

Fundamentals of FRAME: A Tutorial for Building Applications

Gregory S. Barnes Nelson, ASG, Inc. Cary, North Carolina

Abstract

SAS/AF¹ FRAME technology allows us to develop user interfaces that are friendly to use. They are the basis for many SAS applications in which end-users determine what happens with the click-of-a-button. This paper presents a framework for building intuitive user-based applications. Here we will develop a simple application, which builds on itself iteratively. By exploring some of the most common FRAME objects and understanding how events are handled in SAS/AF, we will build a simple interface for an applications.

Through this step-by-step process, we will show you how to build an interface that collects user input, which can be used to front-end traditional SAS macros. Despite its simplicity, this paper should give the reader a methodology for developing user-based applications that can serve to front-end most run-time, interactive macros. This paper is intended for both new users of SAS Software as well as experienced SAS programmers with little or no experience in SAS/AF.

Introduction

The scope of this paper is to introduce new and experienced SAS programmers and developers to the world of SAS/AF FRAME development. Our goal here is to introduce the concept of interactive applications. By choosing to start our discussion of applications development from an existing SAS application, our hope is to focus on the tools and techniques to extend this application and build on an existing skill about SAS.

A common challenge in SAS programming is to take an existing program and turn it into something that anyone can run (even your boss Dogbert!) In this paper, we take a traditional programming problem – making a flatfile from a SAS dataset – and write a simple front-end that a user can point-and-click their way through. In the past, there have been numerous articles published on converting SAS datasets to flatfiles (for example, Whitlock, 1995; Buchecker, 1996). For this paper, we will simplify the options we will give our interactive user and build a simple interface that consists of a few parameters. However, in order to truly learn about SAS/AF FRAME, we would highly recommend you take it upon yourself to build an interface to a more complicated macro such as those presented by Ian Whitlock and Michelle Buchener. Feel free to contact the author as you develop your FRAME application (refer to the contact information presented at the end of this paper.)

If you are approaching this paper without a good grounding in SAS or SAS macros, you are encouraged to read on and extrapolate from your knowledge about other programming

languages. Here we will provide enough information about what we are doing without, hopefully, belaboring the points.

Throughout this paper, we will be providing pointers to other references. In this paper, the goal is not to attack the entire subject of SAS/AF development, but to get you started with some very practical tools in your arsenal.

Basics of FRAME

SAS/AF is a module within the SAS System that provides a framework for developing computer software applications that are easier to use and that help the user focus on the business content rather than forcing the focus to be cryptic command line strings and arcane function key combinations. SAS/AF was introduced in Version 5 and looked and felt very much like a mainframe “green” screen. The menus that you could create were called PROGRAM entries and provided the user an interface into SAS applications such as an interactive reporting application.

In 1992, SAS Institute announced a new technology in Version 6.08 of the SAS System called FRAME technology, which could be used to develop graphical user interfaces (GUI). FRAME allows you to develop the interfaces that users interact with to launch programs, view reports, select criteria for subsetting data, etc. Basically, FRAME allows us to provide intuitive, user-friendly front-ends to our SAS applications. Figure 1 shows a few examples of FRAME applications.

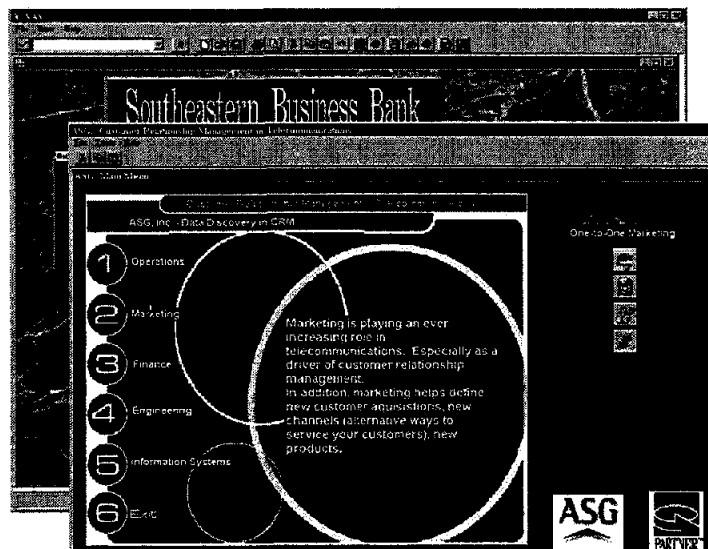


Figure 1. Example SAS/AF FRAME Applications.

The AF Development Environment

Throughout this paper, we will refer to the development environment as the build-time environment. This is where you, as the developer, do your work. We will also reference the run-time environment. This is what the user sees and interacts with.

¹ SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or Trademarks of their respective companies.

Starting the Build Environment

Once you have started SAS in Display Manager System mode (DMS), you can invoke the AF development environment in a variety of ways. A menu driven approach is to select **Globals** → **Develop** → **Application builder**. If you prefer the command line, you can simply type **BUILD** on the command line or command box. This will bring you into the AF build-time environment. You are presented with three windows: Libraries, Catalogs and Entries.

Once you are in the build environment, you can create a new catalog or select an existing catalog. Catalogs are where we will store our application components. The most common types of entries that are used in applications development are listed in the table below.

Entry Type	Description
SOURCE	Entries that contain text information. Usually, non-SCL SAS code like macros, DATA Step or procedural programs are stored here.
MACRO	Macro entries are compiled entries that are not viewable by users. These are binary representations of compiled macros. Their counterparts, SOURCE entries, usually contain the source code.
DATAFORM	Objects that require the Data Set Model store the information in DATAFORM entries. Visual objects such as the Data Table and Data Form use these.
SCL	Much like SOURCE entries, SCL entries store source code. SCL entries are used for storing SCL programming code.
FRAME	FRAME entries are visual containers for interactive applications. This is the subject of this paper.
SLIST	SLIST entries are stored versions of SCL Lists. SCL Lists are used to store information (much like arrays) and can be stored permanently in SLIST entries.
CBT	CBT entries are used to store Computer Based Training information. These are used for help systems and interactive computer based training modules.
TOOLBOX	These are not something that the user will see, but TOOLBOX entries store the information about the toolbars used in your applications. When you launch the SAS system, you are presented with a default toolbar.
IMAGE	Images are stored images. Much like their SAS/GRAPH counterparts (GRSEG), we can use these in our applications to enhance the look and feel of the user interface.
GRSEG	When output from SAS/GRAPH is produced and saved, typically the are saved as GRSEG entries in a SAS catalog. These can be used in the SAS/GRAPH Output object.
FORMAT	Formats in SAS are used to control how data is viewed or printed. FORMAT entries are stored versions of these that can be used throughout the application.
FORMATC	Much like FORMAT entries, FORMATC entries are stored format catalogs, but these store information for character formats.

To create a new catalog so that we can store our catalog entries permanently, we select an existing SAS library from the list. In this case, we have selected the FRAMECLS library (to create a library, refer to the LIBNAME statement). The following figures show the sequence of steps to create a new catalog.

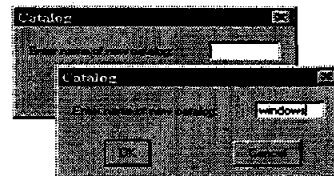
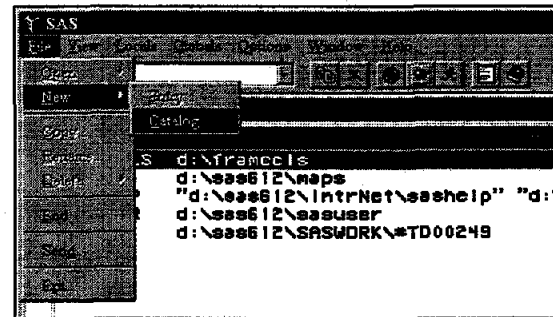


Figure 2. Creating a catalog interactively.

Once we have created a catalog to store our FRAME entry, we can now create the entry itself. To create a new FRAME entry, first select the catalog you have just created and select **File** → **New** → **Entry** from the pull-down menus at the top of the application workspace. Here you are given a choice of the type of entry you want to create. FRAME is selected by default. We simply need to provide a name for the entry we want to create. In the text entry box, type in **EXPORT** to name our FRAME entry.

Once we click **OK**, we are now presented with an empty FRAME object. We can now begin to start building our interactive application. This screen is our blank canvas that we can start adding elements to for our application. To add a visual element or object to our FRAME entry, select **Actions** → **MAKE** from the Pull-down menus (or right-click with your mouse on the FRAME and select MAKE). You will be presented with a list of objects to choose from.

The list of objects shown in Figure 3 is what we will use to focus on in this paper. There are an infinite number of additional objects you can create and add to this list, but these are the ones that SAS Institute provides "out-of-the-box".

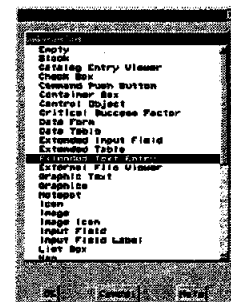


Figure 3. The objects available from the MAKE menu option.

Objects

Several times we have referred to things as *objects*. The term object comes from object-oriented programming and refers to

the fact that they are *instances* (or new cases) of classes that have been provided for us. An *object* is a software "package" that contains a set of related procedures and data elements. A *class* is a template that defines the methods and variables that are available in a particular type of object. A *method* is a procedure that performs a very specific action within an object.

Events

Whenever a user interacts with a computer screen such as pressing a button or selecting an item from a list of elements, an event is occurring. The user expects something to happen. The control transfers from the user (selecting a button) to the computer (reacting to the button click.) This is referred to as event driven programming. In SAS/AF, we represent visual objects on the screen in FRAME entries. The code that drives these events sit behind the scenes in SCL programs. We will talk more about this later when we create an SCL program to respond to a button click.

Common Objects

The following table represents many of the objects that are available to you using the **Make** command in the Build window.

Functional Role	Control/ Widget	Screen Image
Label and input fields	Text Label	This is a label.
Text and label objects can be used to identify an object or simply to provide information to the user. Depending on how you want the user to interact with your application, you may want them to be able to edit/ enter the values for the field (Input Field, Extended Text Entry, Text Entry or Extended Input Field). Other objects may be used for informational purposes only (e.g., Text Label, Extended Text Entry and Graphics Text.)	Extended Text Entry	Extended Text Entry
	Graphic Text	Graphic Text
	Input Field Label	Input Field Label
	Extended Input Field	Extended Input Field 27.70
	Input Field	Input Field 1998
	Text Entry	Text Entry BEVERAGES

Images and Graphics	Image	
Images are often used to provide backgrounds or visually appealing elements to a FRAME or to display the results from a procedure.	SAS/GRAPH Output	

Multiple choice options	Container with check boxes	Please choose one or more hobbies: ☒ Running ☐ Bicycling ☒ Hiking ☐ Racquet Sport
Multiple choices are often required to guide the user through the process of selecting items from a list or check items in a list to turn their values on or off.	List Boxes	Which 3 items go together? Ford Escort Dodge Intrepid Chrysler LHS Chrysler 300M

Action items	Push Button	
Action items are used to launch an event. Events can range from submitting SAS code (e.g., produce a report) to displaying another FRAME entry.	Icon	
	Command Push Button	
	Hotspot	
	Control Object	
Actions items are used to drive events. Other objects, such as the Extended Text Entry can be made to look like a push button, but these are the most common types of action objects.	Toolbar	

Discrete options	Check boxes	☒ Check Box
Like multiple choice options, these can be used to turn something on or off or select an item from a list. However, you can limit the user's options to one choice as in the examples presented here.	Radio Boxes	Please Select One: ☐ Yes ☐ No ☐ Don't Know ☐ Not Applicable
	List Boxes	Select a Region: South North East West

Interfaces to data	Data Table	
Access to data through these objects are often very helpful as it gives the user a visual tool to help them understand the data. All of these objects can also be used to drive events such as clicking on the row of a table and bringing up a data entry form that allows them to edit the data.	Data Form	
	Graphics	
	Critical Success Factor	

Object Functions

As we have just seen, we have a lot of objects to choose from. Deciding upon which ones are most useful can often be a daunting task. There are slight differences in the functionality of many of the objects that, with experience, will help you decide. For example, if you want a text label, you have at least three options: Text Label, Graphics Text and Extended Text Entry. Extended Text Entry allows you to use system fonts (i.e., fonts outside of the SAS font family), whereas the Text Label object does not.

Lists of Items

Often times, parameters contain lists of things. For example, in our export example, we may want to give the user the option of dropping a list of variables from the dataset before we export the

file to an ASCII file. In this case, we need to be able to provide the user with a list of items to select from. The most common choice for representing list of items is the List Box object.

Data Values

Parameters can contain specific data values that you want to pass onto the SAS code behind the scenes. For example, you may want to specify a proportion of a dataset to sample, the number of observations that are to be output or a character string that is evaluated inside the macro. These can easily be represented visually by any of the input objects such as the Extended Input Field, Input Field, Text Entry or the Extended Text Entry objects.

Boolean Values

Boolean values represent a state of being: either on or off. We use these to set flags or to turn something on or off. For example, a parameter might include a Boolean field to turn the debugger on or off inside a DATA Step program. Check Boxes, and Radio Boxes are often the most useful visual objects for setting these kinds of fields.

References to Files and Datasets

Like all modern programming languages, SAS has a rich set of features that provide interfaces to SAS datasets, catalog entries and host specific files such as text and image files. Continuing with our previous example of the export utility, we may want to provide the user with an interface to reference an existing SAS dataset and then let them choose the filename and location for the raw ASCII file that is to be exported. Although we will not show it here, you could use an Extended Text Entry and a Control object along with some additional SCL to allow the user to select an existing SAS dataset. Combined with some SCL programming, this should provide the user with a powerful, yet simple to use interface for selecting datasets and external files.

Types of Output Produced

As with most programs, we often produce some kind of output after the program has successfully run. Often they fall into one of the following three categories: *Textual*, such as SAS output; *Graphical*, such as a bar chart; and *Data*, such as an output dataset from PROC MEANS. The table below provides a few examples of the type of output that is commonly created and the corresponding FRAME object used in viewing/ interacting with that output.

Type of Output	FRAME Object(s) useful for viewing
SAS procedural output	Catalog Entry Viewer & External File Viewer
SAS/GRAPH output	SAS/GRAPH Output
PROC REPORT output	Report Viewer (Experimental)
Dataset	Data Table & Data Form objects

SCL 101

Up to this point, we have focused on what the user sees. That is the graphical user interface. We have also touched on the

different kinds of parameters that can be passed to traditional SAS programs. Here we will discuss some of the behind-the-scenes programming elements. In SAS/AF, we have a powerful programming language called Screen Control Language or SCL. An AF program that contains a SCL entry typically has at least three distinct phases:

- Initialization
- Main Processing
- Termination

The INIT Section

Typically in the INIT section of an SCL program, we will establish the working environment. The kinds of things that are done here include:

- initialize variables
- import values through macro variables
- display initial messages on the window's message line
- open SAS data sets and external files

The INIT section executes before the application window is displayed to the user.

The MAIN Section

The main section is the phase in your program that runs whenever the user interacts with your screen. Specifically, whenever a user presses a button, pulls down a menu, or the user presses enter. The MAIN section is used to:

- validate field names
- calculate values
- invoke secondary windows
- execute commands issued by users
- retrieve values from SAS data sets or external files

The MAIN section executes if any object on the screen is activated. It always executes after object-labeled sections execute.

The TERM Section

Whenever a FRAME is being terminated, the program will enter the TERM, or termination, section. The TERM section is responsible for:

- updating and closing SAS data sets and external files
- exporting values via macro variables
- submit statements to SAS for execution

The TERM section executes when the application window is closed.

The general form of a SAS/AF SCL entry is:

```
INIT: /* Begin the INIT Section */

        Statements to initialize application

RETURN; /* End the INIT Section */

MAIN: /* Begin the MAIN Section */

        Statement to process user input

RETURN; /* End the MAIN Section */

TERM: /* Begin the TERM Section */

        Statements to term the application

RETURN; /* End the TERM Section */
```

Inside each of these labeled sections, we implement our functionality with SCL functions, Data Step code, remote submit commands that utilize client/ server functionality, etc. DATA Step statements such as IF-THEN-ELSE, DO, END, GO and SELECT are supported in SCL. Most SAS functions except LAGn and DIFn are supported in SCL.

The complete list of statements, functions and CALL routines that can be used in SCL programs is detailed in Chapter 7 *Language Elements of the SAS SCL Reference, Version 6*.

Communicating with Objects

Perhaps one of the most powerful features of AF and SCL is the ability to communicate with the objects (e.g., list boxes, check boxes, images, etc.) We often refer to this communication link as "communicating with objects". In general, the way that it works is that you have some screen element that your user interacts with (such as a push button). Once pushed, the AF elements take over. The control is passed from the user to the application environment. The application environment, in this case, is SCL. Some objects such as Command Buttons have inherited properties that execute commands without the use of additional SCL code. But in general, SCL would take over. The program control would shift to the labeled section of the object that the user interacted with. The labeled section containing the appropriate code would run. Then the MAIN section would run and finally return control of the application back to the user.

For example, let's say that there are two objects on our screen: a Push Button and an Extended Text Entry. The attributes for the objects are given below:

Object	Attribute	Value
General Attributes for FRAME entry	Title	My FRAME Application
	Window Appearance: BANNER	NONE
Push Button	Name	BUTTON
	Label	Change Color
Extended Text Entry	Name	ETE
	Initial Value	Welcome to my first FRAME application!

We can easily build the visual objects on the FRAME by using the table above as a guide. For example, we have two objects and our FRAME entry (which is an object in and of itself): a push button and an Extended Text Entry. To create these objects, we can right-click on our empty FRAME and select **MAKE** from the pop up menu (or choose **Actions** → **Make** from the pull-down menus). We are presented with the list of objects.

Select **Push Button** from the list and click **OK**. Once you click **OK** and the object selection list goes away, you can now place the object anywhere on the FRAME. Once you click your mouse button, the object will anchor itself to the FRAME and you will then be brought to the Attributes window for the Push Button. Use the table above the set any attributes such as **NAME** and **LABEL**. You should repeat this process for any additional objects you would like placed on the FRAME entry.

Once you have completed building the objects described above, you are ready to test your application. To test, we will use the **TESTAF** command, which can be found under the Locals pull-down menu.

When we run this application (using the **TESTAF** command under the **LOCALS** menu), the screen should look like Figure 3 below.

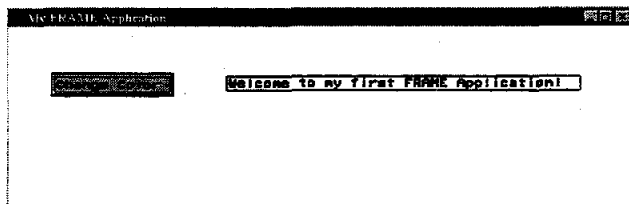


Figure 4. End user view of our first FRAME application

Notice that the title bar at the top has the information that we typed in for the general attributes. This program doesn't actually do anything but is shown here so that we can build on the concepts presented here.

To show an example of what we mean by communicating with objects, let's extend our example to do two additional things: change the color of the text in the extended text entry once we press the push button and secondly, to welcome the user by name once they enter your application. To do this, we will need an additional object on the screen to capture the name of the application user.

First of all, let's define our logic for this application.

1. The user enters the application and, by default, the Extended Text Entry says: "Welcome to my First FRAME Application!"
2. If the user presses the "Change Color" button, the text in the Extended Text Entry changes to Red.
3. If, at any time, the user types in their name in the Input Field (and presses either the button or the RETURN key), the text in the Extended Text Entry changes to add their name: 'Welcome to [NAME]'s FRAME Application! If the NAME field is empty, the color stays red. If the name field contains at least one non-blank character, it changes to blue.

To create the object on the screen, we go back into the **EXPORT FRAME** entry in the **BUILD** mode, right click on the **FRAME** entry (anywhere there is not an object) and select **MAKE**. This brings up the selection list for the out-of-the-box objects that are available to us. Select **INPUT FIELD** and press **OK**. Set the attributes for this object as follows:

Object	Attribute	Value
Input Field	NAME	Name

Leave all of the other attributes as they are by default.

Next, we are going to add the following **SCL** and associate this **SCL** with our **FRAME** entry. To make this association, choose the **LOCALS** menu from the **BUILD** environment and select **EDIT SCL SOURCE**. This will bring up the **SCL** source window. The following **SCL** should be entered here.

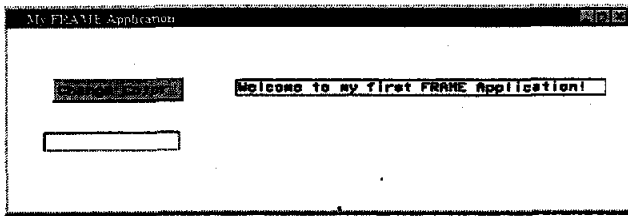
A	INIT: /* Begin the INIT Section */ RETURN; /* End the INIT Section */
C	BUTTON: CALL NOTIFY('ETE','_SET_COLOR_', 'RED'); /* make's the color red (error) */ RETURN;
D	MAIN: /* Being the MAIN Section */ if name = '' then do; cursor name; /* positions the cursor on the NAME object */ Link BUTTON; /* runs code in BUTTON section */ _msg_='Please type in your name'; /* puts a message on the status line */ end; Else do; call notify('ete','_SET_TEXT_', 'Welcome to ' name 's FRAME Application!'); CALL NOTIFY('ETE','_SET_COLOR_', 'BLUE'); end; RETURN; /* End the MAIN Section */
F	TERM: /* Begin the TERM Section */ RETURN; /* End the TERM Section */

Now, let's review the program step by step from the user's perspective. To see what the user will see, you should compile the above code (in the **SCL** window, select **LOCALS** →

COMPILE and review the status line for any errors.) Once compiled you can exit out of this screen by pressing the PF3 function key. Then select LOCALS → TESTAF.

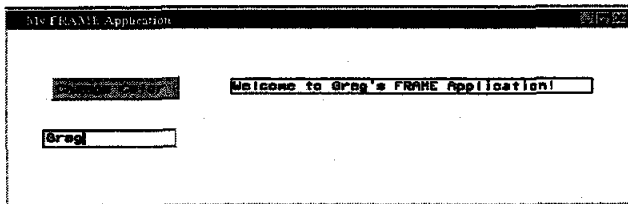
Once you are in TESTAF mode, you should see something similar to the screen below.

B



If you press the "Change Color" button, the text should change to red and the cursor should move to the TEXT INPUT field. If you type in your name and press return, the message in the EXTENDED TEXT ENTRY should change to "Welcome to Greg's FRAME Application" (only if your name was Greg!)

E



Programmatically, what happens is as follows:

Item	Controlling Entity	Description
A	AF	Application starts. INIT section is run and screen is initialized.
B	User	User presses "Change Color"
C	AF	Runs the labeled section call BUTTON. Button section communicates to the extended text entry telling it to set its color to red.
D	AF	- After the BUTTON labeled section is executed, the MAIN section executes. MAIN checks to see if the value of name is set. Here the INPUT FIELD name automatically sets an instance variable called NAME to whatever the value is in the field. If that variable is missing, the program tells the cursor to move to the NAME object and puts a message in the status box at the lower left hand corner of the screen. - If the name variable is anything but missing, the MAIN section executes the ELSE DO which set's the text of the EXTENDED TEXT entry to "Welcome to Greg's FRAME Application" and changes the color of the Extended Text Entry to blue. - Controlled is then returned back to the user after the MAIN section is executed.
E	User	Presses PF3 or issues the END or CANCEL

		command.
F	AF	Executes the TERM section. In this case, nothing is done and the application ends.

A Final Word on Communicating with Objects

One of the most powerful features in SAS/AF and FRAME is that you have a rich set of classes that are available to you from which you can create instances of objects. For example, within the widget class, we have the visual objects we see under the MAKE option when we add objects to our FRAME (e.g., INPUT FIELD). In general, the objects we can see are called visual objects and we communicate to them through methods. That is, each object has a set of methods or interfaces in which we communicate with them. Some of the more common ones are GET and SET methods. These allow us, for example, to get the values from widgets at run-time as well as set them. All of the methods are published in the *SAS/AF Software: FRAME Class Dictionary* as well as contained in the online help files (type **help classes** on the command line to get to the on-line documentation.)

Running Non-SCL SAS Programs

In this simple example, we've seen the use of SCL programming structures (such as IF-THEN-ELSE DO loops), communication with objects (CALL NOTIFY), and special built in SCL statements such as `_msg_ =` and `cursor` statements. To extend this, let's take a look at how to run traditional kinds of SAS programs within SCL.

In your programs you designate the statements you want submitted by SAS by placing the statements in submit blocks. A SUBMIT block has the following characteristics:

- Begins with a SUBMIT statement
- Ends with an ENDSUBMIT statement
- Consists of all statements between the SUBMIT and ENDSUBMIT statements

When arguments can be added to the SUBMIT statement to control when your code is submitted for execution. They are:

CONTINUE – submit immediately

IMMEDIATE – submit immediately then stop

PRIMARY – submit immediately then return to primary window

TERMINATE – submit immediately and then terminate application

Submit blocks are used to process non-SCL code. This is where we would include our BASE SAS code. For example, assume that we have an INPUT FIELD (called DATASET) that contains the name of a SAS dataset that we want to export. A second INPUT FIELD (called OUTFILE) contains the path and filename of the external file we would like to save this information to. A third element might be a push button (called BUTTON) that processes this request and run's the appropriate code.

BEGINNING TUTORIALS

The attributes for this FRAME are as follows:

Object	Attribute	Value
General Attributes for EXPORT FRAME entry	Title	Export Utility
	Window Appearance: BANNER	NONE
Input Field	NAME	dataset
Input Field	Region Title	Enter the name of a SAS dataset:
Input Field	Region Title	Enter the name of an external file:
Push Button	NAME	button

Our code for the BUTTON labeled section would look like this:

BUTTON:

```
SUBMIT CONTINUE;

  proc dbload dbms=xls data=&dataset;

    path='&outfile';

    putnames=yes;

    label;

    load;

  run;

ENDSUBMIT;

RETURN;
```

In this labeled section (BUTTON), we implement a section of code that uses SAS/ACCESS to PC File Formats (PROC DBLOAD). But instead of hard-coding the values for the dataset and the external file we are creating, we have supplied those to the SAS program (in our SUBMIT block) as parameters. Here these look like macro variables (&dataset and &outfile), but are really SCL variables. Whenever, we want to use an SCL variables within a SUBMIT block, we have to represent these with a preceding ampersand (&).

Once you have compiled the SCL code and attempt to test your application using the TESTAF command, you will notice an error message on the status line: **WARNING: Cannot SUBMIT from TESTAF.** In order to run this program you will need to save the FRAME entry (FILE → SAVE) and exit the BUILD FRAME window. On the command line type **af c=libref.catref.FRAME2.FRAME**, where libref is the name of the library this is stored in and catref is the catalog. In this example, we used the following:

```
af c=FRAMECLS.WINDOWS.FRAME2.FRAME
```

The screen for this application is given below.

Once we press the **Submit** button, you can review the log to ensure everything worked as it was supposed to. In the author's case, here a partial log that was produced:

```
13  proc dbload dbms=xls data=sasuser.cheese;
14  path='c:\temp\outfile.xls';
15  putnames=yes;
16  label;
17  load;
18  run;
```

NOTE: Load completed. Examine statistics below.
NOTE: Inserted (9) obs into outfile.xls.

Obviously, you would want to do some more error checking to ensure the user actually got the result he/ she expected. The types of error checking that you should do might include:

- Making sure the SAS dataset was valid
- Making sure the path name and filename were valid
- Prompting the user before overwriting the external file (currently, you will get an error if you try and run this code and the file already exists)
- Popping up a list of valid SAS dataset names
- Providing a window for the user to navigate and to specify the name of the external file

Invoking Your Application

AF vs AFA

Building your application is a development activity. As you have seen, we have tested your application in two ways: running the application in TESTAF mode (Locals → Testaf) and by using the af command in the command window. As we mentioned earlier, you can run an AF program in TESTAF mode if you have a SUBMIT block in your SCL code. Hence the reason to use the af command from the command line.

Occasionally you may see reference to the afa command. The afa command is similar to the af command except that it allows multiple instances of AF applications. It basically allows the application to be run in a separate memory space from other applications.

Configuration Files

There are a variety of ways you can start your AF application so that your user doesn't have to launch the application themselves. The SAS System provides a system option, INITCMD, that lets you invoke SAS/AF programs automatically. You can specify an `initcmd` at startup in either the icon properties or in a modified `config.sas` file. For example, you can add this information to your `config.sas` file to automatically start up the graph-n-go application in SAS:

```
-initcmd "AF c=sashelp.graph.gobject.frame; "
```

For more information about the INITCMD system option, see SAS Software: Changes and Enhancements, Release 6.10.

To do the same thing without touching your `config.sas`, put the following information into the target properties of a desktop icon (Windows 95, NT):

```
D:\sas612\sas.exe -initcmd "af
c=sashelp.graph.gobject.frame" -awstitle 'SAS
Graph-N-Go Application'
```

Where `d:\sas612` is where you have SAS installed on your system.

Advanced Topics (or Where to go from here)

There are a wide variety of techniques that allow you to do much more advanced kinds of things than we are showing here, but this should be helpful in getting familiar with what FRAME entries are and how you can use them. Additional kinds of topics that you will want to explore can take you very deep into the topic of SAS/AF including SCL programming, object-oriented techniques, drag-and-drop, screen sizing (for large and small resolution monitors), etc. The table below should give you some general references that are helpful in your exploration of SAS/AF and FRAME entries.

Topic	Resource
Introduction to FRAME	SAS Institute's Web Site http://www.sas.com/service/edu/courses/prod_af.html
SCL	"Beyond the Obvious with SAS(R) Screen Control Language" by Don Stanley
Getting started with FRAME	"Getting Started with the FRAME Entry: Developing Object-Oriented Applications", Second Edition
SCL Reference	"SAS(R) Screen Control Language: Reference, Version 6", Second Edition
Advanced FRAME and SCL concepts	"SAS/AF(R) Software: FRAME Application Development Concepts", Version 6, First Edition Advanced FRAME (SAS Institute course offering)
SAS/AF Classes	"SAS/AF(R) Software: FRAME Class Dictionary", Version 6, First Edition (NOTE: the online class dictionary contains the most up-to-date information)
CD-ROM with examples	Solutions@Work(TM): Rapid Applications Development Examples
Working with Regions	"SAS/AF® Software: FRAME Entry Usage and Reference Version 6", First Edition chapter 3
Using the SCL	There is a good section on the SCL debugger in

debugger	chapters 33 and 34 of "SAS Screen Control Language, Usage", First Edition
SCL Lists	Michael Davis's paper "SCL for the Rest of Us: Nonvisual Uses of Screen Control Language" has an excellent overview of SCL. Scott Groussmand and Robin Yeldell's 1996 SESUG Paper: "Flexibility of SCL Lists" "Harnessing the Power of SCL Lists" by Lisa Ann Horwitz
Object-Oriented Programming	The best resource is to take the Object-Oriented Programming class from SAS Institute. They also have a new course on Advanced FRAME. Several of the publications reference the information as well as several SUGI articles over the past few years.
Screen Sizing (using attachments)	SAS Observations article (Whitmeyer) "Using Region Attachments Effectively in a FRAME application as well as "SAS/AF® FRAME Application Development Concepts" in chapter 3.
Pull-down Menus	"Introduction to FRAME Course Notes" (SAS Institute) "Using Design Patterns to Implement Object-Oriented Menus in a SAS/AF Application" by Jack Fuller
Using EIS with FRAME	SAS Institute has a one-day course on HOLAP, where they discuss using EIS within FRAME application. Chapter 12 of the "Introduction to Rapid GUI Application Development Using the SAS System Course Notes".
Cursor Tracking	"Cursor Tracking in SAS/AF FRAME Applications" by Don Stanley
Drag and Drop	"SAS/AF® FRAME Application Development Concepts" in Appendix 3.
Packaging Applications for Run-Time	Although SAS doesn't have a runtime version of it's software, there are some tricks and techniques that don't appear to be discussed anywhere very well. There is a brief section in the appendix to the "Introduction to FRAME course notes" from SAS Institute.
Writing help for your applications	"Native Help Technology in SAS/AF Applications" By J. Kruth
Using OLE controls within SAS AF applications	"OLE and the SAS System for Windows, Release 6.12" by Jennifer Clegg and Carol Riggsbee
Cross-platform development	"Cross Platform Applications Development" by Greg Barnes Nelson "Ensuring Portability o SAS/AF Applications" by Schlotzhauer, S.D. & Thompson, E. T. "This is a Fairy Tale... NOT! A Primer on Moving SAS Applications Across Graphical Operating Systems" by J. Hefner

Conclusion

SAS/AF and FRAME are powerful tools that can be used in conjunction with traditional SAS programs and macros to help guide end-users through a series of choices and/or selections. By talking about developing interactive applications in the context of a traditional programming problem – making a flatfile from a SAS dataset – we have applied some familiar concepts with a new technology. Hopefully this has allowed you as the reader a greater sense of what SAS/AF and FRAME is all about. By showing some simple examples of communicating with objects through simple methods like GET and SET, you can explore other objects and how you can use them in your applications.

Acknowledgements

I would like to acknowledge the assistance of Michael Davis and Ian Whitlock for their insightful comments on earlier versions of this manuscript. Also to, Gavin Erickson for sharing his paper on SCL with me.

Bibliography

- Barnes Nelson, G.S. (1998) *Cross-Platform Application Development*. Published in the Proceedings of the Annual Conference of the South Eastern SAS Users Group – Jacksonville, FL.
- Buchecker, M. (1996) *%FLATFILE, and Make your Life Easier*. Paper presented at the annual convention of the International SAS User's Group (SUGI), Chicago, IL.
- Clegg, J. & Riggsbee, C. (1996). *OLE and the SAS System for Windows, Release 6.12*. Published in the proceedings of the 22th Annual Conference of the International SAS Users Group - San Diego, CA. Pp. 257-266.
- Davis, M. (1998) *SCL for the Rest of Us: Nonvisual Uses of Screen Control Language*. Published in the proceedings of the 23rd Annual Conference of the International SAS Users Group – Nashville, TN.
- Fuller, J. (1998) *Using Design Patterns to Implement Object-Oriented Menus in a SAS/AF Application*. Published in the proceedings of the 23rd Annual Conference of the International SAS Users Group – Nashville, TN.
- Groussman, S. & Yeldell, R. (1996) *Flexibility of SCL Lists*. Published in the Proceedings of the Annual Conference of the South Eastern SAS Users Group – Atlanta, GA.
- Hefner, J. (1994) *This is a Fairy Tale... NOT! A Primer on Moving SAS Applications Across Graphical Operating Systems*. Published in the proceedings of the 19th Annual Conference of the International SAS Users Group - Dallas, TX. Pp. 404-409.
- Horowitz, L.A. (1998). *Harnessing the Power of SCL Lists*. Published in the proceedings of the 23rd Annual Conference of the International SAS Users Group – Nashville, TN.
- Kruth, J. (1997). *Native Help Technology in SAS/AF Applications*. Published in the proceedings of the 22th Annual Conference of the International SAS Users Group - San Diego, CA. Pp. 1520-1523.
- SAS Institute Inc. (1990, 1st Qtr.), *Meeting the Challenge: A New Generation of Software for the 90's*. SAS Communications, Vol. XV, No. 3. pp. 3-8.
- SAS Institute Inc (1995), *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition*, Cary, NC: SAS Institute Inc.
- SAS Institute, Inc. (1993), *SAS/AF Software: FRAME Entry, Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc. (1997), *SAS Macro Language, Reference, First Edition*, Cary, NC: SAS Institute Inc.
- SAS Institute Inc (1994) *SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.
- Schlotzhauer, S.D. & Thompson, E. T. (1993) *Ensuring Portability of SAS/AF Applications*. *Observations*. Vol. 2, No. 2. 1st Qtr.
- Stanley, Don (1994), *Beyond the Obvious with SAS Screen Control Language*, Cary, NC: SAS Institute Inc.
- Stanley, Don (1997), *Cursor Tracking in SAS/AF FRAME Applications*, Proceedings of the Twenty-Second Annual SAS Users Group International Conference, 148-153.
- Whitlock, I. (1995) *How to Write a Macro to Make External Flat Files*. Paper presented at the annual convention of the International SAS User's Group (SUGI), Dallas, TX.
- Whitmeyer, B. (1995) *SAS Observations. Using Region Attachments Effectively in a FRAME Application*, 4th Qtr., 1995 Cary, NC: SAS Institute Inc.

Author's Address

Questions/ comments pertaining to this article should be directed to:

Gregory S. Barnes Nelson
ASG, Inc.
2000 Regency Parkway, Suite 355
Cary, NC 27511
919.467.0505 (Voice)
919.467.1195 (Fax)
gbn@asg-inc.com - Employer

gregbn@ix.netcom.com - Home