

Developing Applications with The SAS® System

Juli Staub Perry

SAS Institute Inc., Cary NC

ABSTRACT

The desktop has become an important arena for the generation of key applications in many enterprises. It is also the playground for the most innovative technologies for applications available in the industry. This presentation will demonstrate the methodologies that SAS software uses that allow you to take advantage of graphical user interfaces (GUI) and the features of object-oriented (OO) rapid applications development. You will be introduced to the applications development tools, SAS/EIS® and SAS/AF® software. Included in the discussion will be a comparison of these two tools and how you can use each to develop applications either independently or in combination with one another. Also included is a behind the scenes look at SCL driven SAS/AF FRAME Entries.

INTRODUCTION

The intended audience for this paper are those who are somewhat familiar with the SAS System, but do not know much about its application development tools. Many books and pages and pages of documentation have been carefully written to discuss this topic, so it is not this paper's goal to teach you the details of everything you always wanted to know about SAS software's OO, GUI applications development. Instead, each section provides you with a snapshot of the numerous possibilities you have in creating a powerful interface to meet your application needs. Sprinkled throughout the paper and at the conclusion are references to more detailed works on how to accomplish the tasks described.

SAS SOFTWARE APPLICATIONS DEVELOPMENT TOOLS

SAS software provides you with several applications development tools. You may develop applications using a single tool or you may integrate the tools to create a very powerful and sophisticated application. The two tools most appropriate for this discussion are SAS/EIS and SAS/AF software. You will be given an overview of these modules later in this paper. But first, it is important to distinguish the three types of applications users and to define some of the basic concepts of object-oriented programming (OOP).

Introducing the Applications Users

Most applications entertain varying levels of users. Some users can encompass any or all of the levels, but generally speaking, applications users can be divided into three categories:

- ☐ End User
- ☐ Builder
- ☐ Developer

Traditionally, end user applications were often designed, built and maintained by in-house Information System (IS) shops or out-sourced to consultants. Often this required a structured approach to applications development. We have found, over time, this approach left end users with a sense of lack of control over their application needs. The problem is two-fold:

1. Much of the application life-cycle was spent trying to determine what the users needs were, rather than having a common philosophy about what an application should do.

2. The turn-around time for applications to finally move into production was too long. The end users needs would often change before the production application was ready for use.

Not surprisingly, most applications end users pleaded for a way to rapidly and iteratively create their own applications using a point-and-click approach. That is, they are not necessarily technically inclined to roll up their sleeves and crank out code; nor do they have the desire to spend hours creating a customized application that may be too powerful for their current needs. But they need a quick way to retrieve and summarize their data without having to tap into IS sources. These applications users can be referred to as applications *builders*.

Nonetheless, the applications end users and the applications builders still have a need for even more powerful customized applications. And they still need those in the IS shop to add additional sophisticated functionality to their applications. The people who are trained with the expertise and who can provide that service can be referred to as applications *developers*. Note, however, applications developers can also take advantage of the applications builders' tools. That is, the applications builders tools can be used by applications developers to iteratively develop prototypes or pieces of a production application quickly and easily for the end user to start using. As you will see below, SAS/EIS and SAS/AF software lend themselves well to iterative applications development for the end users, the builders and the developers.

Overview of Object Oriented Technology

Object-oriented technology is not just a way of programming, it is a way of thinking. In fact, object-orientation typically consists of three stages: Object Oriented Analysis, Object Oriented Design and finally Object Oriented Programming (OOP). This paper focuses primarily on the SAS System's toolset for iteratively developing OO-based GUI constructs to create a complete integrated application solution.

A brief description of some OO terminology is in order, however, because both of the SAS System's applications development components discussed below take advantage of these concepts.

Conceptually, object-orientation focuses on grouping data into categories and defining actions that can be performed on the data. These data categories, called *classes* are used as a working template for creating data objects. An *object* consists of both data and services. The data provide information about the attributes of the object (*properties*), and the services (*methods*) define the behavior of the object.

For example, a medical chart for a hospital containing information about a patient is an object. You could name that object CHART. Among the properties of that chart might be a patient's name, a patient's age group, his/her sex and detailed information pertaining to the office/hospital visits made. The types of things you could do with the chart would be to view it, print it, fax it, edit it and delete it. In OO, each of these actions are considered *methods*. So, the class is named CHART and it contains properties for the chart. It also knows who its parent class is. The CHART class only contains a template with no concrete values. It does not contain information about a given patient until it becomes an *object*. Once an object, it becomes a real entity and contains information about a single patient. Specifically, when an *instance* is created for "John S. Patient", the object will contain

information unique to him. For example, he falls into the 35-45 age group category, he is male and his records contain a history of his past yearly check-ups.

Some objects can be as simple as a visual application control (*widget*) and some objects can themselves be entire applications. In the example just mentioned, the chart could be either visual or non-visual. All the objects, whether visible or non-visible, are set up in a class hierarchy, and each object automatically *inherits* the characteristics of its parent class, unless otherwise specified. If you alter an object's original behavior, you can override the object's inherited characteristics by creating a *subclass* of the parent class.

Both SAS/EIS and SAS/AF software support objects of these two types.

APPLICATIONS DEVELOPERS TOOLSET

SAS/EIS software and SAS/AF software are two of the components of the SAS System. They are applications development tools that serve as a GUI-based, object-oriented gateway to the many decision support, reporting, and analytical tools of the SAS System and provides access to a variety of external or third party software packages.

Applications Development: A Builder's Approach

SAS/EIS software uses an object-oriented approach to provide quick access to integrated information. Pre-written application objects offer ways for you to explore data in both traditional and innovative ways. Interactive drill-down and hotspotting methods are inherent in many of the objects provided by the software. SAS/EIS's built-in objects allow rapid prototyping of what otherwise could be a very time consuming development process. For instance, you can create a GUI-based, decision support application without having to write any code. For this reason, SAS/EIS software is a good tool for applications *builders* to use. It is also a productive tool for applications *developers* because it follows the methodology of iterative development.

Recall that following the concepts of OOP, an object consists of both data and methods. Recall also that there can be a distinction between the types of objects defined. SAS/EIS software provides over 30 *application objects*, simplifying tasks ranging from performing host system commands to accessing key informational data to building sophisticated menuing GUI-based interfaces to other application components.

The following list shows you sample SAS/EIS objects available. These objects provide ways to view data in making key business decisions without having to write any code:

Business Graph	Charts and plots based upon data defined to the application. Includes drill down, dynamic selection of other variables in the data set, dynamic changing of the chart or plot type, and other sophisticated display features.
Comparison Report	A three column report accompanied by a horizontal bar chart to view analysis variables and their variance. Drill-down capabilities are supported.
Expanding Report	Report to display multiple columns with drill-down capabilities to retrieve more detailed information while still having the ability to view summarized data. Graphical views of the data are also available.

Forecast	Time-series forecasts produced by the FORECAST and AUTOREG procedure in SAS/ETS® software. It provides the ability to perform what-if analysis by interactively changing independent variables to determine their effect on the dependent variable.
Grouped Bar Chart	Overlapping bar chart used to display values of selected variables. Drill-down capabilities are supported.
Multi-Column Report	Report to display multiple columns of data accompanied by an optional graph representing the data shown. Drill-down capabilities are supported.
Remote Connect	Signon application used to execute SAS/CONNECT® software. Enables access to remote data.
Variance report	Variance application to display values for two variables, showing the difference and percent difference between each pair of values. Includes ability to define drill-down hierarchy as defined to EIS, display graphs of the data, define hotspots and target applications on the reports. Ranges may also be defined to highlight data trends.

Of course, providing a mechanism to view and analyze data is very important in making sound business decisions. But, providing a gateway to that information is just as important. You can use SAS/EIS objects to create GUI-based interfaces. Two of these types of objects are described below:

Block Menu	Menu application containing icons on a single window that are selected to execute other SAS/EIS applications.
Graphics Menu Builder	Point-and-click menu desktop application. Graphical images with SAS/AF FRAME entry elements (ie. standard widgets, user-defined classes of widgets, graphical text and SAS/GRAPH® output) can be used to construct the menu application.

You can choose to use any of the above objects to create a complete GUI application or you can customize the built-in objects to meet your diverse user needs. You can respond to users' requirements by expanding an existing object, or creating completely new objects using SAS/AF software.

Defining Your Data to the SAS/EIS Data Dictionary

Many of the objects provided by SAS/EIS require that the data it presents to you be registered in its metabase. The metabase behaves like a data dictionary and is used to manage and organize your data so any object you create knows what to do with it.

Column attributes of a table include analysis, class, x and y independent and dependent, actual and budget and a host of others. In fact, you can define a table hierarchy by defining the columns as drill attributes.

Suppose you are working with medical data that includes information about patients and their medical coverage. You or your end user are interested in subsetting the data and want to view it using a hierarchical structure. Data is stored in a SAS data set and can be referred to as MEDDOC.EISDATA. The drill variables are shown in Table 1.

PLANNBR	Insurance Plan Number
PROVIDER	Patient's Doctor
VISITGRP	Type of Visit
AGEGRP	Age Group of Patient
SEX	Sex of Patient

Table 1 Hierarchical Data Structure for Medical Insurance Claims

You are also interested in analyzing the total charges and the number of visits for the above hierarchy. In your data dictionary, you can register them as ANALYSIS variables. Table 2 lists the ANALYSIS variables.

TOTCHGS	Total charge amount for each patient
VISITS	Total number of visits for each patient

Table 2 Analysis variables for the Claims hierarchy

You invoke SAS/EIS by either issuing `EIS` on the command line or by traversing the pull-down menus.

Globals → Develop → EIS application builder

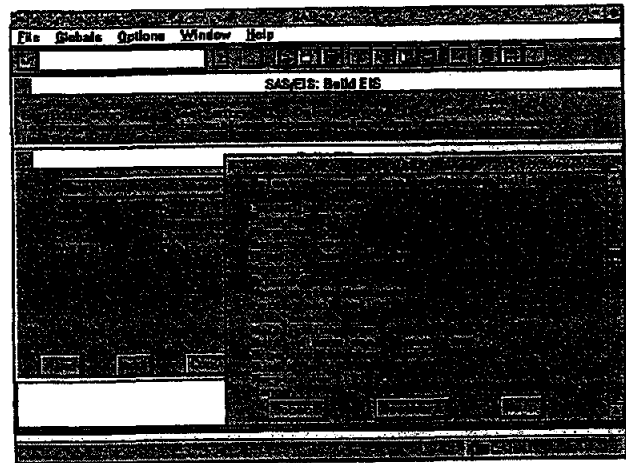
The SAS/EIS main menu appears and you select **Metabase** to register your data and create the hierarchy described above. Once this is done, whenever you create an object that refers to the registered data, the application will automatically take advantage of its definition.

Note: *Getting Started with SAS/EIS Software, Version 6, 1st Edition*, pages 7-14 provides you with a good summary of how to register your data or data table views in the metabase.

Using a SAS/EIS Application Object

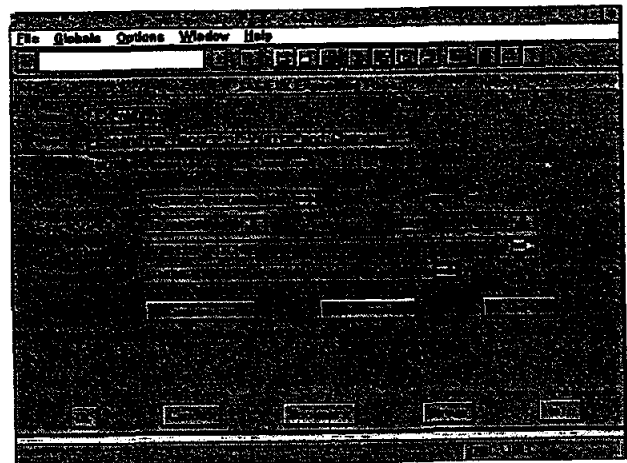
You use an *application database* to store definitions for the application objects you create. To create an application object, you select the **Build EIS** icon from the main menu of SAS/EIS. You then select (or create) your application database and click on **Add** from the command menu at the bottom of the window.

Display 1 illustrates a selection list of available application objects. The list of objects include both the default SAS/EIS application objects as well as those objects you have defined. One such pre-defined object is named *Expanding Report*. The *Expanding Report* object not only enables you to view a data table on a high level, but it also allows drills into the data by expanding the report for the selected items. The expansions can be collapsed as well. As with all of the SAS/EIS pre-defined objects, the *Expanding Report* has its own set of attributes unique to it and you must enter in the information required to create the object successfully.



Display 1 SAS/EIS in Build Mode with Add Window Displayed

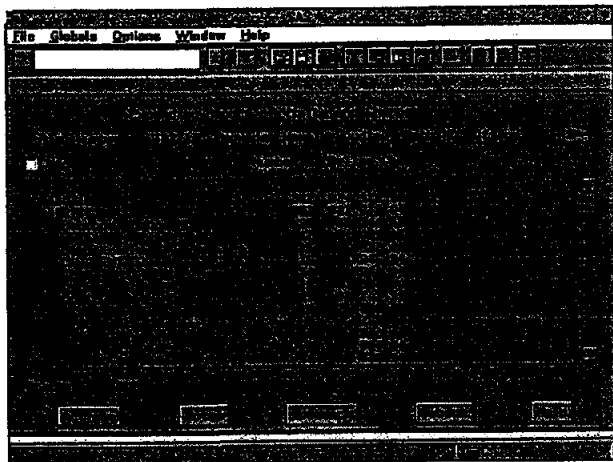
The following example shows you how to create the application object for an *Expanding Report* using the medical claims data described previously in the *Defining Your Data Dictionary* section. Assume you have just selected **Expanding Report** from the list of available application objects. The object's attribute window is displayed (Display 2).



Display 2 Expanding Report Attributes Window

Notice this window requires you to specify a data source, the drill hierarchy, and any variables you want calculated via the statistic you select. The data source can be a local SAS data set (or view), a remote SAS data set (or view) using remote library services, or any table supported by the SAS Systems data SAS/ACCESS® products. You are also given the option to customize your object. You can customize colors, fonts, labels, and even the default chart type.

After you have successfully filled in all of the required fields and any customizations you want to make, you can immediately test the object by clicking on the **Test** button. The result of the test, is illustrated in Display 3.



Display 3 Expanding Report During a Test

Once displayed, you can click on the text of the report and drill-down to the next level of the hierarchy you specified in your data dictionary. A **chart** option is also automatically available for your use to give you a graphical view of your data table. If you select the **chart** option and then click on a line item, a chart pertaining to the item selected will be displayed. You also have the option to change the chart's type by selecting:

Options —> Select Chart Type

and choose between, a pie, a vertical bar or a horizontal bar chart.

Note, the fascinating part of all of this, is that *you didn't have to write a single line of code* to create this fully functional application module.

SAS/EIS software also provides you with the ability to override the default object's behavior to extend and customize your own application object. To accomplish this, you have several options. One option is to use the features of SAS/EIS and override default features. Another option is to link to an existing application object that is either written in SAS or one written with SAS/AF. And still another option, for the applications developer is to create an entirely new customized application object from scratch using SAS/AF and Screen Control Language. You can then add the application object to the available SAS/EIS objects. For complete details on how to do this, refer to *Extending SAS/EIS Software Capabilities, Version 6, First Edition*.

Applications Development: A Developer's Approach

SAS/AF software takes full advantage of the latest technology in applications development. With the advent of the FRAME entry, a design component of SAS/AF software, applications developers can design and implement applications using object-oriented concepts. These applications commonly contain objects such as pushbuttons, scroll bars, list and radio boxes, images and diverse graphics.

FRAME Entries

The SAS/AF software FRAME entry employs OOP techniques to make applications development intuitive and easy to manage. You can create a toolbox of commonly used graphical user interface elements as well as non-visual elements.

SAS/AF FRAME entries are the windows used to create your interface or application objects. They are driven by Screen Control

Language (SCL), a 4GL, OO-based programming language that includes variables, various data structures and an assortment of functions, routines, and statements. SCL source for a FRAME entry is stored in a separate SCL entry and can be shared by other FRAME entries. The name of the SCL program associated with the FRAME entry is stored as one of the entry's general attributes.

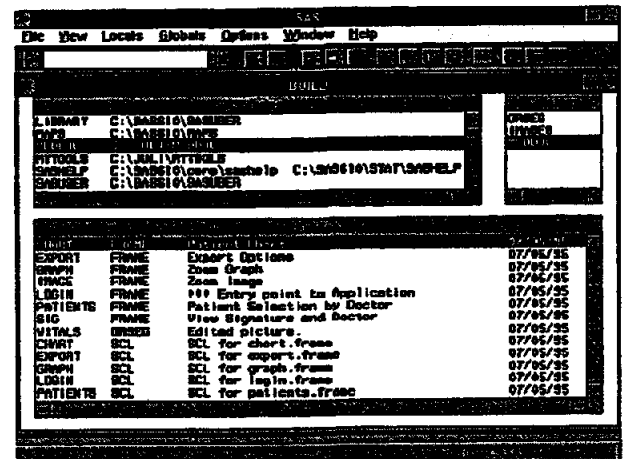
FRAME entries provide you with several techniques for defining visual objects (controls or widgets) used to create an application interface. You can choose a text-based approach, a GUI approach or integrate the two. Whatever approach you choose, you can view the displays in the applications development mode just as they appear during run mode.

Creating the FRAME Entry

Development of SAS/AF FRAME entries requires a vector graphics display device. A mouse is also highly recommended because the development environment relies on point-and-click and drag-and-drop to create and maintain text-based widgets as well as graphical widgets. The end user is not required to run the FRAME entry application on a vector graphics display device, but if one is not used, only text-based widgets can be displayed. It is also strongly recommended that the end user run the application with the use of a mouse.

Creating a new FRAME entry is straight-forward. You can do this by either issuing the **BUILD** command on the command line in any of the SAS System's windows or via the pull-down menus.

Globals —> Develop —> Application builder



Display 4 A SAS/AF Application Development Environment

You can use a point-and-click approach to traverse to or create your *application catalog*. The application catalog is used to hold the entries associated with your application. *Note:* One or more application catalogs can be used for a single application. To create a new FRAME entry, you can use the pull-down menus:

File —> New —> Entry

A dialog box appears where you can specify the name of the new entry and indicate that its entry type is FRAME.

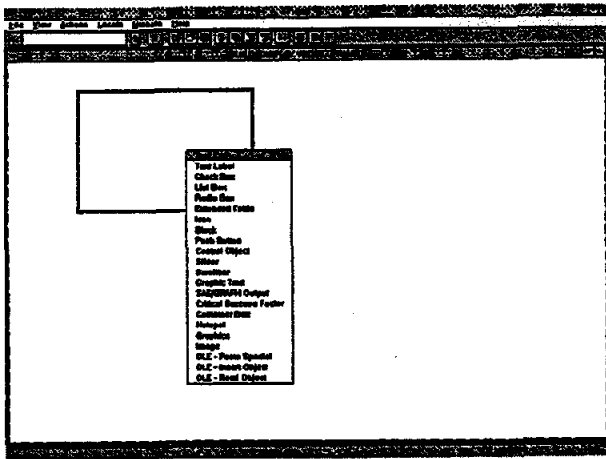
FRAME entries use a graphical object editor called the region manager as the development environment. Thus, once you indicate you want to create or edit an entry, the region manager displays for you to define the appropriate widgets. With the region manager, you have complete control over the widgets' size,

rectangular shape and placement.

The region manager display is part of a FRAME entry's overall structure, but it depends on a set of predefined classes to accept user input or display information. You use the classes to create visible objects or widgets on the display.

One way you can define a widget is to create a rectangular region by pressing the leftmost mouse button, dragging the mouse pointer across the display and releasing the mouse button when the rectangle is the size you want. To fill the region with a widget, press the rightmost mouse button anywhere in the empty region. Select **F111** from the menu option.

After you select **F111**, the application displays another menu that lists the names of all available predefined classes used to create widgets. Display 5 shows this resource list.



Display 5 Creating a Widget on a FRAME Entry

When the resource list appears, select the appropriate class for the widget you want to create.

The widgets and the display work together by sending messages to one another through a set of methods defined to them. To send messages, you can use the CALL NOTIFY routine in your SCL program, specifying the method you want to apply to the designated widget.

Each widget also has an individualized attribute window enabling attribute specification for the selected widget. Information such as the name of the widget is stored here. The name given to a widget indicates the name of an SCL variable and the name of an optional labeled section in your SCL allowing

- ☐ automatic execution of labeled sections of code when users perform an action on an object
- ☐ message sending to the object by use of the NOTIFY routine
- ☐ the ability to assign information to the widget, retrieve information from it, or alter its appearance.

In addition to benefiting from the FRAME entry's supplied objects, you can also define your own customized objects by sub-classing an object's class. Whenever you frequently use a widget with the same attributes, you can speed the development by creating a new class or by sub-classing the existing class. Then, you can add the new class to the resource list, making it available for creating widgets in any application. This lends itself well to keeping the look and behavior of your widgets consistent across your applications.

Creating an Application Module with the FRAME Entry

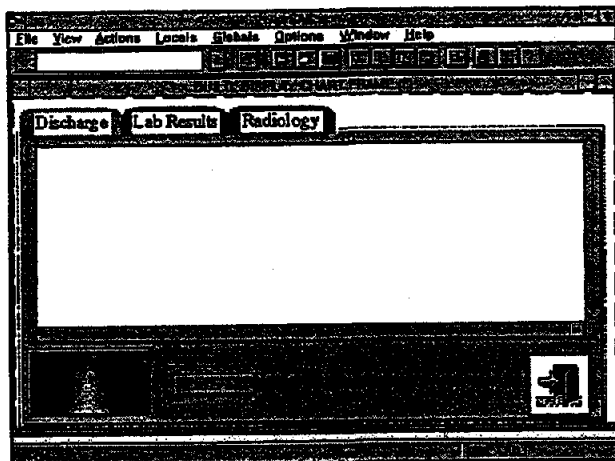
Suppose you want to create a patient chart similar to the one described in the section titled *Overview of Object Oriented Technology*. In this scenario, you want to keep demographic information pertaining to a selected patient (for example, John S. Patient) as well as his medical history. The files contained within the chart are *discharge*, *radiology*, and *lab results*. Depending upon the file selected, you want different actions to be performed. For example, when "discharge" is selected, you want to simply retrieve his hospital discharge information and display it to the end user. But when "radiology" is selected, you want to retrieve the radiology file for the patient as well as any x-ray images that are stored in the system. And when "lab results" is chosen, you want to retrieve his lab result file and give the end user the option to look at the "micro" results. This type specification lends itself well to using a simplistic GUI "file tab" metaphor.

Although SAS/AF software and SCL allow many development alternatives from which to choose, here is just one solution in its most rudimentary form.

1. Create an application catalog. Name it MEDDOC.MEDDOC.
2. Create a new FRAME entry. Name it CHART.
3. Create a *container box* widget to represent a file. Name it FILE.
4. Create a *list box* widget and place it inside the container box. The list box will be used to display the textual contents of that file. Name it OUTPUT and indicate that it is to be populated with the contents of an SCL list named OUTLST.
5. Create *SAS/GRAPH output* widgets to represent file tabs for each patient chart category. Arrange them side by side at the top edge of the container box. Name them TAB1-TAB3. And indicate they are to initially be populated with GRSEGS that look like file tabs located in MEDDOC.MEDDOC.
6. Create an *image* and a *push button* to appear for use when a particular file tab is selected. Place them side by side at the bottom left corner of the window underneath the file. Name the image IMAGE, and populate the it with a sample x-ray and then use it as a push button to access and display the patient's file of x-rays. To make a widget look like a push button you enter its region attributes and alter its outline. During runtime, the x-ray will only appear if the "Radiology" file tab is selected. Similarly, the push button widget will only appear if the "Lab Results" file tab is selected. Name it MICRO and label it with the text "micro." During runtime, "micro" will be used to display the Micro-tab results of the selected patient.
7. Create a different *image* widget to serve as an exit button. Place and size it appropriately at the bottom right hand corner of your FRAME entry, underneath the file. Populate it with the stored exit image, EXIT.IMAGE in your application catalog. Name it EXIT and set its region attributes to indicate it will serve as a push button.

In **BUILD** mode, the FRAME entry looks like Display 6.

In this example, almost every widget created on the FRAME entry contains a labeled section of code with the same name as the widget. The labeled sections are written to either send messages to other widgets on the screen or to branch to different FRAME entries or to link to other sections of code.



Display 6 CHART.FRAME in Build Mode

Notice in the following code that the widgets TAB1 and OUTPUT do not have associated labeled sections. Neither of these widgets require a unique action to take place if selected. OUTPUT does not require any action and TAB1 simply needs to link to the labeled section UPDTTAB. But this is performed in the MAIN section that executes each time a mouse action or the enter key is pressed on a widget.

Sample code for CHART.FRAME (stored in CHART.SCL) is illustrated below.

```
init:
  length ichar $1 text $80 widget $8;
  link hideimg;
  link hidemic;
  outlst = makelist();
  return;
main:
  link updttab;
  return;
term:
  rc = dellist(outlst);
  return;
tab2:
  call notify('micro','_unhide_');
  return;
tab3:
  call notify('image','_unhide_');
  return;
micro:
  dsname = 'micro';
  link fillout;
  return;
image:
  call display('image.frame');
  return;
```

/* Cont. */

```
updttab:
  call notify('.', '_GET_CURRENT_NAME_', widget);
  tab = substr(widget, length(widget));
  do i = 1 to 3;
    ichar = put(i, 1.);
    if tab = put(i, 1.) then do;
      call notify('tab' || ichar, '_SET_GRAPH_',
        'meddoc.grseg.tab' || ichar || '.grseg');
      call notify('chart', '_SET_REGION_COLOR_',
        put(tab, $regcolr.));
    end;
    else
      call notify('tab' || ichar, '_SET_GRAPH_',
        'meddoc.grseg.tabc.grseg');
      call notify('tab' || ichar, '_UPDATE_');
      if tab ^= '2' then link hidemic;
      if tab ^= '3' then link hideimg;
    end;
  link fillout;
  return;
fillout:
  dsname = put(tab, $dsname.);
  rc = clearlist(outlst);
  outid = open('meddoc.' || left(dsname));
  do while (^fetch(outid));
    text = getvarc(outid, varnum(outid, 'TEXT'));
    rc = insertc(outlst, text, -1);
  end;
  rc = close(outid);
  call notify('output', '_repopulate_');
  return;
hideimg:
  call notify('image', '_hide_');
  return;
hidemic:
  call notify('micro', '_hide_');
  return;
exit:
  link term;
  _status_ = 'H';
  return;
```

0 SCL code stored in CHART.SCL

COMPARING THE APPLICATIONS MODULES

If you consider yourself an applications *builder*, then SAS/EIS software is a very good tool for you to rapidly develop an application without writing any code. You can use a point-and-click approach to tap into your data and present the results to you. If, however, you consider yourself an applications *developer* and need to customize and sophisticate your applications to meet your user's diverse needs, then the SAS/AF FRAME entry and SCL are the tools of choice. Each tool provides a unique development environment.

The Applications Development Environments

As you have noticed from the previous examples demonstrating the development of a SAS/EIS application object and a SAS/AF FRAME entry, the applications development environment for these two tools differ. (Display 1 and Display 4). This is because each tool is at a different level of applications development. Just as the SAS software languages sit a layer above C, SAS/EIS sits one OO layer higher than SAS/AF. In fact, the application objects in SAS/EIS software were written using SAS/AF software and SCL.

Summary of Applications Development Modules

Below is a summarized comparison of the two applications development modules. You can use it to help decide which module is most appropriate to use given the list of characteristics.

App Area	SAS-EIS	SAS-AF
Recommended App User	App Builder or App Developer	App Developer
Access	EIS command or EIS Application Builder pmenu	BUILD-command or procedure or Application builder pmenu
Invocation	RUNEIS statement	AF or AFAPPL statement
Storage of App modules	Application Database	SAS Catalogs
Coding	Does not require SCL or Base SAS Code	Commonly has one or many SCL entries to manage its actions
Application Branching	Associated Target Application	Any number of methods may be used
Visual Portability	No necessary preparations	Recommended use of attachments*
Data Dictionary	Utility provided	Utility not provided
Attribute Control	Unique for each application object	Unique for each object
Sub-classing Objects	Use SAS-EIS Object Manager	Use Class Editor

Table 3 Comparison of SAS/EIS and SAS/AF Software

INTEGRATING THE APPLICATIONS MODULES

In addition to using the two application development modules separately, you can use SAS/EIS software in conjunction with SAS/AF software to create a very powerful application for decision support (or any other type of application). In the following example, you will be shown a few alternatives for combining SAS/EIS and SAS/AF software. This also demonstrates how the applications developer can easily work with the applications builder to form a completed intuitive application.

Invoking a SAS/AF Module from SAS/EIS

Suppose you would like to use both the *Chart* example and the *Expanding Report* examples above in a single application. That is, you would like to create an interface and access both SAS/AF and SAS/EIS software from a single entry. You have several options. One option is to call the *Chart* (MEDDOC.MEDDOC.CHART.FRAME) module directly from the *Expanding Report*. A different alternative is to create a new SAS/EIS object, the *Graphics Menu Builder*, to link the two modules together.

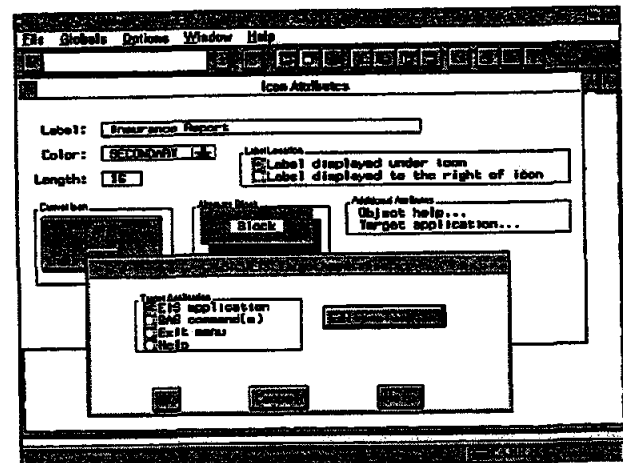
Creating a Menu Application Object

The *Graphics Menu Builder* allows you to create GUI interfaces using FRAME entry technology and make direct invocations to other SAS/EIS objects.

When you select the *Graphics Menu Builder* object and enter into BUILD mode, you are taken into SAS/AF's region manager window and are editing a FRAME entry. This object is a subclass of the FRAME entry. That is, when you fill a region, you are given a subset of the WIDGET class master list to create. This subset represents all objects that are used as menu picks in a FRAME entry.

Another difference between the FRAME entry and the *Graphics Menu Builder* is that the General Attributes offer you a way to point to a *Target Application*. This attribute allows you to specify a call to another SAS/EIS object, invoke a SAS command, directly exit out of the menu, or display a help entry.

You want to specify a call to another SAS/EIS object. For example, create a *Graphics Menu Builder* object and name it MAINEIS. In BUILD mode, create an Icon widget and point it to the EIS application, MEDDOC.MEDDOC.EXPANDER.EXPANDER, in the *Target Application* window. Save and test. Then, create another application object, the SAS/AF *Application* object and name it CHART. In its attributes, point it to MEDDOC.MEDDOC.CHART.FRAME. Once save, it will be stored in the application database as CHART.AF. When you create the second icon and name it EXPANDER, you can point the target application to MEDDOC.MEDDOC.CHART.AF.



Display 7 MAINEIS.DESKTOP in Build Mode

Calling Chart from the Expanding Report

Many of the SAS/EIS objects provide you with the capability of specifying a target application to run. So, just as you can specify a target application as a characteristic of each control (widget) on the *Graphics Menu Builder* FRAME entry, you can specify an associated target application for an EIS object.

For example, suppose instead of creating an interface to invoke the *Chart* and the *Expanding Report* examples separately, you instead want to call the *Chart* from the *Expanding Report*.

Edit the *Expanding Report* (EXPANDER.EXPANDER) and in the *Customize* window, select *Target Application*. In the *Target Application* dialog box, click on *Edit value* to edit the EIS application. Here, you traverse the catalog manager and identify MEDDOC.MEDDOC.CHART.AF as your EIS Application. Save

and return to the *Expanding Report* attribute window. To test, click on Test. Once the report appears, click on the *Options* button and select the *Target Application* option. You will be brought directly into the SAS/AF Chart module.

Adding CHART to the SAS/EIS Application Object List

You can alternatively add the SAS/AF CHART entry to the SAS/EIS object list by using the *Object Manager*. Although this option is a viable one, it is beyond the scope of this paper. You can refer to the SAS Institute Inc. publication, *Extending SAS/EIS Software Capabilities, Version 6, First Edition*, for details on how to create your own SAS/EIS application objects.

Invoking a SAS/EIS Module from SAS/AF

Other alternatives to integrating SAS/EIS and SAS/AF software is to call SAS/EIS application modules directly from SAS/AF. As you may have guessed, there are a variety of ways this can be done. Three of these are to call the SAS/EIS module from the *Command Processing* option in the object attributes of a given widget; or you could use SCL to invoke the module; or you might want to link to a SAS/EIS module from SAS/AF when SAS/EIS is already up and running.

Using the Command Processing Option

You can use the *Command Processing* option in the *Object Attributes* of the object you want to invoke the SAS/EIS module.

Closely following the scenario of creating a *Graphics Menu Builder* object in SAS/EIS, you could create a FRAME entry to serve as your interface to link to the *Expanding Report* and *Chart* modules separately.

First, create a FRAME entry and name it MAINAF. Create two widgets. Typically, menu-pick widgets are icons, SAS/GRAPH output, or Images. You have a variety of options with the list of classes the FRAME entry provides. If you wanted, you could even create a radio box with two items to link to each module. But, in this example, create two Image widgets.

Create the first. Because the object you create is a widget within a FRAME entry, as opposed to a SAS/EIS application object, it must be named. You could keep the default name, OBJ1, or you could provide more meaning to your SCL code by naming it REPORT (or whatever other meaningful name you choose). Initiate the attributes window for REPORT and click on *Command Processing*. In the dialog box for *Command Processing*, type in the following command.

```
runeis appl=meddoc.meddoc.expander.expander
```

Save and exit to the build window of MAINAF.FRAME. **NOTE:** Since this example does not require any SCL, you must go into the *General Attributes* window and delete any reference to an associated SCL entry. You are then free to *testaf* the entry.

Using SCL to invoke a SAS/EIS Module

You can alternatively use the *execcmdi* command to invoke the SAS/EIS module. For example, you name a widget REPORT and in your SCL code you would have the following labeled section.

```
report:
  call execcmdi('runeis
    appl=meddoc.meddoc.expander.expander');
return;
```

The call to CHART.FRAME from MAINAF.FRAME would be made by issuing a CALL DISPLAY in the labeled section for the second icon widget you name CHART.

```
chart:
  call display('CHART.FRAME');
return;
```

Both of the above options start another task, however, and are best suited for single invocations of a SAS/EIS object. You may find that you need to call another SAS/EIS object from within an AF object when a SAS/EIS task is already running. In this case, you can use the set of system defined macros.

Using the %EISENTRY Macro

You can invoke another SAS/EIS object from within a SAS/AF Application object. To do this, you indicate to use *Default EIS Parameters* in the SAS/AF Application object. Then, you place the macros invocations EIS and EISENTRY in your associated SCL entry. To invoke the EIS object, you use the following statement.

```
call display('sashelp.eis.runeis.program',
  &eparmlst,
  'meddoc.meddoc.expander.expander', rc);
```

The RUNEIS PROGRAM entry runs the specified object, MEDDOC.MEDDOC.EXPANDER.EXPANDER. &EPARMLST is a system macro variable used by SAS/EIS that passes the appropriate values for an object to run successfully. RC is a system supplied return call.

THE FUTURE OF THE SAS SYSTEM APPLICATIONS DEVELOPMENT TOOLS

The SAS software System version 6.11 is moving forward to production. Here is a brief summary of enhancements and new tools to come in both SAS/AF and SAS/EIS modules.

The Enhanced Applications Builder Toolset for 6.11

This highlights a few of the features offered with SAS/EIS software, version 6.11.

- ☐ Enhanced Development environment interface
- ☐ Improved Metabase Facility
- ☐ Extended list of application objects that do not require any SCL to make them run successfully.
 - ☐ Multidimensional Report Object
 - ☐ Organizational Chart Object
 - ☐ Graphical Variance Report Object (Experimental)
 - ☐ 3D Business Graphs Object
 - ☐ Graphics with Hotspots Object
 - ☐ Catalog Entry Viewer with Hotspots Object
 - ☐ External File Viewer with Hotspots Object

- ❑ New Client/Server Features
- ❑ Improved application objects such as expanded features of the Graphics Menu Builder Object.

The Enhanced Applications Developer Toolset 6.11

This highlights a few of the features offered with SAS/AF software, version 6.11.

- ❑ Continued support and enhancements for Object Oriented Development through sub-classing, methods, inheritance, delegation and more.
- ❑ New graphical data tools that allow viewing, editing, data validation and data appearance control.

Data Table Allows viewing or editing of your table in a tabular format, similar to a spreadsheet.

Data Form (Experimental) Allows viewing or editing of your table by displaying the data a row at a time.

Any customizations you make to either of these new objects can be saved permanently and used when specified. You can also point to an SCL entry different than that of the associated FRAME entry to write modules unique to the data table or form.

- ❑ An abundant set of new graphical widgets added to the existing set shown in Display 5.

Catalog Entry Viewer Displays the text records of a SAS catalog entry.

Command Push Button An enhanced push button widget that represents an action. It pushes a command that is either selected from the available commands or a customized command. The command button can be shaped to any size.

Extended Input Field An enhanced Text Entry Field. It is a graphical component to allow more flexibility in sizing widgets on the screen. It supports automatic scrolling, the use of host fonts, marking of text, user-defined colors, and paragraph-style word wrap.

Extended Text Entry An enhanced Text Entry Field. It is a graphical component to allow more flexibility in sizing widgets on the screen. Only character values can be contained in it and there is no control over format or informats, but initial values can be set for each row. Like the Extended Input Field, it supports scrolling, marking of text, and paragraph-style word wrap.

Input Field An enhanced Text Entry Field. It can be used to display one-line character or numeric text fields that can accept user input. Control over formats and informats, scrolling support, and the ability to mark text is provided.

Input Field Label Displays non-edit-able graphical text to be used as a label resource.

Image Icon A graphical version of the Icon class. It provides the same functionality of the Icon object, but has some additional options such as changing the image size.

Image Viewer Enables you to display bit-mapped images from an external file, a fileref that points to an external file, or a IMAGE catalog entry. For images larger than the widget, scrolling capabilities are provided.

OCX (Ole Control) Provides access to a wide range of native controls. Two sample OCXs (a text entry widget and a combo box widget) will be available with the software.

Organizational Chart Automatically creates hierarchical charts from data stored in either SAS data sets or SCL lists. During execution time it allows for change in the visual aspects of the tree. This can be accomplished by using the pop-up menu provided or by opening the object's attribute window.

Tool-bar A bar consisting of one or more image and/or label buttons designed to perform a specific action when selected. Allows complete control over the widgets placement, type of behavior (ie. push button, check box, or radio button), and drag and drop functionality.

Video Player Support of multi-media. Allows video clips to be played with developer control over all of the typical playback features.

Work Area Scroll-able area used to group and manage objects. The widgets in a work area can represent data, tasks and even complete applications. Drag and drop capabilities are inherent and it is also possible to dynamically create and position new widgets during execution time. It is also possible to save the work area layout for reuse.

- ❑ Drag and drop capabilities between components on a window and even across windows in your SAS session.
- ❑ The ability to combine graphical components on a window into a single composite widget that can be saved as a class and reused by other applications. This is an extremely powerful feature.
- ❑ An improved *class editor* and *instance variable editor*.
- ❑ Support of EVENT driven applications.
- ❑ Ability to dynamically create widgets on a window during execution time.

This provides you with only a high level overview of the upcoming features of the SAS System, version 6.11. Refer to the references at the end of this paper for a guide to more detail.

THE FINAL WORD

As a rule of thumb, SAS/EIS is a good tool for rapidly building applications that fit the scope of the system's applications objects. These objects can either be supplied by the software or be customized objects developed via SAS/AF software by applications developers. SAS/AF software, along with a powerful applications programming language, provide applications developers with a way to customize user interfaces. Using an object-oriented approach to programming, you can easily obtain, manage and display the information necessary to help the end user make sound business decisions. You can think of SAS/AF as beginning where SAS/EIS leaves off. Because of this, you can

employ the application modules separately, enhance the functionality of one with the other, or integrate the two.

With the advent of version 6.11 on the horizon, the breadth of changes will undoubtedly assist the application user in creating sophisticated GUI applications. New application objects provided with SAS/EIS software will provide the application builder with a wealth of new data driven objects, a greatly improved development interface, and major enhancements to the metabase facility. SAS/AF software will provide the applications developer with drag and drop capabilities, new GUI object classes, and a stronger, yet even more powerful OO development environment.

Every new release of the SAS System for information delivery continues to improve, not only by its ability to combine a powerful system for data access, management, analysis and presentation, but also to provide you with an ever-growing interactive applications development toolset. Whether you consider yourself an applications builder or an applications developer, you can use the object-oriented GUI tools in SAS/EIS and SAS/AF software to respond quickly and effectively to users' evolving needs.

References and Suggested Reading

PAPERS

Finding Business Solutions by Extending SAS/EIS Applications, Merry Rabb, SAS Consulting Services Inc. and Ben Cochran, SAS Institute Inc. SUGI 19, pp. 83-91.

SAS/EIS Software: A Tutorial, Scott Isaacs and Gail Kramer, SAS Institute Inc. SUGI 18, pp. 1326-1332.

Zooming In On New SAS/AF Objects, Tammy Gagliano, SAS Institute Inc., 1995.

If You Had to Choose a Racing Partner, Would You Choose the Tortoise or the Hare?, Tammy Gagliano, SAS Institute Inc., 1995.

Effective Utilization of Region Attachments in a FRAME Application, Bud Whitmeyer, SAS Institute, Inc., Cary, NC, 1995.

DOCUMENTATION

SAS Institute Inc., *Getting Started with the FRAME Entry: Developing Object-Oriented Applications, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1991. 81 pp.

SAS Institute Inc., *Getting Started with SAS/EIS Software, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1993. 64 pp.

SAS Institute Inc., *Extending SAS/EIS Software Capabilities, Version 6, First Edition*, Cary, NC: SAS Institute Inc. 1994. 136 pp.

SAS Institute Inc. (1993) *SAS/EIS Software: Reference, 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., *SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute, Inc., 1994. 648 pp.

And any new material written by SAS Institute Inc. accompanying Version 6.11 of the SAS System for Applications Development.

SAS, SAS/ACCESS, SAS/AF, SAS/CONNECT, SAS/EIS, and SAS/ETS are registered trademarks of the SAS Institute Inc. in the USA and other countries. ® indicates a USA registration.