Paper 22-25

Using @WINDOW to Gather User Criteria

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, mainframe or PC, never has to edit any code. Several examples will demonstrate gathering single and multiple choices per parameter and conditionally gathering criteria.

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. Often, in the crunch of the request, a program is written with the specific criteria hard-coded. When the client wants the same report but for different criteria, someone must edit the code, find and modify the criteria, and maybe even test the changes. 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 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 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, because 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.

Please refer to *SAS ® Guide to Macro Processing, Version 6, Second Edition* for a thorough discussion about the macro facility, macro variables, %WINDOW and %DISPLAY, and all of their options.

%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. %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 one or more 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 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 strongly 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 @20 "Company Name: Title of screen"
   #5 @15 "Please enter the requested data, THEN press
[ENTER]"
   #7 @15 "Company (21, 41, 51): "
        @40 company 2  ATTR=UNDERLINE
REQUIRED=YES
   #9 @15 "Customer Premise ID: "
        @40 premise 7  ATTR=UNDERLINE
REQUIRED=YES
  #11 @15 "Billing Date (YYYYMM):"
         @40 yyyymm 6  ATTR=UNDERLINE
REQUIRED=YES
  ;
```

```
%MACRO macronam ;
%LET null = ;
%LET SYSMSG= Press [ENTER] only after filling in ALL
fields;
%lblstrt: ;

%DISPLAY wndoname ;

%IF (&company NE 21 AND &company NE 41 AND
&company NE 51)
   OR %LENGTH(&yyyymm) LT 6
%THEN %DO ;
         %LET SYSMSG= A value is in ERROR, please
correct, then <ENTER> ;
         %GOTO lblstrt ;
      %END ;

%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 = ;
%LET ansr = ;


%WINDOW vino
  IROW= 10  ICOLUMN= 10  ROWS= 20  COLUMNS= 60

GROUP= wines
   #4 @15 "Please chose which wine(s) you like: "
   #6 @20 vino1 1 ATTR=UNDERLINE  @22 "Burgundy"
COLOR=Magenta
   #7 @20 vino2 1 ATTR=UNDERLINE  @22 "Merlot"
COLOR= Pink
   #8 @20 vino3 1 ATTR=UNDERLINE  @22 "Zinfandel"
COLOR= Cyan
   #9 @20 vino4 1 ATTR=UNDERLINE  @22 "Reisling"
COLOR=Blue
  #15 " *** After entering choice(s), press [ENTER] "

GROUP= zinfandl
   #2 @15 "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 macronom ;

%LET null = ;
%lblstrt: ;


%DISPLAY vino.wines  BLANK ;


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


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

%LET ansr = &null ;


%DISPLAY verify  BELL ;


%IF %UPCASE(&ansr) NE Y
%THEN %GOTO lblstrt ;


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

%MEND macronom ;
%macronom ;
```

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. Notice the use of the &SYSFUNC function to convert macro date variables to SAS® dates. Also, i have used the new &SYSDATE9 system macro variable so that today's date is displayed with a four character year.

```
%LET early = 06APR1989 ;
%LET late = 31DEC2007 ;


%WINDOW adhoc
  IROW= 1  ICOLUMN= 1  ROWS= 25  COLUMNS= 75

  GROUP= hdrftr
   #2  @10 "Ad Hoc Report: Criteria Gathering"
     /  @20 "&SYSDAY   &SYSDATE9"
  #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= verifyB
  #15  @5 "<> A BEGINNING date is REQUIRED !"
     /   @8 "Press [ENTER] to continue ..."


  GROUP= verifyE
  #15  @5 "<> An ENDING date is REQUIRED !"
     /   @8 "Press [ENTER] to continue ..."


  GROUP= verifyBE
  #15  @5 "<> BEGINNING date Must be Later than "
early " !"
     /   @8 "Press [ENTER] to continue ..."


  GROUP= verifyEL
  #15  @5 "<> ENDING date Must be Earlier than " late "
!"
     /   @8 "Press [ENTER] to continue ..."


  GROUP= verifyEB
  #15  @5 "<> ENDING date must be EQUAL to OR
LATER than Beginning date !"
     /   @8 "Press [ENTER] to continue ..."
;
```

```
%WINDOW verify
  IROW= 3  ICOLUMN= 3  ROWS= 25  COLUMNS= 80

  #2  @20 "Ad Hoc Report: Criteria Gathering"
    /  @25 "&SYSDAY   &SYSDATE"
    /  @20 "**********************************"
  #7  @15 "You have chosen these criteria:"
  #9  @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
 #19   @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 adhoc ;
%LET null = ;
%lblstrt: ;
%LET ynsure = ;
%LET SYSMSG= Enter your criteria, TAB between fields,
Press <ENTER> to submit ;

%DISPLAY adhoc.hdrftr  NOINPUT BLANK BELL ;

%DISPLAY adhoc.criteria ;


%IF %SYSFUNC(INPUTN(&begndate, DATE9.)) EQ .
%THEN %DO ;
            %DISPLAY adhoc.verifyB ;
            %GOTO lblstrt ;
        %END ;


%IF %SYSFUNC(INPUTN(&begndate, DATE9.)) LT
%SYSFUNC(INPUTN(&early, DATE9.))
%THEN %DO ;
            %DISPLAY adhoc.verifyBE ;
            %GOTO lblstrt ;
        %END ;


%IF %SYSFUNC(INPUTN(&enddate, DATE9.)) EQ .
%THEN %DO ;
            %DISPLAY adhoc.verifyE ;
            %GOTO lblstrt ;
        %END ;
```

```
%IF %SYSFUNC(INPUTN(&enddate, DATE9.)) GT
%SYSFUNC(INPUTN(&late, DATE9.))
%THEN %DO ;
                %DISPLAY adhoc.verifyEL ;
                %GOTO lblstrt ;
          %END ;

%IF %SYSFUNC(INPUTN(&enddate, DATE9.))  LT
%SYSFUNC(INPUTN(&begndate, DATE9.))
%THEN %DO ;
                %DISPLAY adhoc.verifyEB ;
                %GOTO lblstrt ;
         %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 ;

%LET ynsure = &null ;
%DISPLAY verify  BELL ;

%IF %UPCASE(&ynsure) NE Y
%THEN %GOTO lblstrt ;

%MEND adhoc ;
%adhoc ;
```

Of course, 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

%WINDOW statement provides, in both mainframe and PC environments, an elegant, flexible, user-friendly, and close to error-free means by which to gather user criteria for repetitive, customized, or ad hoc reports.

REFERENCES

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