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 the graphical user interfaces (GUI) and the features of object-oriented (OO) rapid applications development. You will be introduced to two development tools, SAS/AF® and SAS/EIS® 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 applications 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 development tools. You may develop applications using a single environment 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 between two types of applications users and to define some of the basic concepts of object-oriented programming (OOP).

Introducing the Applications Users
Different levels of developer expertise often play a significant role in today’s applications development life-cycle. The individual expertise required to build a powerful application can encompass any number of levels; but generally speaking, applications users can be divided into two categories:

- Applications Assemblers
- Applications Developers

Traditionally, end user systems were often designed, built and maintained by in-house Information Technology (IT) shops or outsourced to consultants. Often this required a structured approach to applications development. We have found, over time, this approach left users with a sense of lack of control over their application needs. The problem is three-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 the system 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.
3. Because diverse sources were often used to create various applications, a duplication of efforts most likely occurred and therefore, promoted lack of application integration.

Not surprisingly, most end users pleased a way to rapidly and iteratively create their own applications using a point-and-click modular 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 application objects they may use in only one place in their system. But they need a quick way to retrieve and summarize their data without having to tap directly into IT sources. These users can be referred to as applications assemblers.

Nonetheless, the assemblers still have a need for even more powerful customized applications. And they still need those in the IT shop to provide them with application components or objects that will give them access to additional sophisticated functionality. The people who are trained with the expertise and who can and want to provide this valuable service can be referred to as applications developers. Although developers often focus their energy on creating customized and reusable objects with fourth generation languages, they can also take advantage of the assemblers tools. That is, the assembler’s more component-based tools can be used by 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 assemblers 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 tool set for iteratively developing OOP-based GUI constructs to create a complete integrated system solution. A brief discussion of some OOP terminology is in order, because both of the SAS System’s applications development components discussed below take advantage of OOP 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 gender and detailed information pertaining to the office/hospital visits made. The types of actions you could perform on 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. The CHART class is merely a template with no concrete values. It does not contain information about a given patient until it becomes an object. Once an object, it is 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 of the objects, regardless of visibility, reside in a class hierarchy and each object automatically inherits the characteristics of its parent class, unless otherwise specified. If you need to alter an object's original behavior, you can override the object's inherited characteristics by creating a subclass of the class. The original class then becomes the parent of the newly created subclass.

Both SAS/EIS and SAS/AF software support and follow OO constructs.

**APPLICATIONS DEVELOPMENT TOOL SET**

SAS/EIS software and SAS/AF software are two 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. They also provide access to a variety of external or third party software packages.

**Applications Development: An Assembler’s Approach**

SAS/EIS and SAS/AF software use an object-oriented approach to provide quick access to integrated information. In SAS/EIS, pre-written application objects (business objects) offer ways for you to explore data in both traditional and innovative ways. Interactive drill-down and hot-spotting methods are inherent in many of the objects provided by the software. SAS/EIS software'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 when there is a need to quickly assemble an end user business application.

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 business 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 business objects. Each category provides you with a way to create business objects so you can view data needed in making key business decisions. Remember that each of the objects can be used to assemble a completed application without having to write any code:

<table>
<thead>
<tr>
<th>Object</th>
<th>Application Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHART</td>
<td>Creates dynamic charts</td>
</tr>
<tr>
<td>GRAPH</td>
<td>Displays dynamic charts</td>
</tr>
<tr>
<td>GRAPHS</td>
<td>3D business graphs</td>
</tr>
<tr>
<td>GROUPBAR</td>
<td>Grouped bar charts</td>
</tr>
<tr>
<td>HOTGRAPH</td>
<td>Graphics with hotspots</td>
</tr>
<tr>
<td>ORGCHT</td>
<td>Organizational charts</td>
</tr>
<tr>
<td>PLOT</td>
<td>Creates dynamic plots</td>
</tr>
<tr>
<td>CATVIEW</td>
<td>Catalog entry viewer with hotspots</td>
</tr>
<tr>
<td>COMPARe</td>
<td>Comparison report</td>
</tr>
<tr>
<td>EXPANDER</td>
<td>Expanding report</td>
</tr>
<tr>
<td>FILEVIEW</td>
<td>External file viewer with hotspots</td>
</tr>
<tr>
<td>REPORT</td>
<td>Displays existing reports</td>
</tr>
<tr>
<td>MULTICOL</td>
<td>Multi-column report</td>
</tr>
<tr>
<td>MULTI</td>
<td>Multidimensional report</td>
</tr>
<tr>
<td>FORECAST</td>
<td>Forecast time series</td>
</tr>
<tr>
<td>CALC</td>
<td>Execute existing spreadsheet application</td>
</tr>
<tr>
<td>VARREP</td>
<td>Produce variance reports</td>
</tr>
<tr>
<td>PREDICT</td>
<td>What-if analysis</td>
</tr>
<tr>
<td>FILENAME</td>
<td>Assign a fileref</td>
</tr>
<tr>
<td>LISNAME</td>
<td>Assign a libref</td>
</tr>
<tr>
<td>SIGNON</td>
<td>Manage remote sessions</td>
</tr>
<tr>
<td>EDIT</td>
<td>Edit or browse SAS data sets</td>
</tr>
<tr>
<td>BLOCK</td>
<td>Create block/Icon menus</td>
</tr>
<tr>
<td>DESKTOP</td>
<td>Create graphical menus</td>
</tr>
<tr>
<td>LISTMENU</td>
<td>Creates list menus</td>
</tr>
<tr>
<td>CSF</td>
<td>Monitor Critical Success Factors</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>Execute a SAS program</td>
</tr>
<tr>
<td>VIEWFILE</td>
<td>Edit or browse external files</td>
</tr>
<tr>
<td>SYSICMD</td>
<td>Execute host system commands</td>
</tr>
<tr>
<td>LETTER</td>
<td>Use existing letter app</td>
</tr>
<tr>
<td>RANGE</td>
<td>Define data ranges</td>
</tr>
<tr>
<td>SASCMD</td>
<td>Execute a SAS command</td>
</tr>
</tbody>
</table>
The SAS/EIS main menu appears allowing you to build a new complete GUI application or you can customize the built-in objects using SASIAF software.

Table 1 - Business Objects available with SAS/EIS Software
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 user’s diverse needs. You can respond to users' requirements by expanding an existing object, or creating completely new objects using SASIAF software.

**Invoking SAS/EIS Applications Development Environment**
You invoke SAS/EIS by either issuing EIS from the command line/window or by traversing the pull-down menus.

Globals → Develop → EIS application builder

The SAS/EIS main menu appears allowing you to build a new application, maintain an existing one, perform utility functions and define your data sources to the data dictionary.

**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. When you register data in a metabase, you assign attributes to each data set and data set variable. This is achieved through the Table and Columns Attribute windows.

Column attributes of a table include analysis, class, x and y independent and dependent, actual and budget and a host of others. Table attributes include information about the table you are defining. For example, common attributes are the table location, pre and post processing criteria, and any drill-down hierarchies.

Suppose you are working with the 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. Specifically, you would like to analyze patient costs under given provider plans and the providers within those plans. In addition, you want to slice the data and view the results categorized by the type of visit the patient has, their age and their gender. All of this can be done in a table format by utilizing the business object, Multihier. First, you must set up two sets of hierarchies for the data to be used in the table. You use the Table Hierarchies window located in the Metabase define your data levels. Press Add in Table Attributes and then select HIERARCH to get to this window. The hierarchy set ups are shown in Table 2 and Table 3. Data is stored in a SAS data set and can be referred to as MEDMOC.EISDATA.

<table>
<thead>
<tr>
<th>Variable/Column</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLANNBR</td>
<td>Insurance Plan Number</td>
</tr>
<tr>
<td>PROVIDER</td>
<td>Patient’s Doctor</td>
</tr>
</tbody>
</table>

**Table 2 Hierarchical Data Structure for Medical Insurance Claims**

<table>
<thead>
<tr>
<th>Variable/Column</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VISITGRP</td>
<td>Type of Visit</td>
</tr>
<tr>
<td>AGEGRP</td>
<td>Age Group of Patient</td>
</tr>
<tr>
<td>SEX</td>
<td>Gender of Patient</td>
</tr>
</tbody>
</table>

**Table 3 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 4 lists the ANALYSIS variables.

<table>
<thead>
<tr>
<th>Variable/Column</th>
<th>Column Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTCHGS</td>
<td>Total charge amount for each patient</td>
</tr>
<tr>
<td>VISITS</td>
<td>Total number of visits for each patient</td>
</tr>
</tbody>
</table>

**Table 4 Analysis variables for the Claims Hierarchy**
To access the metabase in the applications development environment for SAS/EIS, you select the Metabase icon from the main menu. Here is where you register your data and create the hierarchy described above. Once this is done, the application will automatically take advantage of its definition.


**Using a SAS/EIS Business Object**
You use an application database to store definitions for the business objects you create. To create a business object, you first select Build EIS from the main menu. 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 business objects. The list of objects includes both the default SAS/EIS business objects as well as those objects you or your object administrator has incorporated into your EIS. Related objects are grouped together in object databases. The default data bases include Business reports, Business Graphs, Data Access, Utilities, and Menus. One such pre-defined Business Report object is named Multidimensional Report. The Multidimensional Report object not only enables you to view a data table on a high level, but it also allows navigation into the data by drilling (or expanding) the report for selected items. In addition, you or your end-user may subset the report, rotate it, add or delete columns, or alter the appearance of the report. As with all of the SAS/EIS pre-defined objects, the Multidimensional Report has its own set of attributes unique to it.

Display 1 SAS/EIS Applications Development Environment with Add Window Displayed

The following example shows you how to create the business object Multidimensional Report using the medical claims data described previously in the Defining Your Data Dictionary section. Assume you have just selected Multidimensional Report from the list of available Business Report object databases. Select the Build button on the right. The object's attribute window is displayed (Display 2).

Notice this window requires you to specify an object name, a data source (Table), the columns to display, optional subsetting criteria, and a column you want summed. 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 specify titles, customize colors, fonts, labels, formatted values, default hierarchies, indicate computed values you want displayed, alter the report layout and you can also specify the default expansion technique.

Display 2 Multidimensional Report Attributes Window

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 Multidimensional Report During a Test

Once displayed, you can click on the text of the report and drill down or expand to the next level of the hierarchy you specified in your data dictionary. With many of the newer objects, your user has the ability to alter the business object’s appearance. In fact, many of the same build-time options are available for the end user to modify during run-time. Any changes the user makes can be saved in the user’s application profile, while still preserving the assembler’s default report settings. Also, the key part of all of this is that you don’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 business object. To accomplish this, you have several options. One option is to use the features of SAS/EIS and override default methods. Another option is to link to an existing business 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 business object from scratch using SAS/AF and Screen Control Language (SCL). You can then add the business object to the available SAS/EIS objects. For complete details on how to do this, refer to SAS/EIS Software: Reference, Version 6, Second 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, developers can design and implement applications using object-oriented concepts. These applications commonly contain objects such as push buttons, list and radio boxes, data entry tables, viewers, images and diverse graphics. You also have the ability to perform drag and drop operations, send events between components on a window and even across windows in your SAS session. Additionally, you can add, delete and modify objects on a window during execution time. A class and instance variable editor is also provided.

FRAME Entries

The SAS/AF software FRAME entry employs OOP techniques to make 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, business or application objects. They are driven by Screen Control Language (SCL), a 4GL, COBased 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 one or more separate SCL entries and can be shared by other FRAME entries or objects. The name of the SCL program associated with the FRAME entry is stored as one of the entry's general attributes.

Creating the FRAME Entry

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 Applications 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 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 a list of available widgets. With the region manager, you have complete control over the widgets' size, rectangular shape and placement.

The following is a sampling of some of the more recently added controls along with their descriptions that are available in SAS/AF software:

<table>
<thead>
<tr>
<th>Widget/Control</th>
<th>Description of Widget/Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push button control that represents an action. It pushes a command that is either selected from the available commands or a customized command</td>
<td></td>
</tr>
<tr>
<td>Provides you with the ability to combine graphical components on a window into a single widget that can be saved as a class and reused by other applications. Allows viewing or editing of your table in a tabular format, similar to a spreadsheet. SAS/FSP is required to use this control</td>
<td></td>
</tr>
</tbody>
</table>

Displays the text records of a SAS catalog
Allows viewing or editing of your table by displaying the data row by row at a time.

A graphical component to allow more flexibility in sizing controls on the screen. Supports automatic scrolling, the use of host fonts, marking of text, user-defined colors and paragraph-style word wrap.

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

Enables you to display bit-mapped images from an external file, a file ref that points to an external file, or an IMAGE catalog entry. Scroll capabilities are provided.

Enables you to display a list of data values that may be used as a means of selection.

Provides access to a wide range of native controls. Two sample OCXs (a text entry and combo box) are available.

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 widget's placement, type of behavior (i.e. Push button, check box, or radio button), and drag and drop functionality.

Support of Multi-media. Allows video clips to be played with developer control over all typical playback features.

Scrollable 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.

---

Display 5: Creating a Widget on a FRAME Entry

When the resource list appears, select the appropriate class for the control you want to create. The above example (Display 5) illustrates several existing visual widgets on the FRAME entry display and a new widget (middle-right-hand side) being created.

One way the widgets and the display work together is by sending messages to one another through a set of methods defined to them. To send messages, you can either use the CALL NOTIFY or the CALL SEND routines in your SCL program, specifying the method you want to apply to the designated control.

Each control also has an individualized attribute window enabling attribute specification for the selected control. Information such as the name of the control is stored here. The name given to a control 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 control, 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 and object's class or by combining a set of objects together to form a completely new 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 for the patient. 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.

- Create an application catalog. Name it MEDDOC.MEDDOC.
- Create a new FRAME entry. Name it CHART.
- Create a container box control/widget to represent a file. Name it FILE.
- 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 OUTLST.
- 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 be initially populated with GRSEGs that look like file tabs located in MEDDOC.MEDDOC.
- 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 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 run-time, 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-lab results of the selected patient.
- Create a different image control to serve as an exit button. Place and size it appropriately (or better yet, create a subclass of the image widget to use throughout your application) and place 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 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.

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

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

```scl
init:
  length ichar $1 text $80 widget $8;
  link hideimage;
  link hidemic;
  outlst = makelistO;
  return;
main:
  link updttab;
  return;
term:
  rc = dellist(outlst);
  return;
lab2:
  call notify,micro:"_unhide_"
  return;
lab3:
  call notify(image:"_unhide_"
  return;
image:
  call display('image.frame');
  return;
updttab:
  call send_,set_;_GET_CURRENT_NAME_;_widget);
  tab = substr(widget,length(widget));
  do i = 1 to 3;
    ichar = put(i,1.);
    if tab = put(i,) then do;
      call notify(tab,ichar:"_SET_GRAPH_"
      'meddoc:grseg,tab,ichar,grseg');
      call notify(chart,"_SET_REGION_COLOR_"
    end;
```
Applications Development

Applications Development

Figure 1 CHART.SCL used to drive CHART.FRAME

COMPARING THE APPLICATIONS MODULES

As you have noticed from the previous examples demonstrating the development of a SAS/EIS business object and a SAS/AF FRAME entry, the applications development environment for these two tools differs. Refer to Display 1 and Display 4. This is because each tool is at a different level of development. SAS/EIS is a high-level, component-based application tool. It consists of objects that have already been written with SAS/AF. Often, SAS/EIS software can be associated with an assembler because the system being built is done by combining and gathering a framework of components based on the function each one is designed to perform. SAS/AF, on the other hand, is often associated with a developer because you can use the extendibility capabilities to create re-usable objects for the assembler to integrate into his application. This is not to say that SAS/EIS should only be used by assemblers and SAS/AF should only be used by developers. On the contrary, the two compliment each other quite nicely.

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.

Table 5 Comparison of SAS/EIS and SAS/AF Software

<table>
<thead>
<tr>
<th>Application Description</th>
<th>SAS/EIS Software</th>
<th>SAS/AF Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended Application Description</td>
<td>App Assembler or App Developer</td>
<td>EIS command or EIS Application Build menu</td>
</tr>
<tr>
<td>List of Attributes</td>
<td>Application Database</td>
<td>RUNEIS statement</td>
</tr>
<tr>
<td>Description of Application</td>
<td>Does not require SCL or Base SAS Code</td>
<td>Application Template</td>
</tr>
<tr>
<td>Invoking a SAS/AF Module from SAS/EIS</td>
<td>Associated Target Application</td>
<td>No necessary preparations</td>
</tr>
<tr>
<td>Associated Component</td>
<td>Utility provided</td>
<td>Use SAS/EIS Object Manager</td>
</tr>
</tbody>
</table>

INTEGRATING THE APPLICATION 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 system 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 developer can easily work with the assembler 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 Multidimensional 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. On option is to call the Chart (MEDI_DOC.MEDDOC.CHART.FRAME) module directly from the Multidimensional Report. A different alternative is to create a
new SAS/EIS object, the Graphics Menu Builder, to link the two modules together.

Creating a Graphics Menu Builder Object

The Graphics Menu Builder allows you to create GUI interfaces using FRAME entry technology by providing a means for you to link directly to or incorporate other SAS/EIS objects. In addition, EIS applications located on a single Graphics Menu object have the intelligence to send messages to one another.

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 creation. This subset represents all objects that are used a menu picks in a FRAME. Notice that in addition to the FRAME entry’s list, additional items are included, such as the ability to insert several different EIS Applications into a single 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:INSURANCE.MULTIDIM

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:FRAME. Once saved, it will be stored in the application database as CHART:AF. When you create the second icon and name it MULTIDIM, you can point the target application to MEDDOC.MEDDOC:CHART:AF.

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. Refer to the SAS Institute Inc. publication, Extending SAS/EIS Software Capabilities, Version 6, Second Edition, for details on how to create your own SAS/EIS business objects.

Invoking a SAS/EIS Module from SAS/AF

Other alternatives to integrating SAS/EIS and SAS/AF software is to call SAS/EIS business objects 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.

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 business object.

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 Multidimensional Report and Chart modules separately.

First, create a FRAME entry and name it MAINAF. Create two widgets. Typically, menu-pick controls are icons, SAS/GRAPH output, or Images. You have a variety of options with the list of classes of 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 controls.

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

```text
runexes appl=meddoc.meddoc.insurance.multidim
```

Figure 2 Invoking SAS/EIS from the SAS/AF FRAME Entry Command Processing Attribute

Save and exit the build window of MAINAF.FRAME. You are then free to Test the entry.

Using SCL to invoke a SAS/EIS Business Object

You can alternatively use the execcmdn 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.

```text
Report: 
  call execcmdn(‘runexes appl=meddoc.meddoc.insurance.multidim’);
  return;
```

Figure 3 Invoking SAS/EIS from SAS/AF

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. Figure 4 illustrates this.

```text
Chart: 
  call display(‘CHART:FRAME’);
  return;
```

Figure 4 Invoking a SAS/AF FRAME entry from another SAS/AF FRAME entry
THE FINAL WORD

As a rule of thumb, SAS/EIS is a good tool for assemblers to rapidly build applications that fit the scope of the system's business objects. These objects can either be supplied by the software or be customized objects developed via SAS/AF software by developers. SAS/AF software, along with a powerful applications programming language provide developers with a way to customize user interfaces and provide the assemblers with a set of business objects and classes. 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 decision. You can think of SAS/AF as beginning where SAS/EIS leaves off. With this, you can employ the tools separately, enhance the functionality of one with the other, or integrate the two.

With the advent of SAS 6.11 (The Orlando Release), the breadth of enhancements assist the assembler and developer in creating sophisticated GUI applications. New business objects provided with SAS/EIS software now provides the assembler with a wealth of new data driven objects, a greatly improved development interface, and major enhancements to the metabase facility. SAS/AF software provides the developer with drag and drop capabilities, new GUI object classes, and a stronger, yet simple to use OOP 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 application development tool set. Whether you consider yourself an assembler or a 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


