

***Using SAS®***

***SYSPARM***

***And Other Automatic System  
Variables***

**Clarence Wm. Jackson, CSQA**

# *Agenda*

⇒ **SYSPARM**

⇒ **Other Automatic System  
Variables**

⇒ **Questions**

## ***What Are '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.

## ***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'.
- ⇒ 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.

## ***SYSPARM Syntax***

**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

## *Why To Use SYSPARM*

- ⇒ 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 in PVCS VM, etc.

## *Why To Use SYSPARM*

⇒ 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.

## *Why To Use SYSPARM*

⇒ Using SYSPARM instead of or to feed %LET statements and macro parameters reduces possible errors caused by source code updates.

⤴ See previous reasons, but the main result is the need to update source code is reduced, as well as possible reduction in efforts toward correcting problems.

## *When To Use SYSPARM*

- ⇒ 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.

## *When To Use SYSPARM*

- ⇒ 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, 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.

## *When To Use SYSPARM*

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.

## *Example*

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.

## *Sample Data*

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

# ***SYSPARM INVOCATION***

- ⇒ Using SYSPARM, such code would be submitted in batch on the mainframe using the following JCL statements:

```
//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' NUMBER OF DAYS FROM CURRENT DAY
/* OR 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 Source Code***

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

```
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;
```

## *REFUNDS Source Code*

```
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;
```

## *REFUNDS Source Code*

```
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 "CJac Consulting - PROGRAM NAME IS  
&SYSJOBID, RAN on &SYSDATE";  
    TITLE3 "REFUNDS &FIRSTDAY THRU &LASTDAY=";  
    FOOTNOTE "SORTED BY REFUND DATE";  
RUN;  
  
ENDSAS;
```

## *Other 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
  - ↳ identification,
  - ↳ date/time,
  - ↳ 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 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:

# *Operating System Example*

```
%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 SAS Automatic variables**

- ↪ **reduces maintenance of SAS programs**
- ↪ **allow for modular and structured design of code**



# *Contact Info*

Clarence Wm. Jackson, CSQA

[CJac.CSQA@sbcglobal.net](mailto:CJac.CSQA@sbcglobal.net)

214-957-6132 cell





[sasCommunity.org](http://sasCommunity.org)  
A collaborative online community for SAS® users worldwide

**SCSUG** 

**South Central SAS Users Group**  
Organizing and Delivering SAS Educational Conferences  
Throughout the South Central United States



**NoTSUG**  
North Texas SAS Users' Group