icons on the desktop with varying SYSPARM strings, or the user would need to know how to update the SYSPARM in the icon with the correct SYSPARM values. The updates would be the same as would be made in the %LET statement or macro parameter.

Sample Application

In the example below, the task is to create code that will print a report of records that are either (1) 'x' number of days old, or (2) within a range of dates. A sample of the data looks like this:

recdate	acct	name	amount
6/19/2007	123456	Dallas Stars	1000.00
6/19/2007	234567	Dallas Mavs	5000.00
6/19/2007	345678	Cowboys	4000.00
6/20/2007	123456	Dallas Stars	3000.00
6/21/2007	123456	Dallas Stars	6000.00

This program will be requested three-nine times a week by several different managers. To control the selection of records for the report, you could either

- (1) write more than one version of the code,
- (2) update the program for each run with or without a %LET statement, or
- (3) write the code using SYSPARM.

Using SYSPARM, such code would be executed on the mainframe using JCL like this:

```
//CJSPARM JOB (QA07,XXXXXXXX),CJAC.4DS,CLASS=S,NOTIFY=&SYSUID
//S1 EXEC SAS,HOLD=YES,
// OPTIONS='DQUOTE SYSPARM=DAY030 ' <-- COULD BE AS BELOW
//* SYSPARM OPTIONS ARE : 'DAY999' THEN PUT NUMBER OF DAYS FROM CURRENT DAY
//* OR USE START AND STOP DATES : MMDDYYMMDDYY
//*
//SYSIN DD DSN=QA.SASCODE(REFUNDS),DISP=SHR <= LOCATION OF SAS CODE
//
```

Using SYSPARM, such code would be submitted in batch on Windows using the following command line in an icon or run statement:

```
C:/sas.exe q:/qa/sascode/refunds.sas -dquote -SYSPARM "day030"
```

REFUNDS SAS code files contain source code:

```
FILENAME FILEIN 'REFUND-INPUT-FILE';
```

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

```
DATA ONE (KEEP=ACCT RECDATE NAME AMOUNT);
DO UNTIL (END=ALLIN);
IF _N_ EQ 1 THEN DO;
  TASK=SUBSTR((SYSPARM()),3);
IF TASK='DAY' THEN DO;
  DAYS=SUBSTR((SYSPARM()),4,3);
  FIRSTDAY=INPUT(PUT(TODAY()),6.),MMDDYY6.);
  LASTDAY=INPUT(PUT((TODAY())-DAYS),6.MMDDYY6.);
END;
ELSE DO;
  FIRSTDAY=INPUT(PUT(SUBSTR(SYSPARM()),1,6),6.),MMDDYY6.);
  LASTDAY=INPUT(PUT(SUBSTR(SYSPARM()),7,6),6.),MMDDYY6.);
END;
  CALL SYMPUT(FIRSTDAY,FIRSTDAY);
  CALL SYMPUT(LASTDAY,LASTDAY);
  RETAIN FIRSTDAY LASTDAY;
END;
INFILE FILEIN END=ALLIN;
INPUT RECDATE MMDDYY6. @;
IF LASTDAY LE RECDATE GE FIRSTDAY THEN DO;
  INPUT ACCT 6. NAME $30. AMOUNT PD6.2;
  LABEL ACCT='ACCOUNT*NUMBER'
        NAME='PAYER*NAME'
        AMOUNT='REFUND*AMOUNT'
        RECDATE='REFUND*DATE';
  OUTPUT;
  DELETE;
END;
ELSE DO;
  INPUT;
  DELETE;
END;
END;
RUN;
```

Parse 1st 3 characters of SYSPARM looking for 'day'. If it is, then get the number of days back for the report

If not a 'day' report, then parse the start and end dates from SYSPARM

Put the start and end dates from SYSPARM into a macro variable

```
PROC SORT;
BY RECDATE ACCT;
RUN;
```

```
PROC PRINT _N_ U D SPLIT="*";
BY RECDATE;
ID ACCT;
VAR NAME;
FORMAT RECDATE MMDDYY8. AMOUNT COMMA19.2;
SUM AMOUNT;
TITLE "CITY OF DALLAS - PROGRAM NAME IS &SYSJOBID RAN on &SYSDATE";
TITLE3 "REFUNDS &FIRSTDAY THRU &LASTDAY";
FOOTNOTE "SORTED BY REFUND DATE";
RUN;
```

Include SYSJOBID, SYSDATE auto variables in TITLE, and put the values for the first and last date for record selection in TITLE3.

```
ENDSAS;
```

Discussion of Code

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

The code was developed on the mainframe many years ago, but the source code would be the same with the exception of the input file. Assuming that the input is the same format for each operating system, then only the 'filename' would need to be updated. The code was developed to allow multiple managers to execute the SAS program without them having to directly update the code if a %LET statement of macro parms were used. In this way, the source code uses the parm sent via the SYSPARM value to drive the record selection and the report dates within the title.

The 1st section of the code is executed on the 1st loop of the DATA step to query the SYSPARM value. In the case of this program, the values should be either the word 'day' followed by the number of days back from the current date the report should select data, or the start and end date ranges. This code parses the value from SYSPARM, creates the dates for the start and end ranges, then passed to the Macro Symbol table via the CALL SYMPUT function.

The rest of the code is functional to create the report using the dates created in the 1st section. The dates are used again in the final PROC PRINT step in the TITLE statement. Also included in the title group is the name of the job (if mainframe) or user/session name, and the date of the program execution.

WHAT ABOUT THE OTHER AUTOMATIC SYSTEM VARIABLES?

There are other very useful system level automatic variables that can be used to automate your programs. They can be loosely grouped as date/time, identification, operating system info, and SAS session info.

IDENTIFICATION: The variables SYSJOBID, SYSUID, and SYSUSERID return information specific to mainframes, although each will return the user id in Windows. One method of using SYSJOBID on the mainframe is to use the job name as a variable to define code. For instance, a mainframe job name is 'QARmony', where the month and year of the report request is included in the job name. Parsing SYSJOBID can provide the month and year for record selections as in this example:

```
MONTHRPT=SUBSTR(&SYSJOBID,3);
```

DATE/TIME: The variables SYSDAY, SYSDATE, SYSDATE9, and SYSTIME return the starting day-date-time stamps for when the job started regardless of operating systems. These are in standard SAS formats, but are in character format. To use as a date, use the "D" or "T" modifiers to put it into date or time format, such as

```
IF DATE-VAR=&SYSDATE'D;
```

OPERATING SYSTEM: The variables SYSSCP and SYSSCPL return information regarding the operating system that SAS is executing in. If you have SAS code that runs on multiple operating systems, instead of hard coding operating specific code and creating multiple versions, put all the code for each operating system in one version, and use these variables to define which set of code should be executed. For instance, the code

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

used in the previous example executes on the mainframe and Windows. Instead of manually setting the filename at the beginning of the program, code to 'decide' which operating system could be included instead. The SAS code would be in a macro and could look like this:

```
%MACRO SET-INPUT;
    %IF &SYSSCP=OS360 %THEN
        FILENAME FILEIN 'MAINFRAME-REFUND-INPUT-FILE';
    %ELSE
        FILENAME FILEIN 'WINDOWS-REFUND-INPUT-FILE';
%MEND SET-INPUT;

%SET-INPUT;
```

The resulting code that is passed to the SAS Supervisor would be only one of the 'filename' statements which is specific for the operating system.

SAS ENVIRONMENT: The variables SYSVER and SYSVLONG are set to value of the version of SAS being executed. If you are maintaining the same set of code functionality in two versions of SAS, especially V6- and V7+, then this variable could allow the code to 'decide' which set of code to execute that is germane to the SAS version.

CONCLUSION

Using SYSPARM in the same way that macro %LET statements and parameters are used can allow the SAS program to make decisions, conditionally execute parts of programs as required, and reduce program modifications. Unlike %LET statements, SYSPARM does not require the source code to be updated directly with the parm values. Since the value of SYSPARM is set at invocation of a SAS session, the value is available in the SAS system global macro variable table, and doesn't need to be set within the program.

If you have programs that are used by casual or non SAS users that require them to update %LET statement values or macro parm values, you can shield them from updating the source code and reduce their opportunities to make errors. Also, by using SYSPARM instead of %LET statement updates, you can reduce the number of instances of the source code and store it centrally.

Using SYSPARM can save time, reduce changes in source code, reduce possible error points, and provide flexibility in writing programs for multiple clients and uses. Like Macro statements, SYSPARM can be used to aid better software coding procedures and extend the power of your SAS code.

The other automatic system variables each have uses that also can save time and coding and maintaining SAS code. In some cases, using these variables can eliminate the need to update code.

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

Using the automatic variables can extend code, but it does take a little time upfront in setting up the code, the same as in using macro language code. But the benefits are many. If you are a macro user, it will pay to look at these automatic SAS variables.

Using SAS® SYSPARM And Other Automatic System Variables

Clarence Wm. Jackson, CSQA

References

SAS® Language Ref
SAS® Macro Language Guide

Author

Clarence Wm. Jackson. CSQA
City of Dallas
1500 Marilla St, L1AN
Dallas, TX 75201
clarence.jackson@dallascityhall.com (work)
CJac@compuserve.com (home)