

Using SAS/AF® FRAME Entries for Data-Driven Menu Systems

Stuart B. Levine

Consulting Services Department, SAS Institute Inc.

ABSTRACT

One basic application of FRAME technology is a menuing system. A menu can further be enhanced by being developed to populate itself from a data set. This makes changes to the menu dependent on maintaining a data set rather than having to modify code. This Coder's Corner session will demonstrate an example of this type of menu application.

APPLICATION MENUS

Most interactive applications will be made up of multiple modules, most of which can be accessed via a primary menu. There are two ways of developing these menus. The first, and unfortunately more common, is to 'hard-code' the menu by placing text on a PROGRAM or FRAME entry and attaching SCL code to those text items to call another entry when selected. This can be the easier development method at first glance, but maintenance can become a nightmare as text items need to be changed or items have to be added.

The second type of menu that can be developed is a data-driven menu, where the screen is generic and the menu items are controlled by a SAS data set. This can be accomplished in a number of ways, including block menus (using the BLOCK function) and extended tables (using a variety of FRAME objects and the PUTROW section). Following is an example of a data-driven menu system using an extended table and the IMAGE ICON widget available in Release 6.11 of the SAS System.

THE MENULIST DATA SET

The data set used to generate this menu can be created using the following DATA step:

```
data libname.menulist;
  attrib menutext length=$44
    label='Text to display on the selection list';
  attrib callprog length=$35
    label='4-level name for selected program';
  attrib child length=$8
    label='Menu entry program name';
  attrib program length=$8
    label='Entry to call when this option selected';
  attrib order length=8
    label='Sort order to display entries in';
```

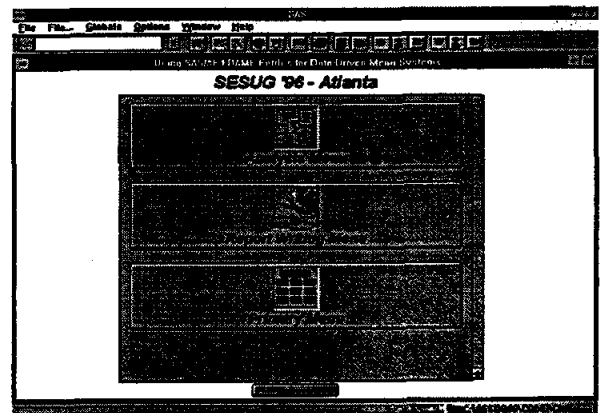
```
attrib imagename length=$8
  label='Image name for icon';
attrib acclevel length=8
  label='User access level';
set libname.menulist; /* If the data set already exists. */
run;
```

An example of an observation from this data set would be:

```
IMAGENAME:  DESNRPRT
MENUTEXT:    Generate Custom Standard Reports
CALLPROG:    SYSTEM.WINDOWS.STDRPTS.FRAME
ENTRY:       REPORTS
PROGRAM:     STDRPTS
ORDER:       20
ACCLEVEL:    6
```

THE MENULIST FRAME AND SCL PROGRAM

The FRAME entry for this menu is very simple, comprised of three widgets: an Extended Text Entry (optional for a screen title), an Extended Table, and a Command Push Button (also optional). The Extended Table widget contains only an Image Icon. It looks like this:



The SCL program that controls this frame follows:

```
entry optional = current $17;

length program $8 /* Sub-menu name. */
  callprog $35 /* 4-level entry name called. */
  child $8 /* Entry name called. */
  menutext $45 /* Text for icon. */
  imagename $8 /* Name of image for row. */ ;

INIT:

  /* To eliminate COMPILE messages. */
  program=program;
  callprog=callprog;
  child=child;
```

/* SCL code continued on next page...*/

```

rc=copy('PARMDL.MENULIST',WORK.MENULIST);

dsid=open('WORK.MENULIST(WHERE=(ACCLEVEL<=!!
          putn(symgetn('USERACC'),'BEST2.'))'),1);

/* Need to open the data set twice. the second is so that when */
/* a user makes a choice, the program can check whether this */
/* entry is called again to display a submenu or whether to */
/* call an AF entry whose name is the value of PROGRAM. */
next=open('WORK.MENULIST');

call set(dsid);
call set(next);

if current=' ' then
do; /* At top level of menu. */
  call symputn('MENULEVL',0);
  current='MENULIST';
end; /* At top level of menu. */

call symputn('MENULEVL',symgetn('MENULEVL')+1);

/* Subset selection list for current menu. */
rc=where(dsid,'CHILD="||current||"');

RETURN; /* INIT */

TERM:

if dsid then
  dsid=close(dsid);
if next then
  next=close(next);

call symputn('MENULEVL',symgetn('MENULEVL')+(-1));
if symgetn('MENULEVL')<=0 and
  exist('WORK.MENULIST') then
  rc=delete('WORK.MENULIST');

RETURN; /* TERM */

GETROW:

rc=fetchobs(dsid,_currow_);
if rc=0 then
do; /* Observation was fetched. */
  call notify('ROW',_SET_LABEL_,menutext);
  if imagename ne '' then
    call notify('ROW',_SET_IMAGE_,imagename);
end; /* Observation was fetched. */
else
  call notify('MENUTABL',_ENDTABLE_);

RETURN; /* GETROW */

PUTROW:

call notify('ROW',_GET_LABEL_,menutext);
choice=locatec(next,vamum(next,'MENUTEXT'),menutext);
rc=fetchobs(next,choice);
current=program;

selected=locatec(next,vamum(next,'CHILD'),program);
if selected<=0 then
do; /* Call selected program/frame. */
  call symput('MENUTEXT',menutext);
  call display(callprog);
end; /* Call selected program/frame. */
else
  call display('MENULIST.FRAME',current);

call notify('MENUTABL',_UNSELECT_ROW_,_currow_);

RETURN; /* PUTROW */

```

```

GOBACK: /* GOBACK push button. */

if symgetn('MENULEVL')<2 then
do; /* At primary menu level. */
  link TERM;
  call execcmd('END');
end; /* At primary menu level. */
else
  call execcmd('END');

RETURN; /* GOBACK */

```

The system works using a parent-child relationship. The data set has been subset based on the highest level 'parent', displaying only the first level of 'children' (the variable CHILD) on the main menu. When an item is selected from the menu and the PUTROW section executes, if any 'children' exist (ENTRY), then the entry calls itself and populates the table with a new subset of observations. If there are none, then the desired program is called (CALLPROG).

The variable ACCLEVEL is used to further subset the list of available menu items by user access level. This allows all possible items to be in the same data set for any level of user rather than having two separate applications, one for the users and a second for system administrators or other people who are responsible for maintaining the system.

This menu system can also be text-based if necessary by replacing the Image Icon object in the Extended Table with one of the available text objects. The data set would not have to be modified and very little code would have to be changed. Only the statements in the GETROW and PUTROW sections that are specific to the Image Icon widget.

CONCLUSION

A data-driven menu system (or any part of an application) can make a system administrator's job much easier. It allows them to control or maintain an application by maintaining a data set rather than having to modify code to change a menu or other selection list. Also, a data-driven application can be more flexible, limited only to the imagination of the developer to add modules. New modules can be integrated into an application simply by adding an observation to the menu data set.

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