SAS/AF® FRAME Entries: A Hands-on Introduction

Vincent L. Timbers
The Pennsylvania State University, University Park, Pa.

ABSTRACT

Frame entries in SAS/AF use graphic display devices that enable application developers to easily build interfaces which permit users to point and click their way through applications. The functionality of frame entries allow end-users with limited computer experience to easily perform activities such as data entry, data processing, and data analysis/reporting.

This paper briefly explains frame entry concepts and demonstrates the steps used in building a sample reporting application with frame entries.[1]

INTRODUCTION

SAS/AF frame entries allow developers to build graphic-oriented applications by placing objects such as icons, push buttons and list boxes in a window with the different objects having different functions. *Graphic text* and *text label objects* display text which can not be changed by the user. *Text entry objects* accept user input and display information or program output. *Icon* and *push button objects* are used for making selections or executing functions. *List box objects* display lists of text from which users can make selections. These are only a few of the objects available and application developers may also create their own objects. Frame entries are controlled by associated Screen Control Language (SCL) entries which contain SCL programming.

The following step-by-step sample application was developed to illustrate the basic steps used in building applications with frame entries. This example provides a working application designed for developers who have little or no experience with frame entries. Although the techniques used in the example are very basic, they provide a basis for developing more complex applications.

The sample application produces two reports. One report provides a proc means of student loans for a specific level of enrollment and the other report produces a proc means for student grants for a specific level of enrollment. The application includes two frame entries and their associated SCL entries. The first frame is the opening menu, which includes the application title and three icons. Selecting the first two icons will branch the application to a report frame from which information for the report is collected and the report is submitted for processing. The third icon exits the application.

The second frame entry is used for producing both reports. This frame contains five push buttons, one list box and one text entry object. One button displays a list box from which the level of enrollment is selected to subset the data for the report. The other four push buttons are for submitting the report for processing, printing the output, clearing the output window, and going back to the opening menu. The text entry object displays the level of enrollment selected for the report.

FRAME ENTRY OBJECT TERMINOLOGY AND LOGIC

Before starting the sample application, a brief review of frame entry terminology and logic may be helpful. For a more in-depth explanation see the listed references.

When building a frame entry, objects are selected from templates known as *classes* and put in the frame. *Classes* identify the actions which objects can perform. These actions are known as *methods.* Classes also identify the object *attributes* such as the name, color, and size of the object. These attributes are stored in what are called *instance variables*. Each type of object, such as an icon or push button has its own class, *icon class* and *push button class.* These classes are subclasses (or children) of a parent class called the *widget class*, which is a subclass of the *object class*. The importance of the class hierarchy is that subclasses inherit the attributes and methods of their parents and may have additional methods or attributes of their own. Note that objects such as icons and push buttons are often called *widgets* - a widget being a displayable object. However, in this paper displayable objects will simply be referred to as objects.

BUILDING THE SAMPLE APPLICATION

The first step in building the application is to invoke the AF Build procedure by issuing the **BUILD** command from the command line. This command will open a

[1]The sample application presented in this paper was developed with SAS® Release 6.12 for the Windows operating system.

window containing the library, catalog and entry lists as seen in Figure 1.

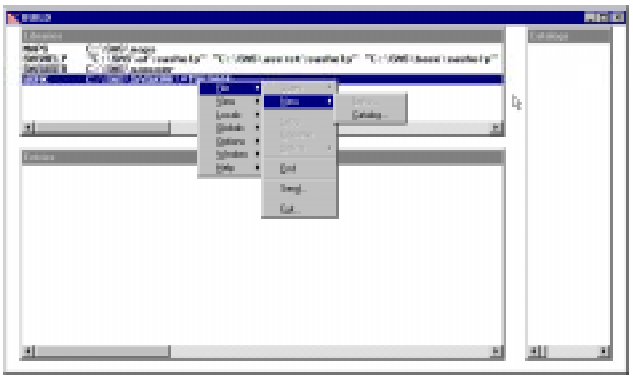

Figure 1: Build Library, Catalog and Entry list window

To create a new catalog for the frame and SCL entries in this example, highlight the **SASUSER** library and from the menu bar choose **FILE, NEW** and **CATALOG**. In the Catalog window, enter the name **SUGI23** and select **OK**.

MAIN MENU FRAME

The first frame entry is created by highlighting the **SUGI23** catalog and choosing **FILE, NEW** and **ENTRY** from the menu bar. In the New Entry window, enter the name **MENU**, select the entry type **FRAME** and select **OK** to open the Build: Display window for building the Menu.Frame entry. When a frame entry is first created, the Build: Display window only contains a master region bordered with solid or broken lines which encompass the entire window. The objects put in a frame entry are contained within regions which are placed inside the master region. The Build: Display window for the Menu.Frame entry is displayed in Figure 2, as it will look after the objects are placed in the frame.

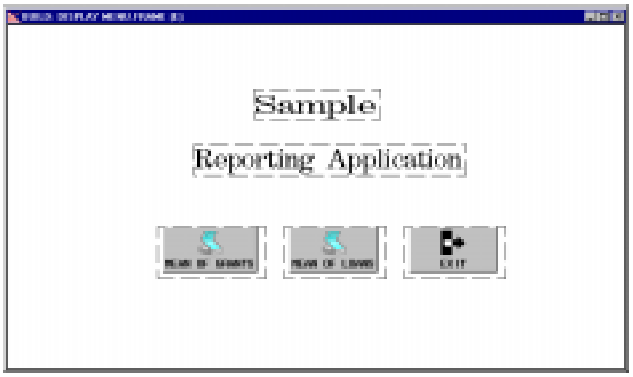

Figure 2: Menu.Frame Build Display window

The Menu.Frame entry in this sample application contains two graphic text objects for the title of the application and three icon objects, which upon selection go to a report frame or exit the application.

The first graphic text object containing the first line of the title is created by selecting **MAKE**, **GRAPHIC TEXT** and **OK** from the master region pop-up menu. After making these selections a new region outlined with broken lines will appear in the master region. Using the mouse, place the new region in the upper middle of the screen with a mouse click. After placing the object, a Graphic Text Attributes window will appear as shown in Figure 3.

Figure 3: Graphic Text Attributes window

The object attributes such as the object name, text and color are entered in this window. For the first graphic text object in this example, enter the following attributes.

Name	Text
Title1	**SAMPLE**

After entering the attributes select **OK** to exit the Graphic Text Attributes window.

*Note: If you need to re-open an Object Attributes window select **OBJECT ATTRIBUTES** from the object region pop-up menu. Open the pop-up menu with a right mouse click in the object's region.*

The second graphic text object, containing the second line of the title, is created the same way the first graphic text object was created using the following attributes.

Name	Text
Title2	**Reporting Application**

Creating the icon objects for the Menu.Frame entry is very similar to creating the graphic text objects. The only differences are choosing **ICON** from the selection list instead of graphic text and selecting a specific icon from the attributes window. To create the GRANT icon select **MAKE**, **ICON** and **OK** from the master region pop-up menu. After placing the icon, enter the following name

and label attributes in the Icon Attributes window as shown in figure 4.

Name	Label	Icon Number
Grant	**MEAN OF GRANTS**	**124**

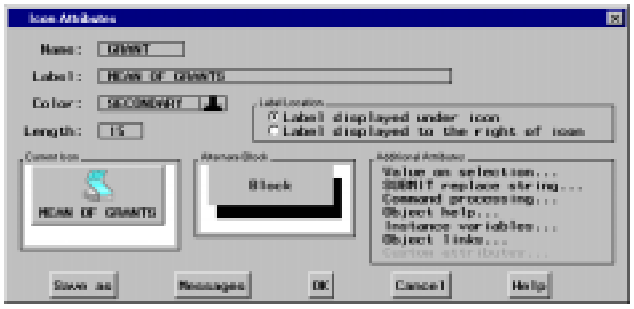

Figure 4: Icon Attributes window

To choose the icon, click the Current Icon area of the Icon Attributes window. (Different areas of an Attributes window are bordered with a single line and identified with the name in the upper left corner.) Once in the Select an Icon window you can scroll through a list of numerically identified icons and make a selection or enter the number of the icon you want (124 for this icon). Then select **OK** to exit the Select an Icon window and **OK** again to exit the Icon Attributes window.

Repeat the process used for the GRANT icon to create the LOAN and EXIT icons using the following attributes.

Name	Label	Icon Number
Loan	**MEAN OF LOANS**	**124**
Exit	**EXIT**	**111**

The EXIT icon requires an additional step. After entering the name and label attributes and selecting the icon an additional attribute needs to be entered to tell SAS what command to execute when the icon is selected. Choose **COMMAND PROCESSING** from the Additional Attributes area of the Icon Attributes window. This will open the Command Processing window where you will type **CANCEL** in the Execute SAS Commands on Selection area. Select **OK** to exit the Command Processing window and **OK** again to exit the Icon Attributes window.

Now that the Menu.Frame entry in this example is created, the SCL entry that controls the frame can be created. Select **LOCALS** and **EDIT SCL SOURCE** from the menu bar. In the Build: Source window that opens, enter the following lines of SCL.

```
INIT:
return;

GRANT:
 name='meangrnt';
 call display('report.frame',name);
return;

LOAN:
 name='meanloan';
 call display('report.frame',name);
return;

MAIN:
return;

TERM:
return;
```

Notice that the Menu.SCL entry has five sections, INIT, GRANT, LOAN, MAIN and TERM. Each section begins with the section name followed by a colon and ends with a return statement. The INIT, MAIN and TERM sections are the three basic sections which many but not all SCL entries may have. The INIT section contains programming that is executed when the frame entry is initiated and the TERM section contains programming that is executed when the frame entry is terminated. The MAIN section will execute any time the Enter key is pressed or an object in the frame is selected with the mouse. However, these sections do not contain any programming because it is not needed for the Menu.Frame entry of this application. When there is no programming in these sections, they do not need to be included in the entry. The GRANT and LOAN sections are executed when the corresponding icons in the frame are selected. Both sections contain two lines of code. The first line sets the variable *name* equal to the name of the report being requested. The second line, **call display('report.frame', name)**, calls the Report.Frame entry which will be created next. Notice the parameter *name*. This passes the variable *name* to the frame being called. The GRANT and LOAN sections of this SCL entry are typical of program sections which correspond to frame objects. *When an object is selected the section of SCL with the same name is executed.*

After typing in the SCL, compile the program by selecting **LOCALS** and **COMPILE** from the menu bar. After the compile is completed, a message will appear at the bottom of the Build: Source window reporting the results of the compile. If there are errors they will be recorded in the Message window. When the SCL is successfully compiled, exit the Build: Source and Build: Display windows and return to the main Build window.

REPORT FRAME

Now that the frame entry and corresponding SCL entry for the main menu of the sample application are created, the entries for the report frame need to be created. The Report.Frame entry is created the same way as the Menu.Frame entry. With the **SUGI23** catalog highlighted, choose **FILE, NEW** and **ENTRY** from the menu bar. In the New Entry window enter the name **REPORT**, select the entry type **FRAME** and select **OK**. The empty Build: Display window for the Report.Frame entry will appear. When complete this frame will contain five push button objects, one text entry object and one list box object as shown in Figure 5. The push button labeled LEVEL displays a list box for choosing the level of enrollment for the report while the text entry object displays the level selected. Prior to the selection of a level the text entry box displays "REQUIRED" indicating a level of enrollment selection is required. The buttons labeled RUN, PRINT, CLEAR OUTPUT and GO BACK do just what their labels indicate.

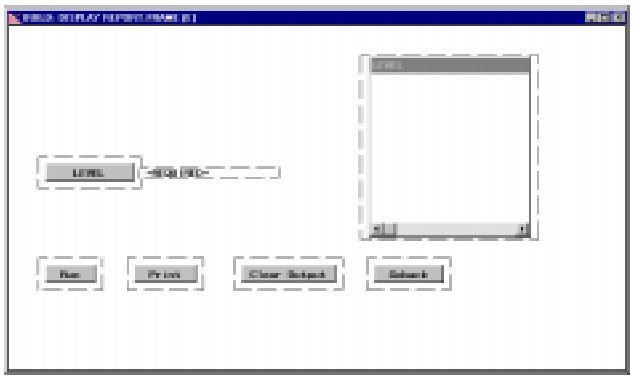

Figure 5: Report.Frame Build: Display window

Objects in this frame are created the same way objects were created in the previous frame. Select **MAKE**, the type of object (**PUSH BUTTON**, **TEXT ENTRY**, or **LIST BOX**) and **OK** from the master regions pop-up menu. Use the following attributes for the push button objects.

Name	Label
Levelpb	**LEVEL**
Run	**RUN**
Print	**PRINT**
Clear	**CLEAR OUTPUT**
Goback	**GO BACK**

Like the EXIT icon in the Menu.Frame entry, additional information needs to be entered in the attributes window to tell SAS what command to execute when the GO BACK push button is selected. As before, choose **COMMAND PROCESSING** from the Additional Attributes area, then type **CANCEL** in the Execute SAS Commands on Selection area of the Command Processing window. Select **OK** to exit the Command Processing window and **OK** again to exit the Push Button Attributes window.

When creating the text entry object the Text Entry Attributes window will appear as shown in Figure 6.

Figure 6: Text Entry Attributes window

Enter **LEVELTE** as the Name attribute for this text entry object. Now an initial value needs to be assigned to the object. From the Additional Attributes area of the window select **INITIAL VALUES/FORMATS**. In the Initial Values/Formats window as shown in Figure 7, enter **–REQUIRED–** in the Value area of the window and select **CHARACTER** in the Data Type area of the window. Select **OK** to exit the Initial Values/Formats window and **OK** again to exit the Text Entry Attributes window.

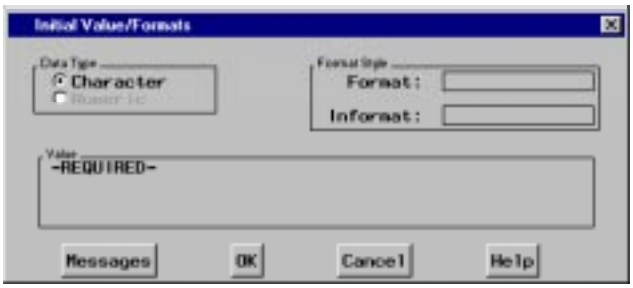

Figure 7: Initial Value/Formats window

Select **MAKE, LIST BOX** and **OK** from the master regions pop-up menu to create the list box object. After placing the object, the List Box Attributes window will appear as shown in Figure 8.

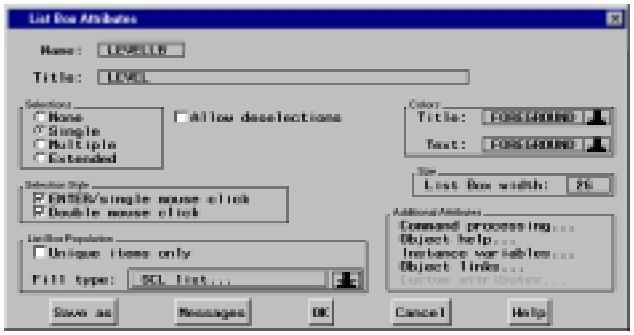

Figure 8: List Box Attributes Window

In the Attributes window enter the following attributes.

Name	Title
Levellb	**LEVEL**

There are many methods of populating list boxes which include entering the values as attributes or telling the list box to populate itself from another source such as a SAS data set or a SCL list. In this application we are going to use a SCL list (to be created later). Populating the list box is done by clicking on the down arrow to the right of **Enter values ...** in the List Box Population area of the List Box Attributes window. Then select **SCL list...** to display the SCL List window as shown is Figure 9.

Figure 9: SCL List Window

Enter **LEVELIST** as the list name and select **OK** to exit the SCL List window and **OK** again to exit the List Box Attributes window.

As with the other frame entry in this application, after the objects are created the SCL needs to be entered in the SCL entry for the frame. Open the Build: Source window by selecting **LOCALS** and **EDIT SCL SOURCE** from the menu bar and enter the following SCL.

```
ENTRY NAME $ 20;

INIT:
   levelist=makelist();
   rc = insertc(levelist, "UNDERGRAD", 1 );
   rc = insertc(levelist, "GRAD", 2 );
   rc = insertc(levelist, "MEDICAL", 3 );
   call notify('levellb','_hide_');
   if name='meangrnt' then
   call notify ('.', '_set_title_', 'PROC MEAN OF GRANT
   AID');
   else if name='meanloan' then
   call notify ('.', '_set_title_', 'PROC MEAN OF LOAN
   AID');
return;

LEVELPB:
   call notify('levellb','_unhide_');
return;

LEVELLB:
   call notify('levellb','_get_last_sel_',row,issell,levelte);
   call notify('levellb','_hide_');
return;

RUN:
   if levelte='-REQUIRED-' then do;
      _msg_='You must select a level before running the
      report!!';
      return;
   end;
   _msg_='Please wait while your request is processed.';
   refresh;
   call display('means.scl', name, levelte);
return;

PRINT:
   rc=woutput ('print','sasuser.profile.default');
   rc=woutput ('clear');
return;

CLEAR:
   rc=woutput ('clear');
return;

TERM:
   rc=woutput('clear');
return;
```

The Report.SCL entry contains the line **ENTRY NAME $;** and seven sections of code labeled INIT, LEVELPB, LEVELLB, RUN, PRINT, CLEAR and TERM. The first line is necessary for passing the variable *name* between the Report.SCL entry and the previous SCL entry, Menu.SCL. The first line in the INIT section creates a SCL list named LEVELIST and the following three lines insert items into the list. A SCL list is an ordered collection of data. SCL lists are very powerful data structures which can be used for many purposes. In this application the SCL list is used to populate the LEVELLB list box with the level of enrollment values.

Earlier it was mentioned that objects can perform actions which are known as methods. The next three statements contain method calls. The line **call notify ('levellb', '_hide_');** calls the list box method , _hide_, which instructs the list box to hide itself. All call notify statements take this form. The first element inside the parentheses is the name of the object being called and the second element is the name of the method to be executed. The second and third statements with method calls check the value of the variable *name* (received from the previous SCL entry) and notifies the current frame entry

to set the text in the title bar. Notice that the first element of the call notify is **'.'**. The period identifies the current frame entry as the object to be notified, while _set_title_ is the method and the third element is the parameter containing the text for the frame title.

The LEVELPB section is executed when the push button named LEVELPB is selected. This section contains a line which tells the list box named LEVELLB to unhide itself. The LEVELLB section contains two lines of code which retrieve the list box selection and hide the list box when the selection is made.

The RUN section is executed when the RUN push button is selected. This section first checks if a level of enrollment is selected and warns the user if not. If a level has been selected the program sends the user a message indicating the report is being processed. Finally, the SCL entry containing the programming to produce the report is called with the variables containing the report name and level of enrollment being passed.

The PRINT section prints the contents of the output window and then clears the output window. The CLEAR section clears the contents of the output window and the TERM section clears the output window when the frame is terminated.

After typing in the SCL, compile the program by selecting **LOCALS** and **COMPILE** from the menu bar. When the SCL is successfully compiled, exit the Build: Source and Build: Display windows and return to the main Build window.

MEANS.SCL ENTRY

The last entry for this sample application is the SCL entry from which the reports are produced. This entry is not associated to a frame entry and is called with a call display statement from the Report.SCL entry previously created (last line of the RUN section). This SCL entry is created the same way the previous frame entries were created. With the **SUGI23** catalog highlighted, choose **FILE, NEW** and **ENTRY** from the menu bar. In the New Entry window enter the name **MEANS**, select the entry type **SCL** and select **OK**.

Enter the code listed below in the Means.SCL Build Source window and compile the program by selecting **LOCALS** and **COMPILE** from the menu bar.

```
ENTRY NAME LEVEL $ 10;

INIT:
   if name='meangrnt' then do;
      submit continue;
      OPTIONS PAGENO=1;
      TITLE;
      TITLE1 'PENN STATE OFFICE OF STUDENT
      AID';
      TITLE2 'MEAN GRANT REPORT';
      PROC MEANS DATA=SASUSER.TEST;
      WHERE LEVEL="&LEVEL" AND GRANT>0;
      VAR GRANT;
      RUN;
      endsubmit;
   end;

   else if name='meanloan' then do;
      submit continue;
      OPTIONS PAGENO=1;
      TITLE;
      TITLE1 'PENN STATE OFFICE OF STUDENT
      AID';
      TITLE2 'MEAN LOAN REPORT';
      PROC MEANS DATA=SASUSER.TEST;
      WHERE LEVEL="&LEVEL" AND LOAN>0;
      VAR LOAN;
      RUN;
      endsubmit;
   end;
return;
```

This SCL entry contains an entry statement and an INIT section. The entry statement is used for passing the *name* and *level* variables between this and the previous entry. The INIT section contains the code to be submitted for processing for both reports. The appropriate programming is submitted based on the value of the *name* variable. As mentioned before, this SCL entry is not associated with a frame entry. Therefore, any code to be executed must be in the INIT section, TERM section or called from the INIT section. Again, the INIT section is executed when the entry is initiated. Another important part of this entry is the submit blocks containing the code which produces the reports. The submit blocks begin with SUBMIT CONTINUE and end with ENDSUBMIT statements. The code between these to statements is submitted to the SAS System for processing.

After the code is compiled, exit the Build: Source window to go back to the main Build window. From here the application can be tested by highlighting the Menu.Frame entry and selecting **LOCALS** and **TESTAF** from the menu bar. The Testaf procedure is similar to running the application, however the reports will not be submitted for processing. The Testaf procedure will record any errors in the application in the Message window.

When the application building and testing is completed close the Build window and return to display manager. The application may be invoked by issuing the command

AF C=SASUSER.SUGI23.MENU.FRAME from the command line.

This sample application uses a SAS data set named **TEST** in a SAS library named **SASUSER**. The data set contains student observations with the variables *loan*, *grant* and *level*. The *loan* and *grant* variables contain the amount of loans and grants each student has and the character variable *level* contains the students level of enrollment with the values GRAD, UNDERGRAD and MEDICAL. To use the sample application as presented in this paper, a SASUSER.TEST data set must be created prior to submitting the reports for processing.

SUMMARY

This step-by-step sample application illustrates the basic steps used in building a data reporting application with SAS/AF frame entries. The application could be described as an interface that collects user input to create SAS programming that is submitted for processing. While the techniques used in the application are very basic, they provide the basis for developing more complex applications. For more in-depth information on SAS/AF frame entries and Screen Control Language, see the following references.

REFERENCES

SAS Institute Inc. (1993). *SAS/AF Software: Frame Entry Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1993). *SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

The author may be contacted at:

The Pennsylvania State University
314 Shields Building
University Park, PA 16802
Phone: (814) 863-0700
E-mail: VLT@PSU.EDU