Getting Started With Frame Technology

Christopher A. Roper, Qualex Consulting Services, Inc., Hillsville, Va.

Introduction

Creating SAS/Frame applications allows users to build SAS applications using an Object Oriented Programming environment. This offers considerable advantage over the more traditional tools since you have routines/objects already defined to aid your development. With SAS version 6.12 there are many objects for displaying data/graphs and actions that let programmers develop a Graphical User Interface (GUI) relatively quickly. It also provides extensive programming control through the use of the SAS Screen Control Language (SCL) to allow users to interact with the GUI.

Creating a Catalog

First we need to create a catalog to store the Frames we are creating. You will need to have the SAS/AF module licensed on your computer to do this. If you are not sure if you have SAS/AF, execute the following code from the Program Editor:

```
Proc setinit;
Run;
```

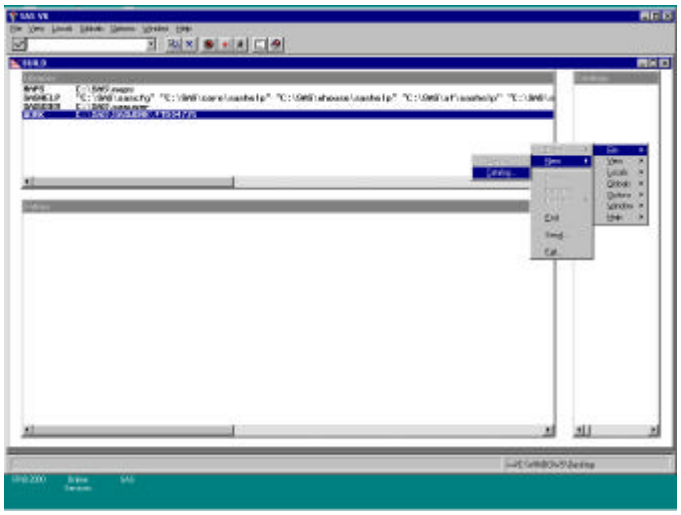
In the SAS Log window, you should see something like the following:

```
Original site validation data
Site name:     'QUALEX CONSULTING SERVICES INC'.
Site number:   99999999.
Expiration:    30JUN99.
Grace Period:  62 days (ending 31AUG99).
Warning Period: 31 days (ending 01OCT99).
System birthday:    23NOV92.
Operating System:    WIN      .
Product expiration dates:
---ACCESS TO R3                    30JUN99 (CPU A)
---ENTERPRISE REPORTER             15JUL99 (CPU A)
---BASE Product                    30JUN99 (CPU A)
---SAS/AF                          30JUN99 (CPU A)

NOTE: The PROCEDURE SETINIT used 0.59 seconds.
```

If you see the reference to SAS/AF, you are all set to begin. A SAS Catalog is a repository of various SAS file formats, and for this paper we will discuss SAS/AF catalogs. A SAS/AF catalog is the repository for all the various entries for a SAS application using AF. These entries can be text based, such as SOURCE, SCL, and PROGRAM entries, or they can be graphical, such as Frame or CBT entries. Some entry types can be either, such as CLASS entries, but those are beyond the scope of this paper. In any case, before you can create a new entry, you must first create the catalog. This is done by executing the BUILD display manager command from any SAS command line or command box. This will present the SAS BUILD Window, and it will look something like:

(Figure 1)



By placing cursor in the Catalog list box (on the right part of the Build window), you can press the right mouse button, select File→ New→ Catalog and get a dialog box to enter the name of a new catalog. Once you have entered a name, and selected OK, the catalog is created.

Creating A Frame Entry

Now, by placing the cursor in the Entries listbox (on the lower half of the Build window), pressing the right mouse button, and selecting File→ New→ Entry, you will get a dialog box to enter the name of a new entry, plus a drop down list of entry types. The default is Frame, so for this example, that does not need to be changed. Once you have entered the name and selected OK, the Frame entry is created. Now you are in the environment that lets you create a GUI using predefined objects and customize them to suit your application needs.

What's a Widget?

Widget is a generic name for any graphical object that you might want to use in an application frame. For this example we will start with a Tab Object, and then place a Data Table Object and a Graphics Object inside the Tab Object. Then we will tie them all together with some SCL, so we can control how the frame behaves.

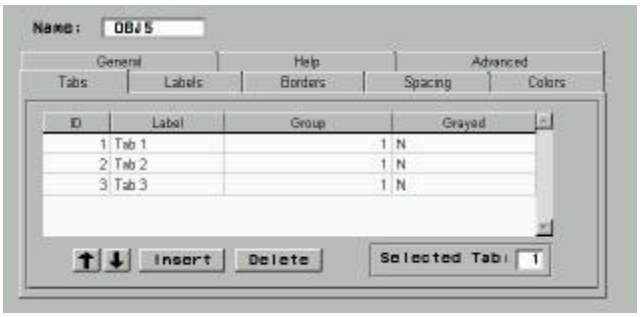
Making The Tab Object

First lets select the Tab Layout object from the Make menu. Select Actions from the menu on the top of the window, and select Make. This will present the default list of widgets available to you. Select Tab Layout from this list, then OK. You will see a box with a dashed outline that moves around the window as you move the mouse, place the box where you want it to be and press the left mouse button. This will anchor the Tab Object in this location and present the Object Attributes window for the Tab Object.

Tab Object Attributes Window

You will notice by default we have 3 tabs within our tab object and the tab labels are set to Tab 1, Tab 2, and Tab 3. We want to have a tab for the Data Table Object that we will add later, and a tab for the Graphics Object, but we don't want the third tab. So lets get rid of the third tab.

(Figure 2)



Click somewhere on the line containing "Tab 3" under the Label column. This will highlight that row, now select the Delete push button. This removes the third tab. Next we want to assign some meaningful names to the two remaining tabs. Select the second and rename the label to Graph and change the first one to Data. Now we have made a couple of simple changes to the object that will relate to our specific application. Select OK to complete the changes. To make the widget a little larger select the side and drag it until it is the desired size.

Making The Data Table Object

Once we have our Tab object we can add the data table object to it by placing the cursor over the object, selecting the right mouse button, then selecting Client , Make, Data Table. Once again we have the object attributes window, this time for the Data Table object.

Data Table Object Attributes Window

In the object attributes window for the Data Table object, we want to set the dataset that will be displayed by the Data Table. Selecting the TABLE right arrow button allows us to set the data that the Data Table Object will display. For this example, lets choose the dataset SESUG.DEMODATA.

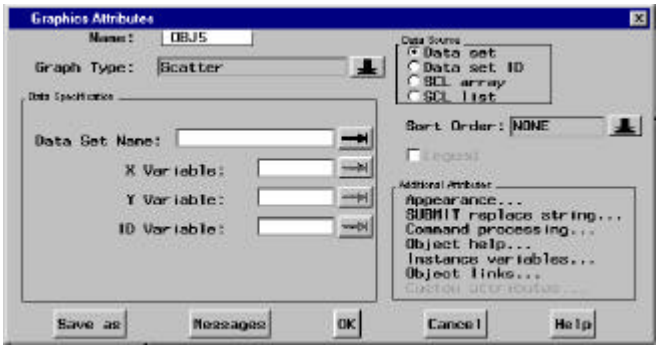
(Figure 3)

Making The Graphics Object

Now for the Graph, we will select the Graph Tab (making it the active region), and choose Client and Make. This time we select the Graphics object. Again, the object attributes window displays.

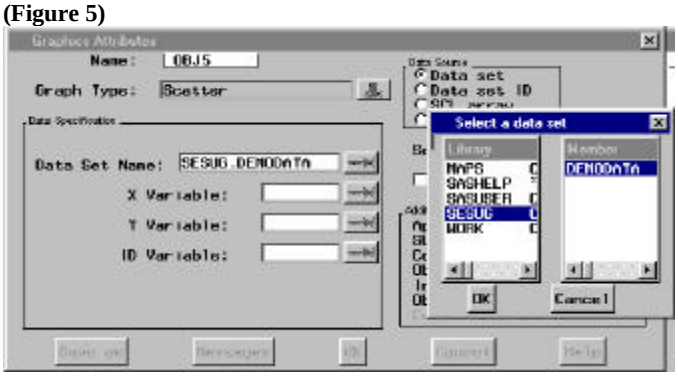
Graphics Object Attributes Window

(Figure 4)



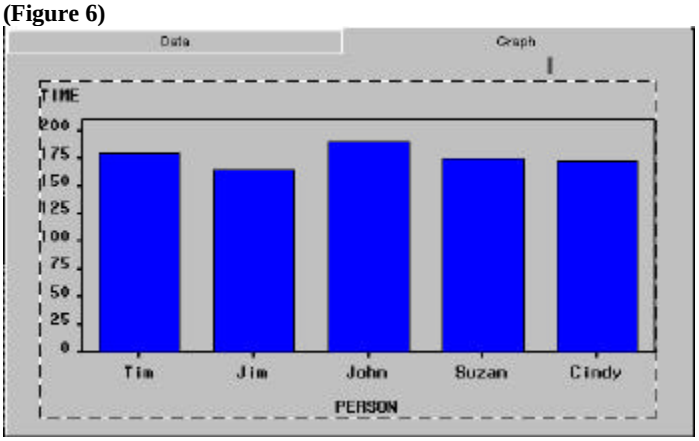
In the object attributes window for the Graphics object, we want to set the dataset that will be graphed by the Graphics object, just as we did for the Data Table object. Selecting the Data Set Name right arrow button allows us to set the data that the Graphics object will display. For this example, lets choose the dataset SESUG.DEMODATA.

(Figure 5)



After selecting the data set we need to choose the variables we wish to display in the graph. The SESUG.DEMODATA dataset has two variables, PERSON and TIME. Set PERSON as the X Variable(Independent variable) and TIME as the Y Variable (Dependent variable). We can also select the type of graph we wish to display. For this example, change the Graph Type from the default Scatter, to a Vertical Bar chart. Selecting OK displays the results of these changes.

(Figure 6)

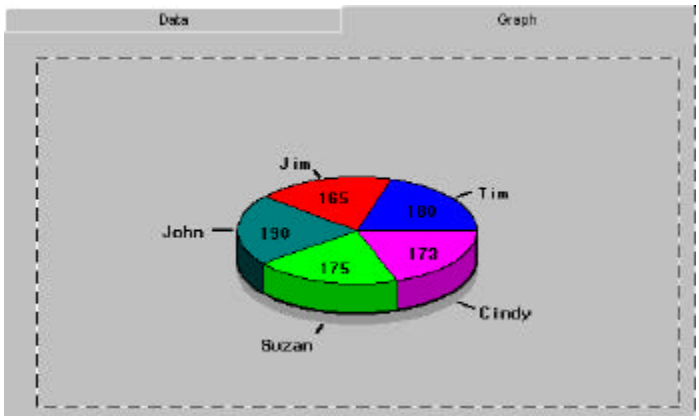


SCL Code

The object attributes can also be controlled by Screen Control Language (SCL) to allow the application user to change the attributes of the object such as the data set it uses or the graph type that is displayed. This is accomplished by using SCL to execute METHODS, which are stored routines that communicate instructions to the intended object.

The SCL code for this example allows the user to change the type of graph being displayed from a bar to a pie chart, using the method _SET_GRAPH_TYPE. These methods are already defined and have specific parameters to control the object. You can also create custom methods to add even more functionality to your application, but that is beyond the scope of this paper. Documentation of the valid methods for an object are available in the SAS online help, as well as the object's attribute window which also explains the valid parameters you can use for these method calls. In the example of the GRAPHICS widget the _SET_GRAPH_TYPE_ method accepts one parameter, an integer ranging from 1 through 25. Value 8 sets it to a Bar Chart and 16 to a 3D Pie Chart. Figure 7 shows what the Graphics object would look like if the SCL entry executed a _SET_GRAPH_TYPE_ method for the Graphics object to change to a 3D pie chart.

(Figure 7)



SCL Entry

The SCL Entry is the program that controls the execution and the logical flow of the Frame. Although an SCL entry is not required to create a Frame, SCL provides the functionality to make the Frame useful and practical. In practice, all Frames usually have an associated SCL entry to control their behavior. SCL code is usually divided into five main types:

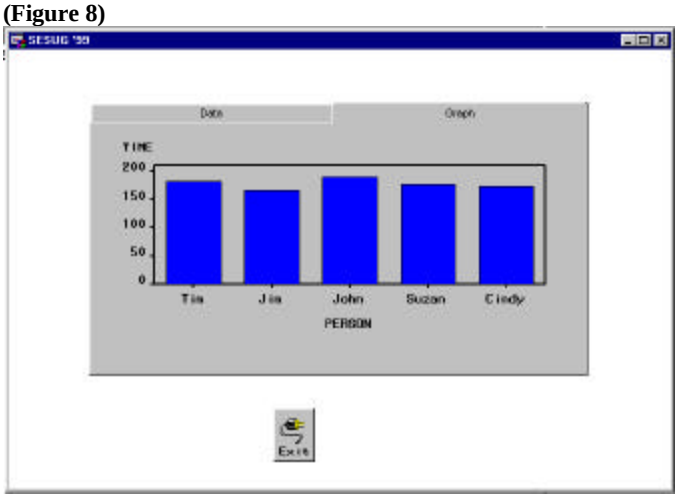
- **INIT**
- **MAIN**
- **TERM**
- **Object Labels**
- **Routines**

The INIT section executes on the initialization of the Frame and can be used to set up the environment for the Frame. MAIN executes between user interactions with the Frame; every time the user presses the enter key or selects a widget on the Frame, the MAIN section executes. The TERM section executes upon exiting the Frame and is an ideal place to perform housekeeping function. The OBJECT label sections are sections of code specific to individual widgets in the window. They execute on user interaction with the same named objects within the Frame. Routines are labeled sections of SCL similar to Object label sections, except they are not associated with any like named widgets on the Frame, they are used as code modules within the SCL entry.

Example Frame

The Frame entry for this example contains the Tab Object, with two tabs, one for the Data Table object, and one for the Graphics object. Also, there is an Image Icon object that is used as the exit button. See Figure 8.

(Figure 8)



Example SCL Code

The INIT section here creates a list that offers the user the ability to toggle between the Bar and the Pie chart on selection of the Graphics object. We are creating it in the INIT section so that it is available to the user once the Frame has been opened.

The MAIN section allows us to trap the exit command issued by the user selecting EXIT to make sure they really wish to exit.

The TERM selection deletes our list on exiting, which is an important part of application resource maintenance. The more lists we have unnecessarily created the more memory our application is taking up.

The Graphics object's labeled section executes once the user selects the Graphics object and it offers the Bar and Pie options and on selection sets the appropriate Graph based on the users selection.

```
length return $8;

INIT:
/*************************************************/
/* Initialize frame                              */
/*************************************************/

 link MAKELIST;

RETURN;


MAIN:
/*************************************************/
/* Process user input                            */
/*************************************************/

 if modified(GOBACK) then
 do; /* make sure user wants to exit */

      link TERM;
      _status_='H';

 end;/* make sure user wants to exit */

RETURN;


TERM:
/*************************************************/
/* Cleanup                                       */
/*************************************************/

 rc=dellist(graph_options);

RETURN;
```

```
MAKELIST:
/*************************************************/
/* Create FRAME lists                            */
/*************************************************/

 graph_options=makelist();
 graph_options=insertc(graph_options,'Bar');
 graph_options=insertc(graph_options,'Pie');

RETURN;


GRAPH:
/*************************************************/
/* Pop up graph options                          */
/*************************************************/

 type=popmenu(graph_options);

 if type=2 then
   call notify('GRAPH','_SET_GRAPH_TYPE_',8);

 if type=1 then
   call notify('GRAPH','_SET_GRAPH_TYPE_',16);

RETURN;


DATA:
/*************************************************/
/* Data options                                  */
/*************************************************/

RETURN;
```

Conclusion

SAS/Frame provides a programming environment within SAS that allows application developers to use a suite of objects developed by SAS Institute. These predefined objects greatly enhance the SAS application developer's ability to deliver excellent applications in a timely manner as these objects have already debugged and tested. Since this environment is developed by SAS Institute, the application developer can easily take advantage of the full power of the SAS System and integrate all the other products SAS Institute offers to make a truly powerful and effective application to meet the needs of any information consumer.

Acknowledgements

Author Contact

Christopher A. Roper
Qualex Consulting Services, Inc.
25 R Way
Hillsville, Va. 24343
Chris.Roper@qlx.com
http://www.qlx.com