Paper 27-28

Developing SAS/AF® Applications Made Easy

Bernd E. Imken, Patented Medicine Prices Review Board, Ottawa, Canada

ABSTRACT - WHAT IS SAS/AF®

Using SAS/AF software, developers can create customized, point and click applications. All the power of SAS for data access, management, analysis and presentation is available to your organization's end users. They can access current information quickly and easily by using a mouse or function keys, without having to write SAS code or knowing how to use the SAS Windowing environment.

SAS/AF gives the developer an object oriented applications development environment. User interfaces can easily be created with graphics, icons, pull-down menus, push buttons and many other objects (called Visual Components starting in SAS V8, including radio buttons, slider bars etc.). These components can also be easily customized to suit differing needs and requirements.

SAS/AF can also be used in conjunction with other SAS products such as SAS/EIS.

REQUIREMENTS TO BUILD AN APPLICATION

In order to develop applications you must have a license for SAS/AF software and SAS/AF must be licensed on your computer. This is only a requirement if you are going to develop a new application or modify an existing one. The Base SAS license is of course also required.

REQUIREMENTS TO RUN AN APPLICATION=

Users can run SAS/AF applications on their computers without a SAS/AF license on their machines. They do however need a license for Base SAS as well as any other packages such as SAS/Graph which may have been used in developing the application.

AN EXAMPLE

Figure #1 below, shows a screen developed to enter customer address information. A user can easily navigate through the data base in a number of ways.

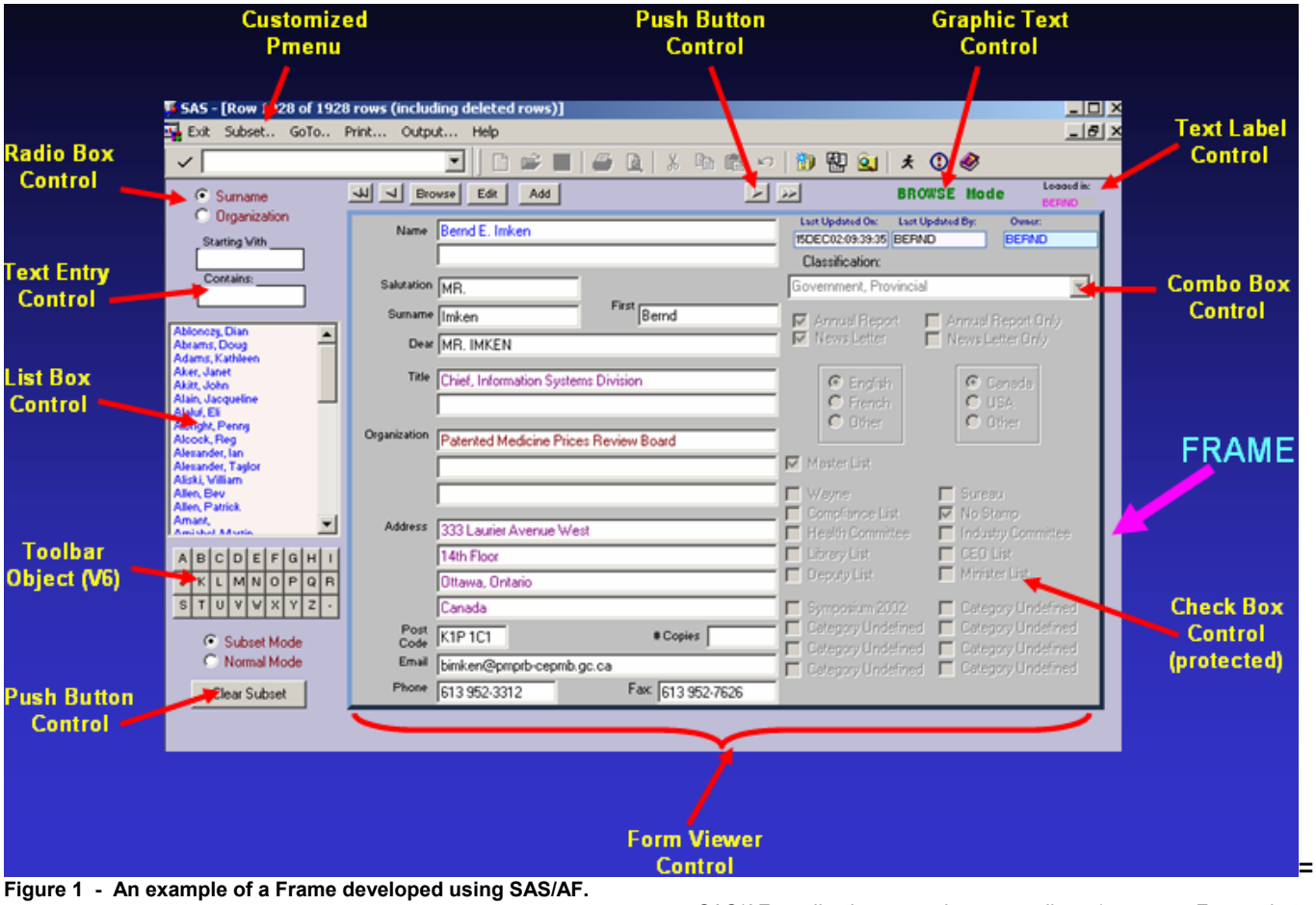


Figure 1 - An example of a Frame developed using SAS/AF.

When the user selects the name of a person or an organization from the List Box Control, the Form Viewer Control displays the associated demographic data. The user can select a letter from the Toolbar Object to subset what is displayed in the List Box Control. The Radio Box Control determines whether the List Box Control displays names or organizations. Push Button Controls above the Form Viewer Control allow the users to navigate easily with the data base.

SAS/AF applications may be as small as 1 page or Frame, but frequently they contain many Frames. SAS/AF provides you with the tools to build the screens as well as the tools to navigate between screens.

This paper will show you how to develop such applications easily. Using a step by step methodology you will first develop a very simple 1 screen application. This will then further be enhanced by adding other objects and additional screens until you have mastered most of the facilities required to develop elaborate SAS/AF applications.

A. CREATING A SIMPLE SAS/AF APPLICATION

Figure 2 below shows a simple Frame which has been populated with a number of SAS Version 8 Control Components. When the user types in the employee's original salary in the box provided, the system automatically calculates a new salary which is 10% higher. When the "Bonus" push button is pressed, an additional $500. will be added to the new salary.

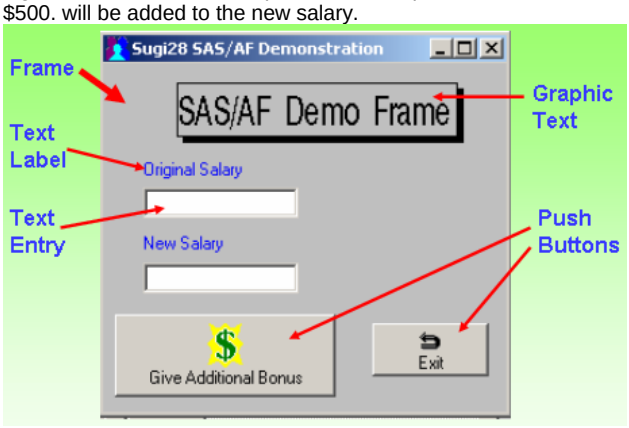


Figure 2 - Developing a Simple Frame

The instructions given below will show you, step by step, how to develop a blank Frame and then populate it with the List Box and other controls. You will then be shown how to change component "Properties" to make the controls look like they are in Figure 2. Finally you will be shown how to add SCL code to your application to actually produce the print out for you.

A1. CREATE A NEW SAS/AF CATALOG

You can create an AF catalog within any SAS library. In this demonstration, we will continue by using the already existing WORK library.

Figure 3 shows you 4 partial SAS screens giving you all the steps required to create the new catalog. The table below Figure 3 takes you though the process, step by step. Please follow the steps exactly as described. (Note: within SAS there are frequently alternate methods of doing the same thing. Within this paper I attempt to show you the easiest approach. Alternate methods may also be discussed if they are appropriate.)

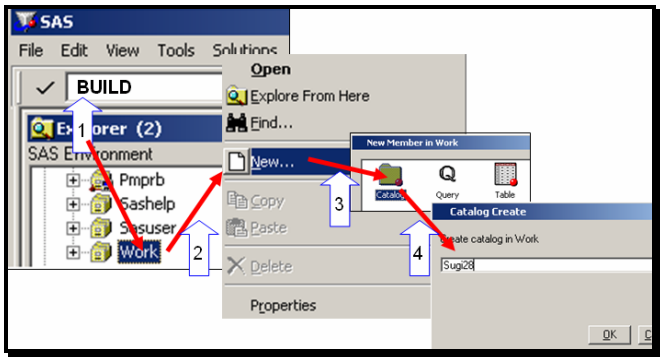


Figure 3 - Creating a SAS/AF Catalog

#	What to Do	What Happens
1	Type **BUILD** on the command line	The Explorer windows opens up as shown in figure # 3 displaying all the libraries which are presently assigned
2	Click your RIGHT MOUSE Button on the library in which you would like to create the catalog (eg.WORK)	Another window opens up with "Open" at the top and "Properties at the bottom.
3	Select **New**	Another window opens up with the title "New Member in Work"
4	Click on **Catalog Icon**	Another windows entitled "Catalog Create" opens.
5	Type catalog name (eg. Sugi28) and press OK	Catalog Create screen closes and you are returned to SAS Explorer

Alternate Approach : It is also possible to create a catalog directly by typing in the following command within the SAS Command line/box.

BUILD work.sugi28

Or, you could have submitted the following SAS code from the Program Editor.

```
Proc build c=work.sugi28;run;
```

A2. CREATE A NEW FRAME

Now that we have created a new catalog, let's next create our first screen, called a FRAME within this catalog. Please follow the steps as shown in Figure 4.

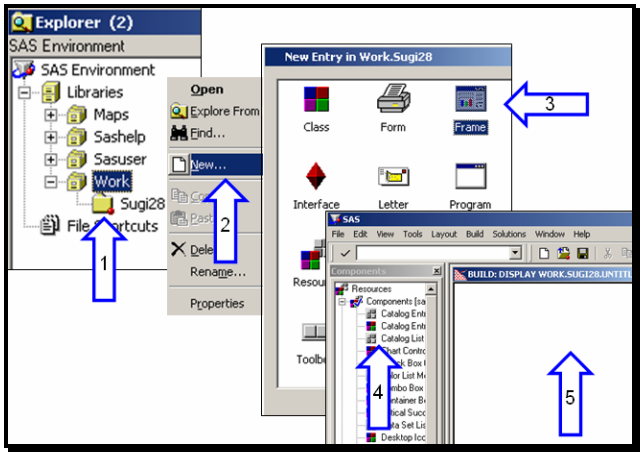


Figure 4 - Creating a Frame entry

#	What to Do	What Happens
1	Right Click the newly created catalog Sugi28	A selection window is displayed with Open at the top and Properties at the bottom.
2	Select **NEW**	A new window is displayed entitled "New Entry in Work.Sugi28"
3	Select the **Frame icon**	2 new windows open up. To the left is the components window and to the right is a window entitled **build:display**

		work.sugi28.untitled1.frame
4	Move your cursor over the newly displayed Components Window	The components window displays all the SAS Version 8 components and Version 6 Object which may be used on the Frame
5	Click on the Build Window	This is your first blank Frame.

You have now just created your first FRAME, Let's now make some initial modifications to this Frame.

Alternate Approach: It is also possible to create a catalog directly and a new Frame at the same time by typing in the following command within the SAS Command line/box.

BUILD work.sugi28.demo.frame

Or, you could have submitted the following SAS code from the Program Editor.

Proc build c=work.sugi28.demo.frame;run;

A3. MODIFY APPEARANCE OF THE BLANK FRAME

Now that we have created our first frame let me show you how to

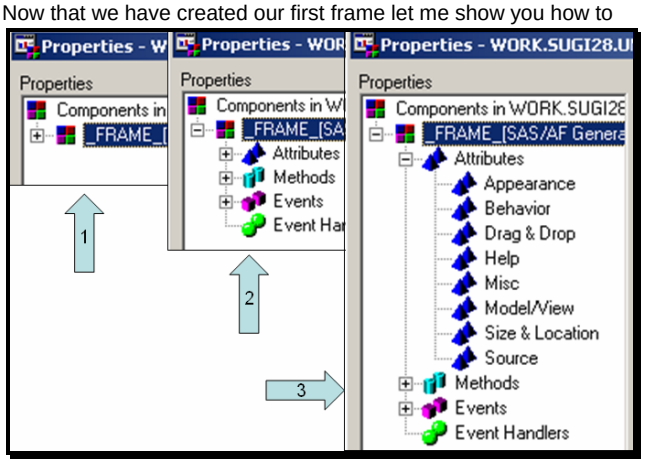


Figure 5 Drill Down on Properties Demonstration

#	What to Do	What Happens
	Right Click anywhere on the blank Frame	Window opens up with "Open" at the top and "Properties" at te bottom
	Click on "Properties"	The properties window opens up as shown to the left of Figure #5.
1	Click on _FRAME_	Attribute window on the right displays All the attributes associated with the Frame
2	Click on + sign to the left of _FRAME_	Provides a drill down now showing Attributes, Methods etc.
3	Click on Attributes	Drill down continues, this time displaying Appearance, Behavior etc.
4	Click on Appearance	Notice now that the attribute screen on the right only displays the attributes associated with the Frame's appearance.
5	If you want to change the prompt at the top of the screen modify the attribute bannertype. When you chick on the word "Command" a new push button is displayed beside Command. bannerType Command Change the BannerType to "Select"	When the button is pressed you can choose Command, Select or None.
6	Next close the Properties window by pressing the close botton	You are returned to the Frame window.
7		

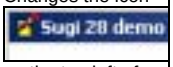
Select Build on the Menu at the top of your screen. Press Test to bring up your new frame.

Notice that the screen now has the Select banner.

The title at the top of the Frame can be changed by changing the Title Attribute.

In the next table, the most frequently modified frame attributes discussed. Using these you can make quick modifications. Although **there are many other attributes available**, not all work within Windows (eg. Blinking banners). Other attributes, methods etc will be discussed later in this paper.

Frequenty Accessed _Frame_ Component Attributes

Type	Attribute	Selection	Explanation
Appearance	**backgroundColor**	Black, Blue, Brown, Cyan, Gray etc.	Most colors can be selected.
	bannerType	Command, Select, None	Most commonly "None" is selected
	title	Any title you desire	Your choice
Behavior	**icon**	Select icon number	Changes the icon  on the top left of the Frame
	keysEntry	Specify location of keys entry in the SASUSER PROFILE catalog	Modify your Function keys by typing "keys" in the command box while in Build mode. Then select "Save AS" from under the File menu
	pmenuEntry (very useful)	Specify location of pmenu in the catalog	New Pmenus can be developed using Proc Pmenu see SAS Help for an explanation
	type	Standard or Dialog	Most frames are Standard
Source	**SCLEntry**	Specify location of the Frame's SCL code	SCL will be discussed in the next section.

A4. SAVE THE FRAME

If you want to save the Frame which you have just developed, click on Save As under the File Menu at the top of the screen. On subsequent Saves it is sufficient just to exit the frame

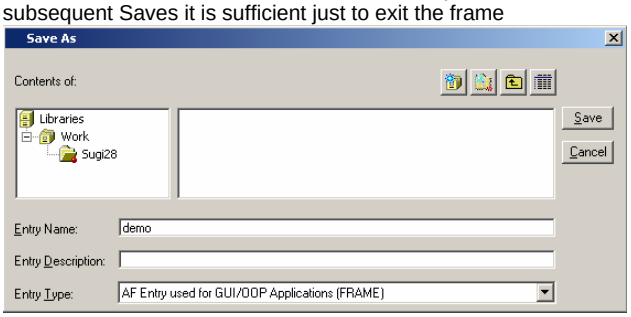


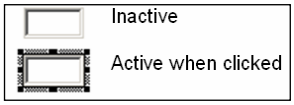
Figure 6 - Saving the Frame Entry

A5. ADD COMPONENTS (Objects) TO THE FRAME

Next, let's start adding some components/objects to the blank

frame which we have created. To do this click on the components which you desire on the left and drag it over the frame and drop it in the desired location. Or, you can double click the component and the component will be opened at the top left corner of the Frame.

An example of creating a Text Entry Control Component is show in Figure 7. Here we dragged the Text Entry Control from the Component window and dropped it on the Frame. You will also notice that when you click on the newly created component it becomes active and can be recognized as active by it's different appearance. You will also see that if you move your



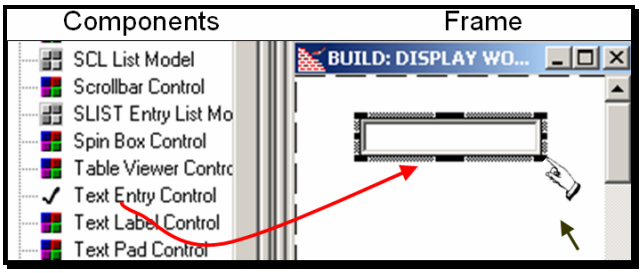


Figure 6 - Creating a Text Entry Control on the Frame

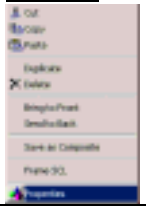
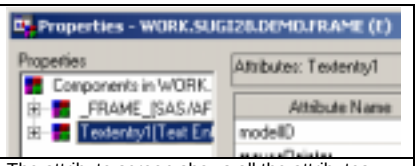
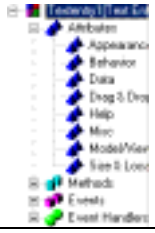
cursor over the active border it turns into a hand - anytime that the hand appears, you can reposition the control component to another space on the frame. If the cursor is within the component or is positioned outside the component, it has it's regular shape. The components can also be re-sized by dragging the corners of midpoints of the line around the component.

Now try creating the 2 text entry components and 2 text label components which were originally shown in Figure 2. Drag 2 of each onto the Frame. When they are first created they should look like the illustration on the left.

After this has been completed you would like to change the caption "Label" to "Original Salary" and the second caption to "New Salary". This is done by modifying the component attitudes as is described in the section below.

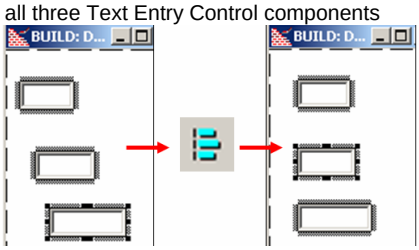
You may also at this stage create the Graphic Text and Push Button Components. The attributes for these components will also have to be modified to get to the appearance as shown in Figure 2 - however I will concentrate first in the Labels and Text Entry components.

A6. MODIFY COMPONENTS

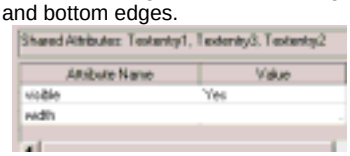
#	What to Do	What Happens
1	If you want to modify the component, right click anywhere on the Text Entry , the graphic text, label, or push button control	A new selection window is displayed with **Properties** at the bottom. 
2	Click on **Properties**.	A new Properties window is displayed showing the _Frame_ and all added components on the left.
		
3	Click on "+" sign to the left of the component , eg. "Text Entry Control". Drill down just as you did with the Frame. 	The attribute screen shows all the attributes pertaining to a check box control. The properties window now displays different selections from Appearance to Size…etc. When you click on Appearance only the Attributes related to Appearance are displayed.

Many attributes are common to most of the Control Components. For example the NAME Attribute is used on all Components and allows the user to substitute a more meaningful name to replace the one automatically assigned by SAS. If you create your first Text Entry Control it will be called Textentry1 by SAS, if you create a second one it is called Textentry2 etc. These could be renamed to OldSalary and NewSalary. (If you right click on the component it can be deleted, however SAS will not modify the names originally assigned.)
(Hint: the system always points to the name attribute first when the properties window is opened.)

It is also possible to align components. For example in the diagram above make all three Text Entry Control components active at once by pressing down on the UpShift key at the same time as you click each component sequentially. Once all three are active, click on the align left icon at



the top of the Frame screen (on the menu). Notice all three components are properly re-aligned. At the top of the screen you will also see alignment icons for aligning right edges, top edges and bottom edges.



Activating multiple controls can also be used when you want to modify certain attributes such as sizing. For example if you wanted all the Text Entry Controls in the previous example to have the same width, activate all three and then right click and select Properties from the select list. The attributes list now displays the shared attributes for visible and width. Changing the width attribute gives you the desired result.

A7. ADD SCL TO THE FRAME

Now that we know the basics about setting up a frame and adding elementary components such as a Text Entry Control, let's look at how we can use this to gradually build up simple applications. Using the example in Figure #2 a portion of

which is displayed to the left, we want the user to type in a value in the old salary box and then I want the system to automatically increase the salary shown by 10% and display it in the New Salary box. I have also renamed TextEntry1 to OldSal and TextEntry2 to NewSal.

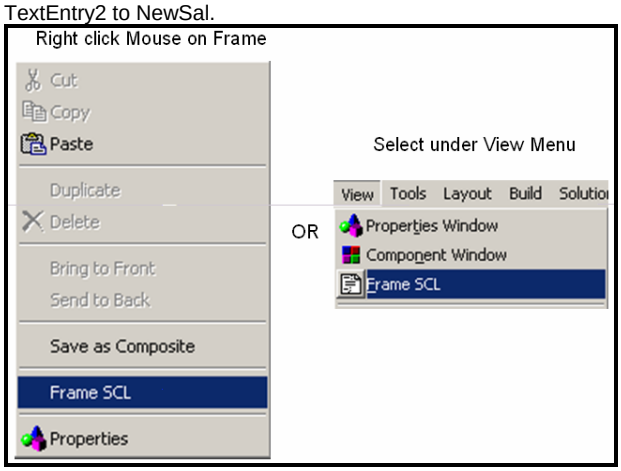


Figure 7 - Alternate Ways to add SCL to a Frame

SAS Component Language can be added to every Frame. By using SCL you can programmatically control how the frame appears or behaves. You can add SCL to a frame by right clicking on the Frame or by selecting Frame SCL under the View menu at the top of the screen. In order that we can do the calculation of the new salary, the SCL behind the screen will have to be added as shown to the right. (Color has been added to the SCL code for emphasis only)

```
INIT:
    _Frame_.backgroundColor='Gray';
    _Frame_.title='Sugi28 SAS/AF
                   Demonstartion';
    _Frame.icon=247;
    call libname('demodata', 'c:\sugi28');
Return;

MAIN:
    NewSal.text = OldSal.text * 1.10;
Return;

TERM:
Return;
```

The INIT Section execute only once, when the frame is initially opened. In this example, the INIT uses SCL dot notation to change the background color, title and icon. The syntax being:

```
Object.attribute=value;
```

Most of the attributes can be modified easily in this manner. The INIT section also demonstrates a SCL CALL Routine:

```
call libname('demodata','c:\sugi28\data');
```

Let's now test the application to see if it works. After Compiling the SCL and checking the message line to see that it was successful we can do a Test. Both of these commands are found under "Build" on the menu at the top of the screen On the left you can see that the user typed in the value 100000 and then pressed the ENTER key. The MAIN section in the SCL behind the screen automatically does the calculations each time the ENTER key is depressed.

Next I would like to add a Push Button Control Component to the screen. You can see on the screen at the right that I have also changed the label attribute to "Add Bonus", the icon attribute to "699" the icon value representing the "$" sign and have also changed the buttonStyle attribute to "Icon with Text to Right". Next I will add some additional SCL code to add $500 to the new salary, only when the push button is pressed. (pushButton1.name = "AddBonus")

```
INIT:
Return;

OLDSAL:
   NewSal.text = OldSal.text * 1.10;
Return;

ADDBONUS:
   Link OLDBAL;
   NewSal.text = NewSal.text + 500.;
Return;

MAIN:
Return;
```

You can see in the SCL code at the left that I have added 2 new labeled sections corresponding with the names of the OldSal Text Entry Control and the Push Button Control. The calculation of the 1.10 increase happens every time a change is name to the OldSal Text Entry Object. I have also added a line to the ADDBONUS section to calculate an addition bonus.

Section Heading	Usage
INIT:	Executes when a row in the data set described by the model is displayed. With a Table Viewer, INIT runs for each displayed row when the viewer is populated, when scrolling, or when a row is locked.
column_name :	Labeled sections that correspond to variables in the data set described by the model are executed when the value of the variable is modified via the viewer. This section does not execute if the value is changed programmatically via SCL.
MAIN:	This executes when any data in the data set is modified, prior to the execution of any frame SCL.
TERM:	This executes as the user leaves the current row only when in edit mode.

Why did I remove the 1.10 increase from MAIN? This was done because the MAIN Section always executes after any labeled section. If this had not been done the bonus of 500 would have been wiped out because the MAIN would have recalculated the 1.10 increase last. The order that the sections appear in the SCL code does not impact the processing order.

A8. PRODUCTION RUN

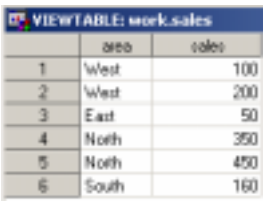
To run an application in production, exit SAS/AF then enter the following command on the command line.

```
af c=sasuser.sugi28.demo.frame
```

It is also possible to run multiple SAS/AF sessions concurrently by using the "afa" command.

B. MORE ELABORATE APPLICATIONS

This section will show you how to develop more sophisticated multi Frame applications. It will also demonstrate how SAS code can be executed within SCL.



In the next example, I would like to show you how to navigate between different frames which you have created. For this example I have created a simple data set called work.sales, which is displayed in the ViewTable at the left.

The first frame which I will develop will ask the user if he wants to see a detailed or summarized sales report; These choices will be given by 2 push button control components. Depending on which button is selected the user will be directed to one of two other frames.

This example is shown in Figure #8. Frame #1 shows the choice or menu Frame. Number 2 shows the Detailed frame and #3 shows the summary Frame. The results output from this Frames is shown on the right.

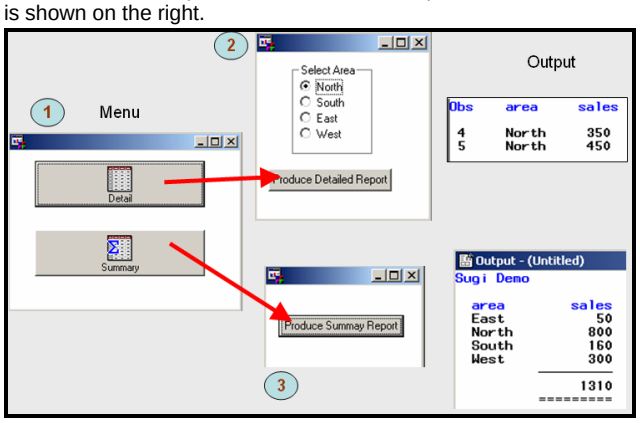
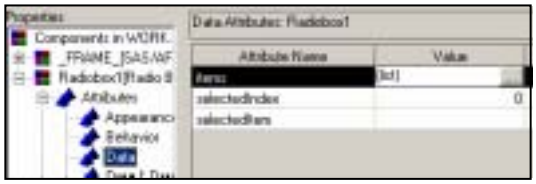


Figure 8 - Aother Example showing 3 Frames

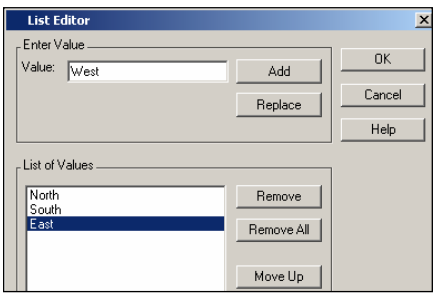
The first frame was very easy to develop and was similar to one's we've already done. For the push buttons I used the large icon attribute with text below. The major difference is in the SCL code which determines what happens when each push button control is pushed. In the SCL code below you will see I have added 2 labeled sections which correspond with the names of the 2 push button controls. The DETAIL section passes control from the current Frame to another Frame entitled SALES_DETAIL using the CALL DISPLAY statement. (If you are going to another catalogue or library the full four part naming must be used.). Similarly the SUMMARY section passes control to the SALES_SUMMARY Frame.

```
INIT:
return;

DETAIL:
    call display('sales_detail.frame');
return;

SUMMARY:
    call display('sales_summary.frame');
return;
```

In Frame #2 Figure 8, the SALES_DETAIL Frame introduces another component - the RADIO BOX CONTROL. We can change the Appearance attributes for the radio box exactly as we did before, however, this control component illustrates how we can use DATA Attributes to identify North, South etc. as selection buttons.



Drilling down to Data will display the data attributes for a radio box. Next, on the Value for the items attribute to display the screen below. The panel is fairly self explanatory allowing you to easily ADD, REMOVE or REORDER the radio buttons. Once you have entered the values press OK to return to the attribute screen. If you want to initialize the radio buttons to start on one specific area, you can select that area be clicking on selecteditem. In this example I also renamed the component from RadioBox1 to SelArea. The SCL code below shows the newly added label section named after the pushbutton PRODREP in the Frame.




```
INIT:
return;

PRODREP:
    submit continue;
      PROC PRINT data=work.sales
      (where=(area="&SelArea.selecteditem"));
     run;
    endsubmit;
return;
```

Notice that a labeled section of for the radio box SelArea was not required. Again, we have the Frame submitting code to the SAS processor for execution, as seen by the code within the submit block. Also notice that I am passing the value of the radio box SelArea.selecteditem directly into the SAS code by prefixing it with a "&" sign, in addition, since the result is a character string I must also enclose it in quotation marks.

How could I get a full detailed listing? Instead of initializing the selecedItem attribute to one of the values, you could have left it blank. If you then recompiled and tested the Frame you would get are "error" in the execution due to **"where = (area = ' ')"** . Nothing would match the area criteria. To get around this problem you could have changed the SCL code as shown in the example below.

```
INIT:
return;

PRODREP:
    submit continue;
      PROC PRINT data=work.sales
    endsubmit;

  if SelArea.selecteditem ne ' ' then do;
    submit continue;
       (where=(area="&SelArea.selecteditem"))
    endsubmit;
  end;

    submit continue;
     ;run;
    endsubmit;
return;
```

You can see how I have broken up the code into three separate submit blocks. The second one with the where clause can only be executed when a "where subset" is desired. Now if nothing is selected from the radio box, all the detail is printed out.

You can now see how you can use SCL and submit blocks to conditionally execute SAS code. This is an extremely powerful feature and all of it has been completed without having to rely upon SAS macros etc.

The SCL code for the SALES_SUMMARY Frame is much easier, it consists of only 1 submit block using a Proc report to produce the results.

```
INIT:
return;

ProfDSummary:
    submit continue;
     title "Sugi Demo";
     PROC REPORT data=work.sales ls=122 ps=50
       split='/' nocenter nowd;
     COLUMN area sales;
     DEFINE area / GROUP FORMAT=$6. WIDTH=6
        LEFT "Area";
     DEFINE sales / SUM FORMAT=BEST9.
        WIDTH=9 COLOR=MAGENTA "Sales";
     RBREAK AFTER / OL DUL SUMMARIZE;
```

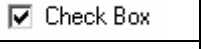
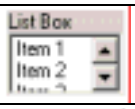
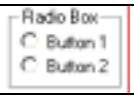
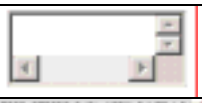
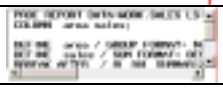
```
        run;
      endsubmit;
return;
```

C. COMMON ATTRIBUTES

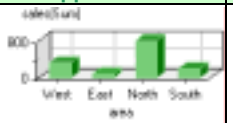
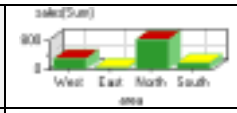
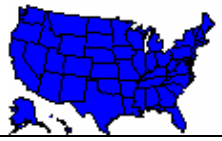
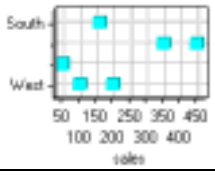
Certain attributes exist in almost all components. For example each visible component has a **NAME, WIDTH, HEIGHT, VISIBLE** attributes. Many other attributes are also common among many attributes eg. LABEL, BACKGROUNDCOLOR, BORDERCOLOR, FONT, TITLE.

When you first start to use these attributes, they might appear overwhelming, soon you will find that you become very familiar with them and can find what you are after very quickly.

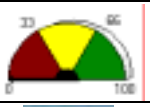
D. FREQUENTLY USED COMPONENTS

Component Name	Component Appearance	Comments or Main Attribute (MA)
Check Box		MA: selected = "Yes"
Combo Box		MA: selectedItem = ' ', supports text completion
Desktop Icon		Requires double click, more text than push button
Graphic Text		Similar to Label control generally used for large titles
List Box		MA: selectedItem=' ', also supports multiple selections, uses more space than combo box
Push Button		Can add Icons and orient text
Radio Button		MA: selectedItem = ' ', select one item from many possible choices
Text Entry		MA: text = ' ', may specify char, numeric etc, only 1 line
Text Label	Label	Generally used to label text entry controls etc.
Text Pad		MA: text = ' ', Supports multi lines text, wrapping etc.
External File Viewer		MA: filename = ' ', Used to display external text files

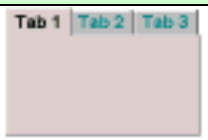
Commonly Used Graphics Control Components

Component Name	Component Appearance	Comments or Main Attribute (MA)
Chart		MA: dataSet = ' ', XVariable = ' ', YVariable = ' ', Also Supports, BAR, BOX, LINE and AREA charts
Pie		MA: dataSet = ' ', rowVariable = ' ', sliceVariable = ' ', Also supports height attribute.
Histogram		Similar to chart, can also produce vertical histograms etc.
Map		MA: mapDataSetName = ' ', mapIDVariable = ' ', Maps are available with SAS/Graph
Scatter Diagram		

Other Commonly Used Control Components

Component Name	Component Appearance	Comments or Main Attribute (MA)
Spin Box		MA: minimum = ' ' Maximum = ' ' text = ' '
Scroll Bar		MA: minimum = ' ' Maximum = ' ' value = ' '
Progress Control		MA: minimum = ' ' Maximum = ' ' value = ' '
Critical Success Factor		MA: dataSet = ' ' value = ' ' variableName = ' '
Image		MA: image = ' ' Just need to specify the location of the image
Container Box		Used frequently to nest other components to improve organization

Version 6 Objects

Object Name	Object Appearance	Comments
Tab Layout		Extremely powerful Object, allows you to place many controls on the frame, all of which are addressable by the Frame SCL.
Video Player		Used to play videos

E. USING METHODS TO COMMUNICATE WITH COMPONENTS

Using SAS Version 8, communications can now generally be done between SCL and a component by modifying attributes using dot notation. Methods also allow you to communicate with controls, when it may not be possible with just attributes.

In many ways you can do the same thing using methods as you can using attributes. For example the SCL code to hide a listbox on the frame is:

```
listbox1.visible='No';
```

The same thing can be done using a method in your SCL code:

```
listbox1._hide();
```

There are sometimes however certain methods that give you even more control over the components than is available by modifying a component's attributes. An example of this is the method shown below to gray out a ListBox rather than making it invisible.

```
listbox._gray();
```

All the methods available to a component are found under Methods as shown in Figure 9. Another example the Pmenu for the frame is changes using the SCL code:

```
_frame_._setPmenu(sasuser.sugi28.exit.pmenu);
```

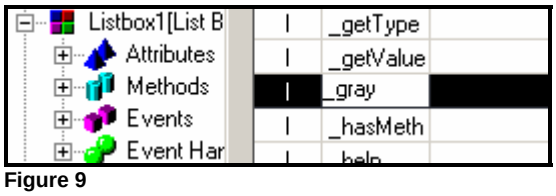


Figure 9

F. DRAG & DROP

Another method of communicating of the Frame is by Dragging a value from one control and dropping it on another. In the example below, you can see how "Imken" can be selected from the List box and simply dropped onto the test entry control, eliminating the need to type the result.

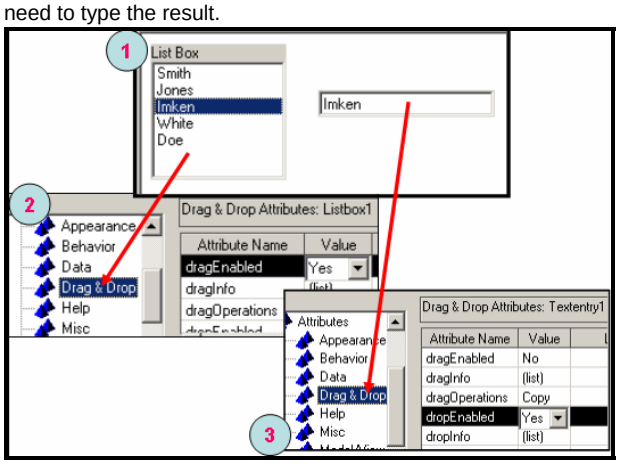


Figure 10 - Setting Drag & Drop Attributes

Drag Data from a Component and Drop it on another

#	What to Do	What Happens
1	Create a List Box Control and a Text Entry Control on the frame. Populate the List Box but keep the Text Entry blank	Components created as shown in Figure 10.
2	Right Click on List box, select Prpoerties, then drill down and click on Drag & Drop	Drag & Drop Attributes open for Listbox1
3	Set value of dragEnable to 'Yes'	
4	Right Click on Text Entry, select Prpoerties, then drill down and click on Drag & Drop	Drag & Drop Attributes open for Text Entry
5	Set value of dropEnabled to 'Yes'	
6	Test the Frame by dragging an employee from the List box to the Text Entry control	The properties screen is displayed.

F. CONTROLS vs. MODELS

In the previous example I demonstrated how you could populate a Radio Box Control by clicking on items under data attributes and manually ADDing each separate button (eg. North, South etc.). Frequently however, you might want to populate the selections from the values contained in a SAS data set instead. This is extremely useful if you have a large number of unique values.

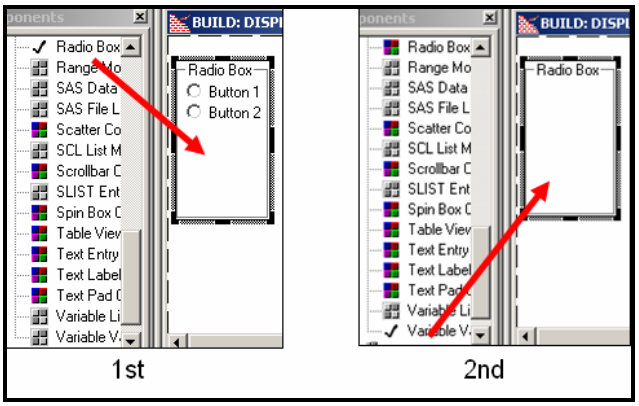


Figure 11 - Filling Controls using Model Components

This will be demonstrated below with a Radio Box Component.

#	What to Do	What Happens
1	Click on the Radio Box Control Component in the Component window and drag it over the Frame. Drop were desired.	The Radio Box is displayed with it's default attributes as shown in Figure 11
2	Click on the "**Variable Values List MODEL**" within Components, DRAG it to the Frame and Drop it OVER the Radio Box Control	The Radio Box Control default data is removed (i.e. Button #s disappear.)
3	Right click on the radio Box Control and select Properties from the selection List.	The properties Window opens as shown in Figure #12 below. Notice the addition of the Variablevalueslist1 the attached model.

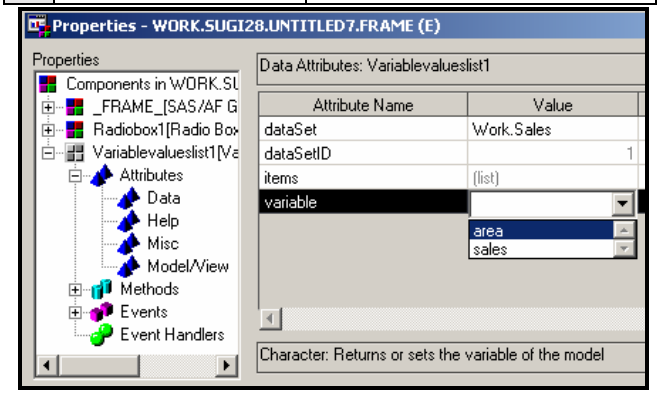


Figure 12 - Using the Variable Values List Model

4	Drill down on the attribute icons till you get to the data icon and select it.	The right hand side of the screen now displays the Data Attributes for the model Variablevalueslist1.
5	Click on the Value column beside the dataset Attribute Name	Enter the name for the data set you want to select the values from. In this case work.sales.
6	Click on the Value column beside the variable Attribute Name	All the variables in the data set which you selected will be shown
7	Select Desired Variable. Then close properties/Attributes screen.	The radio Box Control on the Frame now contains only the UNIQUE Values of the Variable you selected in alphabetic order.

This approach of overlaying a Control Component with a Model Component is extremely useful. Not only can it make it easier to enter data values, but it can make your applications more dynamic. For example, if your application contained many radio boxes on different frame which displayed employee names, a simple change to the data set containing the names (add, delete, or edit) would automatically adjust all the radio boxes within your application.

Many other Control Components support Model overlays. In the example below, I had overlaid the Color List model, first over a List Box Control, Then a Radio Box Control and finally over a Combo Box control.

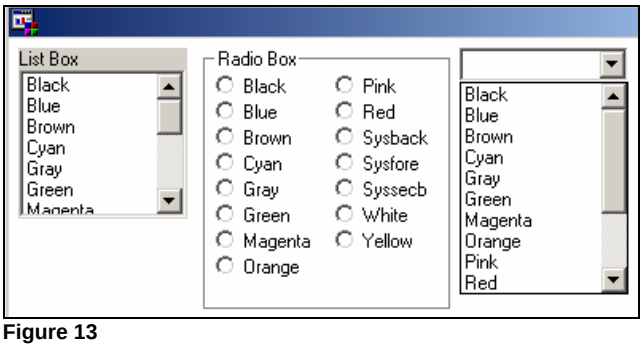


Figure 13

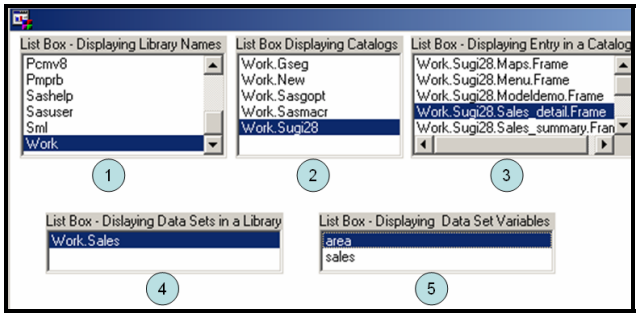


Figure 14 - Examples of filling using Models

In Figure 14 you can find a series of list boxes each which has been attached with a different model.

1. List Box Control - populated by - Library List Model
2. List Box Control - populated by - Catalog List Model
3. List Box Control - populated by - Catalog Entry List Model
4. List Box Control - populated by - Data Set List Model
5. List Box Control - populated by - Variable List Model

H. ATTRIBUTE LINKING

Figure 15 below illustrates the use of attribute linking. In the diagram on the left I have created 2 Combo Box Controls, the one labeled Select Map is attached to a Data Set List Model which points to the library Maps (SAS/Graph Map data set). When the Combo Box is clicked it will displays a list of countries for which maps are available. The second Combo Box Control is attached to a Color List Model, when clicked a list of primary colors is displayed. Below the two Combo Boxes I have placed a Map Control. Attribute linking allows one Control Component to automatically link to the attributes of a second Control Component.

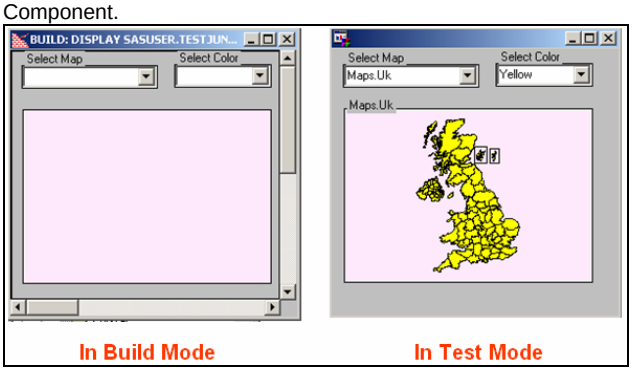


Figure 15 - Example of Attribute Linking

Normally when you create a map control you populate it using the Data Attribute "mapDataSetName". Using attribute linking you use the Linked To Column as shown in Figure 16 instead.

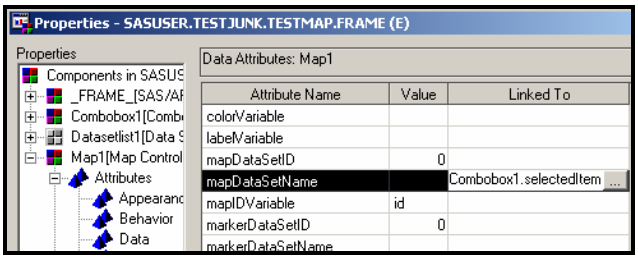


Figure 16 - Map Linked to a ComboBox

When you click on the button in the Linked To column beside the MapDataSetName (which is originally empty) you are shown a window as is displayed below.

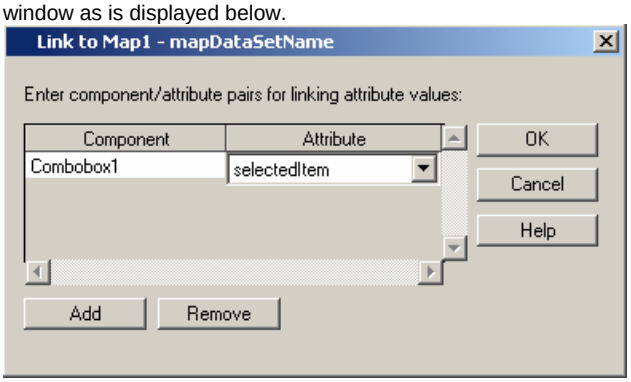


Figure 17 - Completing the Linkage

Figure 17 shows us linking Map1 to the selecteditem from ComboBox1 (labeled Select Map). Once this linkage has been done then simply clicking on the Combo Box will populate the Map Control without having to write any SCL code. The example in Figure 15 also shows a similar attribute linkage having been done for the map color attribute and the map control title.

I . USING HOT STOPS

IThe frame below has been almost filled it a Graphic Output Control which is used to display GRSEGs which have been developed using SAS/Graph. In this case I have set the graph output named "graphout" as follows:

```
graphout.graph=
    'sashelp.eisgrph.world2.grseg';
```

Next I select a Version 6 object called "HotSpot" and drag it over Africa as shown in Figure 18.

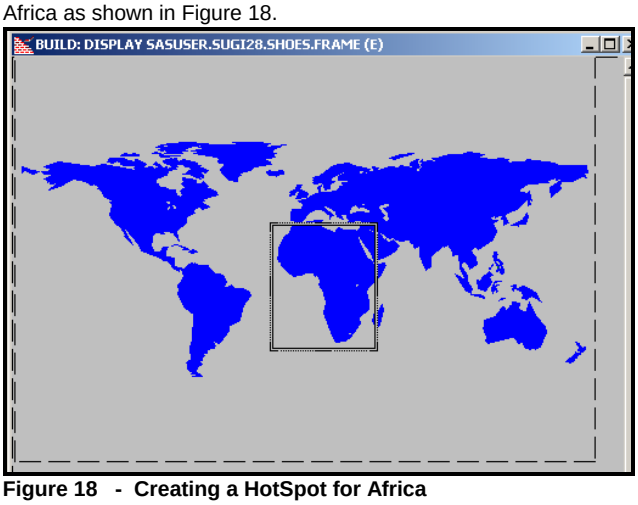


Figure 18 - Creating a HotSpot for Africa

Next right click the hot spot container you have created over Africa, and select Object attributes. That will open the screen shown in Figure 19. Here I have named the hotspot "Africa". Once the hotspot has been named, the label "Africa" can be used in your SCL code as shown below Figure 19.

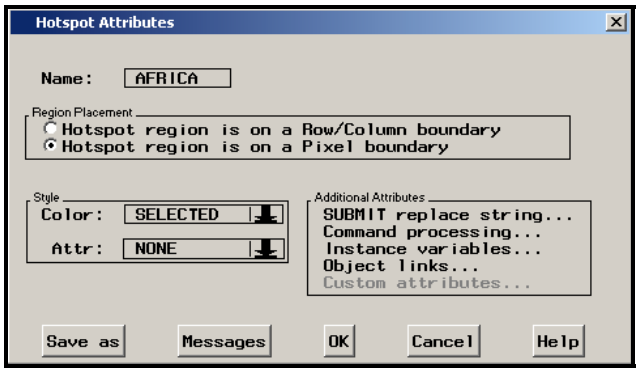


Figure 19 - Creating the Africa HotSpot

```
AFRICA:
      submit continue;
        proc print data=sashelp.shoes
                (where=(region="Africa"));run;
     endsubmit;
return;
```

When the frame with the map is now executed, the printout will be generated whenever you click on Africa on the map. Similar hotspotting could also be done to other controls such as histograms etc. When completed for a histogram, each bar of the histogram would require a hot spot to be set.

J. FORM & TABLE VIEWERS

Actually using Form and Table Viewers is another example of using a Control Component together with a Model Component. I will demonstrate this by first creating a data set called **work.employee** as seen below in Figure 20.

	name	City	Country	Phone	Position
1	Bernd Imken	Ottawa	Canada	613 952-3312	Chief
2	John Smith	Seattle	USA	999 999-9999	Manager
3	Joan White	Miami	USA	333 444-5555	President

Figure 20 - Sample Data for a Form Viewer

Figure 21 below shows how the Form Viewer is created and linked to the data set model. Steps3 and 4 show the Form Viewer in Test/Production mode.

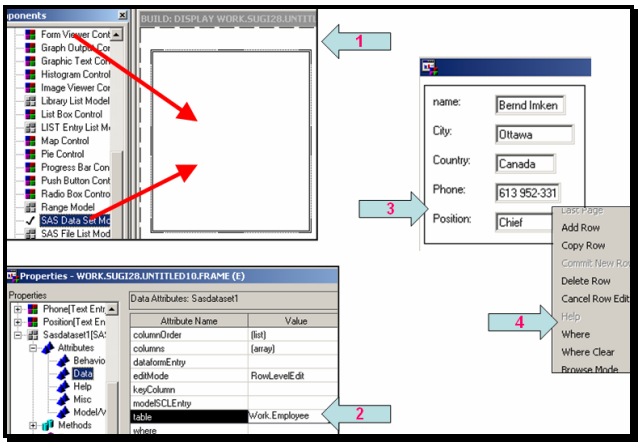


Figure 21 - Creating and Using a Form Viewer

#	What to Do	What Happens
1	Click of the Form Viewer and drag it over the Frame	An empty box is displayed
2	Click on the Data Set Model and drag it over the Control just created	Box remains empty.
3	Now right click on the empty box and select Properties from the selection menu	The properties screen is displayed.
4	Drill down on SASdataSet1 till you get to Data	Data attribute panel shown on right side.
5	For the table attribute name select the Value WORK.EMPLOYEE and close screen	Empty box get filled in by the variables in the data set. Variable values will not be displayed until you go into TEST.
6	Test the Frame and then right click you mouse	Another selection list panel is displayed, allowing you to go forwards, backwards, into edit/browse, do where searches etc.

Form Viewer Control Method examples:

```
Formviewer1._gotoRowNumber(500);   - goto OBS #500
Formviewer1._gotoRowNumber(-1);    - goto first record
Formviewer1._gotoRowNumber(-2) ;   - goto prev. record
Formviewer1._gotoRowNumber(-3);    - goto next record
Formviewer1._gotoRowNumber(-4) ;   - goto last record
```

Commonly used Data Set Model methods:

```
Sasdataset1._addRow();
Sasdataset1._commitNewRow();
Sasdataset1._rereadCurrentRow();
```

Additional discussion of methods is beyond the approaches of this paper. Check the References section of this paper for other paper which deal with it.

CONCLUSION

SAS/AF provides the application Developer with a powerful yet easy to use tool. These applications are developed to be used on a desktop which already runs SAS software. APPDEV STUDIO which has also been developed by SAS, may be used to develop applications similar in appearance and functionality. The major difference between the two is that SAS/AF must run from within an existing SAS session, whereas, APPDEV STUDIO is run from within an Internet browser. If your user is already running SAS on his desktop, using SAS/AF will eliminate Browser / HTML / JAVA overhead. Both are excellent products.

REFERENCES

Other SUGI Papers written by ***BERND E. IMKEN***

Developing SAS/AF Applications with Form Viewers and Table Viewers	***28th SAS Users Group International Proceedings***
Developing Master/Detail Applications Using Form & Table Viewers	***27th SAS Users Group International Proceedings***
A SAS Based Correspondence Management System	***26th SAS Users Group International Proceedings***
The Use of Data Forms, Tables and Other new Objects in SAS/AF	***24th SAS Users Group International Proceedings***
Improving SAS Data Access - An Evolutionary Experience	***24th SAS Users Group International Proceedings***

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Bernd E. Imken
Chief, Information Systems Division
Patented Medicine Prices Review Board
333 Laurier Ave West
Ottawa, Ontario, Canada K1P 1C1
Work: (613) 952-3312
eMail: bimken@pmprb-cepmb.gc.ca