

Obtaining User Criteria Without Editing the Code

Michael A. Mace, SPS Software Services Inc., Canton, Ohio

ABSTRACT

%WINDOW provides an elegant means to gather user criteria for customized or ad hoc reports. Some of the benefits are: only Base SAS® is needed (no SAS/AF®), there is only one program to write and maintain, and the end-user never has to edit any code. Several examples will demonstrate gathering single and multiple choices per parameter and conditionally gathering criteria. In addition, the use of %WINDOW will be compared to using %PUT and %INPUT.

INTRODUCTION

The SAS® system provides a variety of information reporting capabilities which can be used for canned, hard-coded reports that are produced on a regular basis and also reports with basic layouts whose specific contents change based on some criteria input by the user. Sometimes programs are written with %LET statements and then given to users to edit with the criteria for that particular execution. Yikes! This paper examines several techniques to interactively, i.e. at execution time, gather the criteria for customized or ad hoc reports without anyone having to edit the code.

If your site has licensed SAS/AF®, a program entry could be created to gather user criteria. While this technique does provide a great deal of flexibility and macro programming options, it does add another layer of complexity to an application. What I proffer here is the use of Base SAS® only, thus keeping things as simple as possible, for the client and the developer / programmer / report-writer.

My first choice for gathering a few, simple criteria, at least when I am using mainframe versions of the SAS® system prior to 6.09, is usually %PUT, %INPUT, along with &SYSBUFFR, tools of the macro facility which comes with the Base SAS® system. These provide a means to step the user through the criteria-gathering process, expecting typed responses. That can be a concern; after all, computers are persnickety things and will produce output commensurate with the input given to them (GIGO).

%WINDOW provides a very elegant interface to the user along with all of the flexibility and programming power your application should need. In fact, my use of %WINDOW came about because a client, with a very simple application, required the ability to enter multiple values for a given parameter and did not want the bother of typing precisely and I did not want to create the SAS/AF® program entries that would have been required.

Let us examine the use of %PUT - %INPUT and %WINDOW. Please refer to *SAS® Guide to Macro Processing, Version 6, Second Edition* for a thorough discussion about the macro facility, macro variables, %PUT, %INPUT, &SYSBUFFR, %WINDOW and %DISPLAY.

%PUT and %INPUT

This technique, which can be used on the mainframe with versions of the SAS® system before 6.09, steps the user through the criteria-gathering process. %PUT is used to display the prompt; %INPUT is used to capture the response in a macro variable. An accessory to %INPUT is the automatic macro variable, &SYSBUFFR, which accepts any response to a %INPUT prompt that is not matched with a specific variable. Here is a short example:

```
%PUT Please enter the variable for which you want  
a frequency table ;  
%PUT Choices: varname1 varname2 varname3 ...  
varnameN ;  
%INPUT freqvar ;
```

```
PROC FREQ DATA= fileref.dsname ;  
  TABLES &freqvar / options ;  
RUN ;
```

When working with dates, I make the prompt very specific, remembering that macro variable values are text. Therefore I request the date values in DATE9. format so that no data conversion is necessary. Also notice that I have requested a four-character year to make sure everyone remembers that the turn of the century is upon us.

```
%PUT Please enter the BEGINNING date as
DDMONYYYY: ;
%PUT i.e. 06APR2007 ;
%INPUT begndate ;
```

The response could be used thusly:

```
SET filref.dsn
WHERE=("&begndate"D<=datevar<="&enddate"D))
;
```

Multiple values can be accepted with %INPUT in one macro variable; however the format of the response is critical, because a space in the response delimits the value. Here is an example within a macro.

```
%MACRO macronam ;
CLEAR ;
%PUT Please enter the first three characters of
desired homebase(s), ;
%PUT in single quotes, - separate multiple
homebases with commas, ;
%PUT ex.: '041','042','037' ;
%INPUT hb ;

DATA dsname ;
SET filref.dsname ;
%IF %LENGTH(&hb) GT 0
%THEN %DO ;
        IF SUBSTR(homebase,1,3) IN(&hb) ;
%END ;
RUN ;
%MEND macronam ;
%macronam ;
```

&SYSBUFFR can also be used to gather multiple values in one macro variable because it accepts whatever the user enters. This allows a great deal of flexibility, independence, and yes, uncertainty; however it is crucial for our purposes. Whether you use %INPUT with multiple macro variables or &SYSBUFFR, in a particular situation, will be determined by how many different pieces of information are you expecting from the user and how the response is going to be used. Several examples of how I have used &SYSBUFFR are:

```
CLEAR ;
%PUT Please list ALL of the variables you want on
the report ;
%PUT in the desired order: ;
%PUT THEN press <ENTER> ;
%INPUT ;
%LET var = &SYSBUFFR ;
%PUT %STR( ) ;
```

```
%PUT If you want to specify an additional
subsetting criterion, ;
%PUT please enter it as:
%PUT variable comparison_operator value
%PUT -OR- ;
%PUT FUNCTION(variable, parameters)
comparison_operator value ;
%INPUT ;
%LET criteria = &SYSBUFFR ;
TITLE4 "%UPCASE(&criteria)" ;
```

```
/* This code is necessary to prepend the word 'IF'
and append the ';' to the macro variable &criteria */
%MACRO criteria ;
%IF %LENGTH(&criteria) = 0
%THEN ;
%ELSE %STR(IF &criteria ;) ;
%MEND criteria ;
```

For a complete ad hoc program using %PUT, %INPUT, and &SYSBUFFR, please see my paper, "Ad Hoc Reports for Users", in the NESUG '97 proceedings.

One assumption I have made in my use of %PUT and %INPUT is that the client will enter exactly what is requested as requested; if the criteria are not input correctly, an empty report will be produced. I do not do any error checking when it comes to the client's inputs; that could be incorporated if deemed necessary.

I should mention that when I work with my clients in creating an application, I constantly emphasize that they are the owners and users of the application and need to know the basics of what things are, how things work, and what can be reasonably expected. Just as they educate me about their needs, I try to educate them about the tools of the SAS® system. This mutual understanding helps not only in the initial development but also during any troubleshooting or future modifications.

To conclude our discussion of %PUT and %INPUT, please notice that the only control over the user interface is the use of CLEAR; and %STR(). While functional for this 'one question at a time' method, it is not the most appealing. Also, let me re-emphasize that when using %PUT and %INPUT, generally we are asking for one parameter or value at a time and sometimes we expect the response to be in a specific format.

%WINDOW

The %WINDOW statement defines macro windows, which are displayed with the %DISPLAY statement, both of which are part of the macro language included in the Base SAS® system.

Please refer to *SAS® Guide to Macro Processing, Version 6, Second Edition* for a thorough explanation of these two statements and all of their options. %WINDOW combines the functionality of the %PUT statement, displaying text, and the %INPUT statement, accepting input into macro variables. What it adds is the ability to format the display and open the entire user interface at one time.

The %WINDOW statement can define a single set or multiple groups of fields. Once defined, a macro window exists until the end of the SAS® session and hence can be used again as needed. The fields of a macro window are either text, existing macro variables, or new macro variables. Variables can be protected or not, and if editable, required or not. Once displayed and after entering values in any required fields, the window is removed by pressing [ENTER]. Therefore a prompt should be provided to inform the user how to maneuver (use [TAB], the cursor keys or mouse) and indicate that input is complete.

The %WINDOW and %DISPLAY statements can be placed within or outside a macro. If placed inside, the macro window will be defined each time the macro is executed. For efficiency, place the %WINDOW statement in open code or the outermost layer of macro nesting. Macro variables created in a macro window are stored in the current referencing environment. When the %DISPLAY statement is placed inside a macro, more than one group of fields can be merged into a single display and groups of fields can be conditionally displayed. Also note, only one windowing environment is created when the %DISPLAY statements are inside a macro.

Two automatic macro variables are created with the %WINDOW statement:

SYSMSG contains specified text (probably by means of a %LET statement) that is displayed on the message line of the window. Its value is set to null before each execution of a %DISPLAY statement.

SYSCMD contains the last command line command not recognized by the display manager. Its value can be used for conditional processing.

Let's see how to define a macro window.

```
%WINDOW wndoname <window-options>
    <GROUP= grupname>
    field-definitions
    row column text | macro-variable <length>
    <options>
;
```

The window, group, and variable names must conform to standard SAS® naming conventions. Notice that the %WINDOW statement completely defines the display, with all text, fields, and formatting in one statement, i.e. there is one semi-colon, at the end of the definition.

The window options include:

```
COLOR= background color
IROW=
ICOLUMN=
ROWS= (including borders)
COLUMNS= (including borders)
KEYS= libref.catalog.key-entry
      (default:SASUSER.PROFILE)
MENU= libref.catalog.pmenu-entry
      (default:SASUSER.PROFILE)
```

Field definitions consist of constant text or a macro variable to be displayed, its position within the window, and any attributes. Text must be enclosed in single or double quotes. Macro variables must be macro variable *names*, not macro variable references. The integer field length indicates the number of characters used for display or input of macro variable values. It is recommended that the length always be specified. (If the current value of the macro variable is null, such as when the macro variable is defined in a %GLOBAL or %LOCAL statement or when a new macro variable is being defined within the macro window, the field length is zero and nothing can be input.) When defining macro variables within the macro window, it is also recommended that the UNDERLINE attribute (see below) be used to indicate the position and length of requested input.

The position is identified by row and column specifications which consist of the following pointer controls and a number or macro expression that evaluates to a number:

```
# - specifies the row within the window
/ - moves the pointer to column 1 of the next row
@ - specifies the column within the window
+ - moves the pointer the specified number of
positions
```

The field definition options are:
ATTR= <BLINK HIGHLIGHT REV_VIDEO
UNDERLINE>
AUTOSKIP= <YES | NO>
COLOR= color
DISPLAY= <YES | NO>
PROTECT= <YES | NO>
REQUIRED= <YES | NO>

The syntax for the %DISPLAY statement is:

```
%DISPLAY windowname<.groupname> <options> ;
```

The options are:

NOINPUT - within a macro, used with a group, causes the group to remain visible when later groups are displayed

BLANK - clears the display, preventing fields from a previous display from appearing; useful only within a macro definition

BELL - (obvious)

Note that when a macro window is displayed, SAS® Display Manager commands can be entered, as well as customized commands, the value of which are captured in &SYSCMD.

Here is an example, enclosed in a macro in order that user input can be validated. Pointer controls are not used before the first text field to demonstrate that the default positioning for the first field is row 1, column 1.

Also, while the %WINDOW and %DISPLAY statements do not need to be within a macro, the %IF and %THEN and macro label statements do.

```
%WINDOW wndoname
  ICOLUMN= 15
  IROW= 10
  COLUMNS= 80
  ROWS= 20
```

```
"< ROW 1, COLUMN 1 of the displayed window"
```

```
  #3 @5 "Company Name: Title of screen"
  #15 @15 "Please enter the requested data,
THEN press [ENTER]"
  #6 @15 "Company (21,41,51): "
    @40 company 2
    ATTR= UNDERLINE REQUIRED= YES
  #8 @15 "Customer Premise ID: "
    @40 premise 7
    ATTR= UNDERLINE REQUIRED= YES
  #10 @15 "Billing date (YYYYMM):"
    @40 yyyyymm 6
    ATTR= UNDERLINE REQUIRED= YES
  ;
```

```
%MACRO macronam ;
%LET null = ;
```

```
%lblstr: ;
%LET SYMSG= Press [ENTER] only after filling in
ALL fields;
%DISPLAY wndoname ;
```

```
%IF (&company NE 21 AND &company NE 41 AND
&company NE 51)
OR %LENGTH(&yyyyymm) LT 6
%THEN %GOTO lblstr ;
/* NOTE: Previous values are not cleared */
```

```
%MEND macronam ;
%macronam ;
```

Great!; but that is still just asking for one value per parameter and the user is still typing the responses. What about when the user wants to select several values and doesn't want to spell out anything?

```
%LET vino1 = ; %LET vino2 = ;
%LET vino3 = ; %LET vino4 = ;
```

```
%WINDOW critria1
  IROW= 10
  ICOLUMN= 10
  ROWS= 20
  COLUMNS= 60
  GROUP= wines
  #4 @15 "Please choose which wine(s) you like:"
  #6 @20 vino1 1 ATTR= UNDERLINE
    @22 "Burgundy" COLOR= MAGENTA
  #7 @20 vino2 1 ATTR= UNDERLINE
    @22 "Rose" COLOR= RED
  #8 @20 vino3 1 ATTR= UNDERLINE
    @22 "Zinfandel" COLOR= PINK
  #9 @20 vino4 1 ATTR= UNDERLINE
    @22 "Chardonnay" COLOR= YELLOW
  #15 "**** After entering choice(s), press
[ENTER]"
  GROUP= zinfandl
  #2 @5 "Oh! You really must try the Zinfandel"
  ;
```

```
%WINDOW verify
  IROW= 10
  ICOLUMN= 20
  ROWS= 15
  COLUMNS= 60
  #3 @5 "You have chosen the following:"
  / @5 wines PROTECT= YES
  // @5 "Is this correct (Y/N)?"
    +3 ansr 1
    ATTR=UNDERLINE REQUIRED= YES
  ;
```

```

%MACRO macronam ;
%lblstr: ;
%DISPLAY critria1.wines BLANK ;
%LET null = ;

%IF &vino3 EQ &null
%THEN %DO ;
    %LET SYMSG= "Oh! You really
must try the Zin";
    %DISPLAY critria1.zinfandl BELL ;
    %GOTO lblstr ;
%END ;

%IF &vino1 NE &null
%THEN %LET vino1 = BURGUNDY ;
%IF &vino2 NE &null
%THEN %LET vino2 = ROSE ;
%IF &vino3 NE &null
%THEN %LET vino3 = ZINFANDEL ;
%IF &vino4 NE &null
%THEN %LET vino4 = CHARDONNAY ;
%LET wines = &vino1 &vino2 &vino3 &vino4 ;

%LET ansr = &NULL ;
%DISPLAY verify BELL ;

%IF %UPCASE(&ansr) EQ N
%THEN %GOTO lblstr ;

/* &wines can now be used as the specified criteria*/

%MEND macronam ;
%macronam ;

```

Now the user simply tabs to the underline in front of his/her choice(s) and types any character. The %IF - %THEN statements determine what value(s) were chosen and there is no concern about misspelling or forgetting a quote or comma. Feedback is provided to the user regarding their selections along with a means to revise them.

Finally, here is a general criteria-gathering example that utilizes window groups for specialized error messages.

```

%WINDOW example3
GROUP= hdrftr
#1 @20 "Ad Hoc Report: Criteria Gathering"
/ @30 "&SYSDAY &SYSDATE"
#20 @5 "**** After specifying criteria, press
[ENTER]"

```

```

GROUP= criteria
#5 @5 "Enter the beginning
date(DDMONYYYY):"
+3 begndate 9
ATTR=UNDERLINE AUTOSKIP=YES
#6 @5 "Enter the ending date(DDMONYYYY):"
+3 enddate 9
ATTR=UNDERLINE AUTOSKIP=YES
#8 @5 "Select which variable(s) by which to
SORT:"
#9 @5 var1 1
ATTR=UNDERLINE AUTOSKIP=YES
+1 "varname1"
+3 var2 1
ATTR=UNDERLINE AUTOSKIP=YES
+1 "varname2"
+3 var3 1
ATTR=UNDERLINE AUTOSKIP=YES
+1 "varname3"
+3 var4 1
ATTR=UNDERLINE AUTOSKIP=YES
+1 "varname4"
#11 @5 "Enter a specific title, if applicable:"
/ @5 titlex 40 ATTR=UNDERLINE

```

```

GROUP= verify1
#15 @7 "<> A BEGINNING date is REQUIRED.
Press [ENTER] to continue <>"
GROUP= verify2
#15 @7 "<> An ENDING date is REQUIRED.
Press [ENTER] to continue <>"
;

```

```

%WINDOW verify
#1 @20 "Ad Hoc Report: Criteria Gathering"
/ @25 "&SYSDAY &SYSDATE"
/ @20 "*****"
#5 @15 "You have chosen these criteria:"
#7 @15 "Beginning Date: " begndate
PROTECT=YES COLOR=BLUE
/ @15 "Ending Date: " enddate
PROTECT=YES COLOR=BLUE
/ @15 "SORT variables: " sortvars
PROTECT=YES COLOR=BLUE
/ @15 "Specific title: " titlex
PROTECT=YES COLOR=BLUE
#17 @7 "<> One more chance --- are you sure of
these choices (Y/N)?"
+2 ynsure 1
ATTR=UNDERLINE REQUIRED=YES +2
"<>"
;
%LET begndate = ;
%LET enddate = ;
%LET var1 = ; %LET var2 = ; %LET var3 = ;
%LET var4 = ;
%LET titlex = ;

```

```

%MACRO example3 ;
%LET null = ;
%lblstr: ;

%DISPLAY example3.hdrftr
      NOINPUT BLANK BELL ;
%DISPLAY example3.criteria ;

%IF &begndate EQ &>null
%THEN %DO ;
      %DISPLAY example3.verify1 ;
      %GOTO lblstr ;
%END ;
%ELSE %IF &enddate EQ &>null
%THEN %DO ;
      %DISPLAY example3.verify2 ;
      %GOTO lblstr ;
%END ;

%IF &var1 NE &>null
%THEN %LET var1 = varname1 ;
%IF &var2 NE &>null
%THEN %LET var2 = varname2 ;
%IF &var3 NE &>null
%THEN %LET var3 = varname3 ;
%IF &var4 NE &>null
%THEN %LET var4 = varname4 ;
%LET sortvars = &var1 &var2 &var3 &var4 ;

%DISPLAY verify BELL ;
%IF %UPCASE(&yensure) NE Y
%THEN %GOTO lblstr ;

%MEND example3 ;
%example3 ;

```

Since we are working with macro variables, with some creativity and a bit more work, a dynamic window could be created. The displayed text and underlying values could come from a data set; the positioning of the fields in the window could be determined based on the number of values. SAS/AF® software provides several means to accomplish this dynamic interface. Please refer to *SAS/AF® Software: Usage and Reference, Version 6, First Edition*. Remember though, that the present purpose is simply to gather criteria to produce reports, and everything presented here is available in the Base SAS® system.

CONCLUSION

I have presented two techniques for obtaining user criteria, the first using %PUT and %INPUT along with &SYSBUFFR, and the second using the %WINDOW and %DISPLAY statements. If the responses sought are single values, a date, one variable name, or one value, and you are using a mainframe version of the SAS® system prior to 6.09, then using %PUT and %INPUT is my first choice. It is simple and quick to code and virtually maintenance-free, depending on the prompts used.

However, if there are multiple, static choices or you want to present your clients with all the questions at once, the %WINDOW statement, while requiring more up front programming, is much more elegant, flexible, user-friendly, and comes closer to being error-free. Both of these methods are extremely useful and I trust they will prove to be valuable tools in meeting the reporting needs of your clients.

SAS and SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® Indicates USA registration.

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

REFERENCES

SAS Institute Inc. (1990), *SAS ® Guide to Macro Processing, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1989), *SAS/AF ®Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

Michael A. Mace
 1152 E 344 ST
 Eastlake, OH 44095-2942
 440-953-8924 (H)
 330-796-3981 (W)
 e-mail: michael.a.mace@worldnet.att.net