Finding Business Solutions by Extending SAS/EIS® Applications
Merry Rabb, SAS Consulting Services Inc.
Cary, NC
Ben Cochran, SAS Institute Inc. Cary NC

ABSTRACT

SAS/EIS software provides an interactive applications development environment and toolkit that can be very useful for building end-user applications. The application objects provided in the product range from ones that are used to issue a simple command to the much more complex Variance Report or Business Graphics application objects.

Applications developers who are new to SAS/EIS sometimes find that the pre-existing objects provide much of the functionality they need in their EIS (executive information system), but only up to a point. Eventually, they get to the stage where they must extend their EIS by making use of the other tools provided as part of the SAS® System. This paper addresses some strategies for extending your SAS/EIS applications. We present examples of business information delivery needs that you may have and then present present us a possible solution. This paper addresses the following basic categories of techniques:

- using existing objects creatively
- extending the functionality of existing objects
- using SAS/Ap application and Screen Control Language (SCL).

USING EXISTING EIS OBJECTS CREATIVELY

SAS/EIS software comes with 27 prewritten objects that allow the user to perform many of the SAS System’s most popular tasks. The Add window, shown in Figure 1, displays the applications that can be performed with the default objects.

<table>
<thead>
<tr>
<th>Select a type of application to build:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign a fileref...</td>
</tr>
<tr>
<td>Assign a libref...</td>
</tr>
<tr>
<td>Block menu...</td>
</tr>
<tr>
<td>Business graphs...</td>
</tr>
<tr>
<td>Critical success factors...</td>
</tr>
<tr>
<td>* Execute a SAS program...</td>
</tr>
<tr>
<td>External file...</td>
</tr>
<tr>
<td>Forecasting...</td>
</tr>
<tr>
<td>General reporting...</td>
</tr>
<tr>
<td>Graphic menu builder...</td>
</tr>
<tr>
<td>Letter application...</td>
</tr>
<tr>
<td>List menu...</td>
</tr>
<tr>
<td>Output entry...</td>
</tr>
<tr>
<td>Range...</td>
</tr>
<tr>
<td>Remote connect...</td>
</tr>
<tr>
<td>* SAS commands...</td>
</tr>
<tr>
<td>SAS data set...</td>
</tr>
<tr>
<td>* SAS/Ap application...</td>
</tr>
<tr>
<td>* Script of SAS/EIS applications...</td>
</tr>
<tr>
<td>Source entry...</td>
</tr>
<tr>
<td>* Spreadsheet application...</td>
</tr>
<tr>
<td>Variance report...</td>
</tr>
<tr>
<td>Welcome window...</td>
</tr>
</tbody>
</table>

Figure 1  Add Window ("Applications discussed in this paper")

These objects allow the users a great deal of flexibility and power in building an EIS application. This section of the paper discusses the things that can be done with SAS/EIS that make it such a useful product. This section is divided into five parts that will cover the following examples:

- using the execute object to create a drill-down report
- using the spreadsheet object to create a hotspot
- using the script object to combine other objects
- using the AF object to execute a SAS/Ap application
- a miscellaneous section showing access to other SAS software products.

The REPORT Procedure

Using the REPORT procedure, one can write a program that has drill-down and subsetting capabilities. This allows the user to go from a summary report to a detailed report. The program can then be stored in a catalog and referenced by the execute object in the SAS/EIS application.

First, the user must create and store the report definitions that generate the necessary reports. In this example, the report with the most detail will be generated first. The report will be created from the SASUSER.BIZINT91 data set and will have a column for the variables NATION, INDUSTRY, COMPANY, SALES, and PROFITS. Submit the following code from a windowing environment to begin the process:

```
proc report data=sasuser.bizint91 prompt;
run;
```

The structure of the data set is such that there are several companies within each industry, and there are several industries within each nation. After selecting only the columns desired, the first five rows of the report will have the following appearance:

<table>
<thead>
<tr>
<th>NATION</th>
<th>INDUSTRY</th>
<th>COMPANY</th>
<th>SALES</th>
<th>PROFITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>Oil</td>
<td>YPF</td>
<td>$5,000</td>
<td>$300</td>
</tr>
<tr>
<td>Australia</td>
<td>Oil</td>
<td>Broken Hill</td>
<td>$12,474</td>
<td>$1,118</td>
</tr>
<tr>
<td>Australia</td>
<td>Oil</td>
<td>Pioneer Intl</td>
<td>$3,476</td>
<td>$128</td>
</tr>
<tr>
<td>Australia</td>
<td>Beverage</td>
<td>Foster’s</td>
<td>$5,490</td>
<td>$3-4</td>
</tr>
<tr>
<td>Australia</td>
<td>Food</td>
<td>GFW</td>
<td>$3,095</td>
<td>$86</td>
</tr>
<tr>
<td>Austria</td>
<td>Metals</td>
<td>Austrian Ind</td>
<td>$14,548</td>
<td>$24</td>
</tr>
</tbody>
</table>

The variables are defined as follows: NATION is a display variable; INDUSTRY and COMPANY are group variables; and SALES and PROFITS are analysis variables. Store the report in SASUSER.SUGI.REPORT1.

The second report will be generated from the same data set, SASUSER.BIZINT91 and will have columns for NATION, INDUSTRY, SALES and PROFITS.
The second report has the following format:

<table>
<thead>
<tr>
<th>NATION</th>
<th>INDUSTRY</th>
<th>SALES</th>
<th>PROFITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>Oil</td>
<td>$5,000</td>
<td>$300</td>
</tr>
<tr>
<td>Australia</td>
<td>Beverage</td>
<td>$5,450</td>
<td>$-34</td>
</tr>
<tr>
<td></td>
<td>Food</td>
<td>$3,095</td>
<td>$86</td>
</tr>
<tr>
<td></td>
<td>Oil</td>
<td>$15,950</td>
<td>$1,247</td>
</tr>
<tr>
<td>Australia</td>
<td></td>
<td>$24,535</td>
<td>$1,289</td>
</tr>
</tbody>
</table>

The variables are defined as follows: NATION and INDUSTRY are defined as group variables and SALES and PROFITS are defined as analysis variables. When defining the INDUSTRY variable, select the [Edit Program] pushbutton inside the DEFINITION window. In the COMPUTE window, write the following program:

```sql
call define('industry', 'command',
            'rload sasuser.sugi.reporting1;
            where industry="[\'\[industry\]\\"\\"]');
```

Store the report in SASUSER.SUGLREPORT2.

The third report will be generated from the same data set and will have columns for NATION, SALES, and PROFITS. NATION is defined as a group variable, while SALES and PROFITS are defined as analysis variables.

<table>
<thead>
<tr>
<th>NATION</th>
<th>SALES</th>
<th>PROFITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>$5,000</td>
<td>$300</td>
</tr>
<tr>
<td>Australia</td>
<td>$24,535</td>
<td>$1,299</td>
</tr>
<tr>
<td>Austria</td>
<td>$14,548</td>
<td>$24</td>
</tr>
<tr>
<td>Belgium</td>
<td>$33,720</td>
<td>$777</td>
</tr>
</tbody>
</table>

In defining the NATION variable, be sure to activate the [Edit Program] pushbutton inside the DEFINITION window. In the COMPUTE window, write the following program:

```sql
call define('nation', 'command',
            'rload sasuser.sugi.reporting1;
            where industry="[\'\[nation\]\\"\\"]');
```

Store the report in SASUSER.SUGLREPORT3.

Submit the following to generate the summary report:

```sql
proc report data=sasuser.bizint91
    report=sasuser.sugi.reporting3;
    title 'Proc Report
          footnote 'F1 to Drilldown, use F2 to Drillup';
run;
```

While in the REPORT procedure, define your function keys so that F1 is 'EXECUTE' and F2 is defined as 'WHERE UNDO; END'. Store the program in SASUSER.SUGLINITIAL.SOURCE.

Next, get into SAS/EIS, select the Build EIS button, select an application database, then select the [Add] pushbutton. When the Add window opens, select the Execute a SAS program... application. Supply a name and a definition, then select the Source in catalog... option. When the Source in Catalog window opens, supply the four-level name of the program location: SASUSER.SUGLINITIAL.SOURCE.

Go back to the Execute a SAS Program window and select [Test] to execute the PROC REPORT application. Initially you will see report number 3, the one with a single line for each country. Move the cursor to any country and press the ENTER key, and then press F1 (or the function keys that is assigned the EXECUTE command). You then see report number 2, which is subset for the country upon which the cursor was placed in the previous report. While viewing this report, you can then move the cursor to any industry, press ENTER, and then press F1 (or again, the function key that is assigned the EXECUTE command). Now you are taken to report number 1, which is subset based on the industry that was just chosen in the previous report. In other words, the report now shows all the observations for the industry within the country originally chosen. To return, press F2, or the function key with the "WHERE UNDO;END" definition. You are then taken back to report number 2, where another industry may be chosen, or F2 may be pressed to return you to the initial report. Another country may now be chosen or you can press END to go back.

SAS/CALC® Software

Along with drill-down capabilities, another commonly requested feature of SAS/EIS is hotspotting. One method of hotspotting that is not commonly used is accomplished using a SAS/CALC spreadsheet. The user can define a cell that is 'hot' and when the cursor is placed on this spot, be taken to a report generated from spreadsheet information or any other data.

For purposes of illustration, the VARIANCE spreadsheet is created with twelve rows and five columns. Eight of the rows will be for cosmetic purposes, and there will be one row each for four different types of claims; Major Medical, Dental, Vision, and HMO. There will be one column for identifying types of claims, one for the average number of days to process the claim, one for the number of days to process the claim, and two variance columns, one for the number of days, and one for the percentage difference. The spreadsheet looks like the following:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>ACTUAL</th>
<th>TARGET</th>
<th>VARIANCE</th>
<th>PCTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>maj_med</td>
<td>11.43</td>
<td>15.00</td>
<td>3.58</td>
<td>0.24</td>
</tr>
<tr>
<td>dental</td>
<td>14.77</td>
<td>15.00</td>
<td>0.23</td>
<td>0.02</td>
</tr>
<tr>
<td>vision</td>
<td>18.13</td>
<td>15.00</td>
<td>-3.13</td>
<td>-0.21</td>
</tr>
<tr>
<td>hmo</td>
<td>37.16</td>
<td>15.00</td>
<td>-2.16</td>
<td>-0.14</td>
</tr>
</tbody>
</table>

All of the cells in the TYPE column are hotspots. When the cursor is placed in the last cell of the TYPE column and the ENTER key is pressed, a PROC PARETO step runs that generates a report showing the reasons why HMO claims were delayed in their processing. Pressing END returns the user back to the spreadsheet.

The key to the hotspotting in the spreadsheet is in the program that drives it. A quick look at a part of the VARIANCE.PGM program explains how the hotspotting is done.

```sql
if (_cursorcol_=@type) and (_cursorpage_=1) then do:
    if (_cursorrow_=@hmo) then
        call display('hmo.program');
    /* other program code */
end:
```

In this case, placing the cursor in the last cell of the TYPE column actually invokes the HMO.PROGRAM. A look at the source window of HMO.PROGRAM explains how the PARETO charts were generated.

```sql
init;
return;
```
Next, go back to the SAS/EIS main menu, select the the Build EIS button and get back into your application database. Select the ADD button again. This time, when the Add window opens, select Spreadsheet Application... When the Spreadsheet Application window opens, supply a name and a description. Finally, give the four-level name of the spreadsheet: SASUSER,CALC,VARIANCE,CALC. Select the Test pushbutton to make sure everything works.

SCRIPT Objects

A SCRIPT object is a simple mechanism for combining several SAS/EIS objects together and having them execute in sequential order. The reasons for doing this are best illustrated with an example. Suppose you would like to select a Critical Success Factor (CSF) from your main menu whose needle displays a value from a spreadsheet. The spreadsheet is the target application for the Critical Success Factor. Ideally, the user can do some 'what if' analysis in the spreadsheet, and then when the spreadsheet closes, control will be returned to the Critical Success Factor and the needle will point to the new value reflected by the spreadsheet.

If the spreadsheet does not already exist, it needs to be created along with the program that drives it. This program uses a COMMAND statement that creates a new data set, or it will overwrite an existing one, for example:

```sas
_command_='create data=sasuser.dataset range=x noprompt replace; end';
```

This data set needs to be registered in the metabase with a variable that has the CSF attribute.

Next, a spreadsheet application needs to be added to the EIS. Supply the four-level name of the spreadsheet that will be invoked when this application is chosen.

Next, edit your main menu (a SAS/EIS DESKTOP application) and create a new region. Fill the region with a Critical Success Factor and name the spreadsheet application as the Target Application.

So far, so good. If you test the application now, starting with the main menu and choosing the Critical Success Factor, you will go into the spreadsheet as planned. Do a little 'what if' analysis, play around, and when finished, and out of the spreadsheet. Control is returned to the main menu just as expected, with one exception. The needle on the Critical Success Factor does not reflect the new value from the spreadsheet. If you close the main menu and then return to it, the new value is then displayed in the Critical Success Factor.

One solution to this problem is to use the SCRIPT object. This object allows the user to combine several objects and have them execute in sequential order. In this application, the SCRIPT object needs to 'string' together two other objects that are not yet created.

The first object that needs to be created is the SASCMD object, which executes a SAS command. From the Build EIS window, select the ADD pushbutton, then choose the SAS commands... application. Supply a name, like SASCALC, a description, and the following command to be executed:

```sas
calc libname.catalog.ssnname.calc open nodir
```

The second object that will be placed in the SCRIPT object is the one that calls the main menu. In this example, the application that calls the main menu is MAIN.DESKTOP.

Now, you are ready to create the SCRIPT object. From the Build EIS window, select the ADD pushbutton, then choose the Script of SAS/EIS applications... application. Supply a name, like SCRIPT1, a description, and then the names of the applications to execute as follows:

```sas
SASCALC SASCMD
MAIN.DESKTOP
```

From the Build EIS window, select the object that executes the main menu, in this case, MAIN.DESKTOP. Select Build, then go to the region that contains the CSF and activate it. Select the first application from the pull-down menu, then Object attributes, and then Target Application. Make the newly created SCRIPT object, SCRIPT1, the target application.

Go back to the Graphic Menu Builder window and select the Test pushbutton to make sure the application does what you want it to do.

Once you are familiar with the SCRIPT object, you can come up with many uses for it. For example, suppose you want to use a certain data set in an EIS object but need to preprocess it first. Perhaps you need to subset the data, and the type of object you're using doesn't have subset capability. Perhaps you want to use a summarized version of the data in the object, but don't want to store the data permanently in both summarized and unsummarized form. This is easily accomplished with a SCRIPT. The first application in the SCRIPT object is an EXECUTE object that runs a SAS program to subset, summarize, or otherwise preprocess your data. This step can create a temporary data set, and the second application in the SCRIPT object will be built to access the temporary, processed data set rather than the original one.

SAS/AF Applications

Perhaps there are some SAS/AF applications already written, and you want to organize them behind a main menu or simply plug one into your EIS. This is easily done using the AF object in SAS/EIS. In a later section this paper discusses more about incorporating various kinds of user written SAS/AF applications into your EIS. This first example illustrates the functionality of the AF object by linking to a component of SAS/ASSIST®. Since SAS/ASSIST is written in SAS/AF, it can provide you with prewritten SAS/AF modules to access from your EIS.

Suppose you want to generate a cross-tabulation report. You know that this can be done using SAS/ASSIST, but you don't know what module within SAS/ASSIST is used to do it. The first thing that needs to be done is to define a function key that will issue the ID command, which can be used to identify what
REPORT WRITING --> COUNTS --> Crosstabulation

When the Two-Way Crosstabulation window opens, press the ID function key. The four-level name of this window, SASHELP.ASSIST.REPFTWOPROGRAM, should appear. Leave SAS/ASSIST and go to the Add window of SAS/EIS. Select SAS/AF application ..., and when the SAS/AF Application window opens, as shown in Display 1, select the SAS/AF application pushbutton. Next, supply SASHELP.ASSIST.REPFTWOPROGRAM as the name of the application. Select the [Test] pushbutton to make sure it works.

Linking to Other Features

Though the default SAS/EIS objects provide extensive functionality, you can extend the capabilities of your EIS by providing access to any SAS software product or SAS/AF application in addition to SAS/ASSIST.

Using the AF object, you can also build into your EIS access to the

- ADX Menu System (SAS/QC® and SAS/STAT® software required)
- QC Menu System (SAS/QC software required)
- PROJMAN Menu System (SAS/OR® software required)
- SAS/LAB software
- SAS/INSIGHT software
- Time Series Forecasting System (SAS/ETS® required).

Using the SYSCMD object, you can also build into your EIS access to

- the native editor
- operating system utilities
- an electronic mailer
- a news service

Obviously, there are a number of ways to extend the capability of default SAS/EIS. However, if the user wants even more flexibility, there is always the option of using other application development tools provided with SAS software. The remainder of this paper addresses these issues.
Command (Menu) Methods

Processing of standard SAS/EIS commands such as Bookmark, Notes, and Setup, is handled internally by SAS/EIS. You can generate your own command menu that includes some or all of these EIS commands, and you will not need any additional processing to handle these commands. For example, suppose you would like a Business Graph without the PRINT option. In this case you can write PMENU procedure code to create a PMENU entry that will include only the items or commands you need. For example:

```sas
proc pmenu c=sugil9.windows;
  MENU bgrfmenu;
  ITEM 'View' selection=view;
  ITEM 'Print' selection=print;
  ITEM 'Bookmark' selection=bookmark;
  ITEM 'Goback' selection=goback;
  ITEM 'Help' selection=help;
  selection view 'VIEW';
  selection print 'PRINT';
  selection bookmark 'BOOKMARK';
  selection goback 'END';
  selection help 'HELP';
run;
quit;
```

After writing and submitting this code to create the PMENU entry, the next step is to point to this PMENU entry in the Methods window of the CHART application. There will already be a default PMENU entry defined, and you can either type over the entry name or use the control arrow to bring up a selection window and choose the appropriate entry name from a window as shown in Display 4.

Now, whenever you test or run this application, you will see only the command items you have defined, as shown in Display 5.

Initialization Methods

Both Initialization methods and Selection Action methods are SCL code modules that can be attached to a SAS/EIS application. The Initialization method is called only when the object first begins to execute, so it can be used to perform tasks that only need to be done once. Both types of methods receive information from the object in the form of an SCL list containing named items.

For example, suppose you have a series of financial reports and graphs in your application, and you would like to monitor which ones are used the most. You can write some SCL that would write to a logging file whenever a report or graph is selected for...
execution. Since we will obtain the name of the application from the method list, the code is completely generalized, and there is nothing hardcoded relating to this specific report or graph. The same method can be attached to any object that accepts methods, so the SCL program can be reused throughout your EIS. Here is sample code for an SCL entry to perform this task:

```scl
/* sample initialization method for logging */
%eis;
%eisentry methlist 8 rc 8;

init:
    appl=getnitemc(methlist, 'APPLNAME');
    /* on mainframe sysjobid=userid */
    user=symget('SYSJOBID');
    datetime=datetime();
    dsid=open('SUGI19.LOGFILE', 'U');
    call set (dsid);
    rc=append(dsid);
    dsid=close(dsid);
return;
```

You reference the name of this SCL entry in the Methods window for the objects where it is to be used, as shown in Display 6.

Display 6  Define Which SCL Entry To Use

Selection Action Methods

Selection Action methods are SCL code modules that will execute any time a user clicks anywhere on the display of the object. From the method list, you can obtain some information regarding what the user clicked on. For example, did the user click on a hotspot, and if so, which one? Or if they clicked on text, what was the text under the cursor? You can also find out if a command was issued, and if so, which one. If you have written PMENU code to generate custom commands, this is where you would process those commands.

Many objects have tasks that are performed automatically when the user clicks on the display, such as drill down or navigation in the Business Graphics and Variance Reports, and in many cases you can also branch to another application if the user clicks on a hotspot. Sometimes the task required is too complex or customized to implement with just those features. For example, suppose your users would like to be able to click on a row of the report in a General Report object (Display 7) and dynamically generate a graph from the data based on which row (which Product and Product Group) they selected (Display 8). This is a good application for the Selection Action method.
The program needs to detect whether the user clicked on a data row, and if so, what values of Product and Product Group were on that data row, and then generate the appropriate graph. A sample SCL entry program to do this is shown below. You point to the name of this entry in the General Report object Methods window.

/* sample selection method to produce a graph */
%eis;
%eisentry methlist 8 rc 8;

init:
  if getnicenc(methlist,'TYPE') = 'D' then
    text=left(getnicenc(methlist,'TEXT'));
  if text=' , then return;
  prodgrp=scan(TEXT,1,' ');
  product=scan(TEXT,2,' ');
  submit continue;
  %plotit(prodgrp=&prodgrp,product=&product);
endsubmit;
return;

ADDING NEW FEATURES AND APPLICATIONS

You have seen that by being knowledgeable about the capabilities of the existing EIS objects and what can be done with methods, you can customize your application without writing extensive new code. Occasionally you will have a need that is so specific to your company or your application that what you really need is a whole new application component.

SAS/AF is an extremely flexible applications development tool that can be used to write stand-alone interactive applications or to write applications for integration into your EIS. The sample displays shown in Displays 9 and 10 are examples of SAS/AF FRAME entry applications that can be integrated into an EIS.

Display 9 Sample Application with Report and Graphics

Display 10 Sample Application with Organizational Chart

There are a number of approaches you can take in developing your own solutions depending on your specific needs, such as the following:

- link a pre-existing SAS/AF application to your EIS with little or no modification
- write a new SAS/AF FRAME application to your customized specifications, and link it to your EIS
- create a new SAS/EIS object
- create a new SAS/AF class that you can add to SAS/EIS DESKTOP applications.

Linking SAS/AF Applications to an EIS

When adding either a pre-existing or a specially written SAS/AF application to your EIS, you will define it as an AF object. You saw an example of this earlier where the SAS/AF application referenced was actually part of SAS/ASSIST. Your AF application can be used as a target application to be invoked from a Graphics menu (DESKTOP object), a LISTMENU, a hotspot in a graph or report, or it could be one of the applications invoked as part of a script.

Adding an already written SAS/AF application to an EIS is commonly used in cases in which you have a significant investment in prior SAS/AF applications development and your users are already familiar with using the applications in question. In other cases, you may want to write a new SAS/AF application. This application can be written using FRAME entries, which will give it the same Graphical User Interface (GUI) look and feel as SAS/EIS, but will allow you to have a much more customized appearance and function.

Note that the Build window for the AF object has several choices for passing parameters: no parameters, SAS/ASSIST parameters, or SAS/EIS parameters. In other words, there is no place here to supply custom parameters that you may be used to supplying in your DISPLAY function calls in SCL. If your SAS/AF SCL program requires such values, you will need to supply them some other way. Some possibilities include:

- hardcoding the values in your SCL. This is obviously not the first choice, especially if you have gone to the trouble of generalizing this program and want to use it in several places with several different combinations of parameter values.
creating macro variables before invoking the AF application
(possibly by running variables as an EXECUTE object)
and having the SCL program obtain the values with the
SYMGET function.
- storing information in an SCL environment list (possibly by
running an SCL program as another AF object) and having the
SCL program obtain the values with SCL list functions.

Perhaps this generalized AF application always needs to be
called as the second item in a SCRIPT application, where the first
item is an EXECUTE or AF application that sets up the required
macro variables or SCL list items as described previously.

You can also add SAS/EIS command menu items such as
"Bookmark" to your own applications to give them even more of
an EIS look and feel. To do this, your AF FRAME or PROGRAM
entry will need to point to a PMENU entry to be used, and you will
need to add some code to process the special commands. For example, suppose you have written the PMENU Procedure code
to produce a command menu with the following items:

WhatIf Bookmark Go back Help

where WHATIF is a command specific to our application, and
BOOKMARK is the standard SAS/EIS Bookmark facility. In order to
process this command in your SCL program:
1. include calls to the standard macros %EIS and %EISENTRY
at the beginning of the program to set the SAS/EIS
environment and receive standard SAS/EIS parameters.
2. in the INIT section, include the following code to add your
application to the Bookmark list:
call display(%acatalog(eis,PSHSTCK),
   'name of your application', pos);
3. in the MAIN section, include the following code to process
commands:
   %select(pos, gbrc);
   /* add code for your own custom commands */
   when( 'WHATIF')...
   otherwise:
   end;
4. If you call any other programs that use Bookmark, then after
you call them, you need to add the following code to determine
if this application now needs to close down:
call display(%acatalog(eis, CHKSTCK), pos, gbrc);
   if gbrc=1 then
   do;
   link term;
   return;
   end;
5. if in the TERM section you check for an error condition that will
prevent the application from accepting an END or CANCEL
command (in other words you won't let the user go back), then
you will also need code to remove the Bookmark request when you
resume processing:
   if <your error condition> then
   do;
   /* other processing related to the error */
   if gbrc then
   call display(%acatalog(eis, RMVSTCK));
   _status_="R";
   return;
   end;
6. take your application off the Bookmark list when you go back;
call display(%acatalog(eis, POPSTCK));

This code makes use of SAS macros and routines that are
available as part of the product, so what you have seen in this
paper is all of the extra coding that is required on your part. It is
relatively simple, and helps make your SAS/AF application seem
even more a part of the rest of your EIS.

Creating New Objects or Classes

Both writing a SAS/EIS object and creating a SAS/AF class for
use in a DESKTOP application require a more in-depth
knowledge of SAS/AF and of how SAS/EIS works behind the
scenarios. These are good approaches in situations where you will
want to make repeated use of a specific technique or module of
code. This section discusses some of the pros and cons of the
several approaches you might be considering when developing
new SAS/AF applications for your EIS.

Writing your own SAS/EIS object is a multi-step process which
generally begins with writing a FRAME application that performs
the task you have in mind and looks and behaves in a way that will
please your users. If you have already written a FRAME
application to add to your EIS, then you are more than halfway
there. This FRAME entry will become the Run method for your
SAS/EIS object or the window that you see when you test or
execute the application. To convert the existing FRAME entry to a
Run method follow these basic steps:
1. Generalize the FRAME entry as much as possible so that
certain components (such as what data set to use, what
variables, and other options) are not hardcoded.
2. Add code so that these various parameters are obtained from
an SCL list. In an EIS object, the Run method obtains the
values of parameters or instance variables from a stored SCL
list that is created when the application or Instance of the
object is built.
3. Add any other code specific to SAS/EIS, such as code to
process SAS/EIS commands (as shown above) or code to
access the metabase.

In addition to the Run method you will need to write a Create or
Build method, which is the window the applications developer
sees when choosing to add or edit a SAS/EIS application. This can
be a FRAME entry or a PROGRAM entry, and it needs to
prompt the developer for values of all of the instance variables or
parameters that will be needed by the Run method. Code also
needs to be included to add the application to the SAS/EIS
applications database, which involves adding an observation to a
data set and creating a stored SCL list.

Although this may sound like a complicated process, once you
have done it one time, you will find that you use basically the
same techniques and the same or similar SCL code each time
you write a new object. This process, along with using SAS
macros to generate the EIS-specific SCL code that is needed
each time, is documented in a manual tentatively titled "A Guide
to Extending and Customizing SAS/EIS Software."

Suppose you have written a FRAME entry that you have added to
your EIS as an AF object as discussed previously. Why not make
it an EIS object? If the FRAME entry is very specific to a task,
and the same basic technique will only be used once or twice in
your EIS, then the effort it would take to generalize the Frame
entry and convert it to an object may not be worth the time it
takes. However, it is worthwhile to look critically at your FRAME
applications and ask with some changes, can this be reused over
and over in other contexts? Perhaps you should show your
application to other applications developers to see if it is similar to
something they need to write or if they think they could use it. If it
appears to be a candidate for reuse, then the effort required to
genitalize the code is worth the time it takes. Writing reusable,
genilized tools in SAS/AF is probably not a new concept to you,
so the question remains, why take the extra step to convert this
genilized program to an EIS object? Using the Create method
and Run method development scenario described previously
provides an efficient, structured way to obtain the values needed
for the parameters or instance variables and pass them at
evaluation time via the stored SCL list. Creating an EIS object also
allows you to turn some of your EIS development over to
applications developers who are familiar with using EIS objects as
building blocks, but who are not really familiar with using SAS/AF,
calling entries, passing parameters and so on.

The last approach to discuss is subclassing, or creating a SAS/AF
class that can be added to a SAS/EIS DESKTOP application. This
also is a good approach for situations where you have something
you want to use repeatedly in your EIS applications. For example,
suppose you like the concept of the CSF object to show the status
of a particular measurement, but you don't want that specific
graphic. Suppose also that you have a programmer who has
written code to generate graphics that display similar information
but in a different form (shown in Display 11).

The CONCLUSIONS

This paper has presented some of the standard features of
SAS/EIS software. By gaining a deeper understanding of the
flexibility of the existing objects, you can customize your EIS to
meet your needs. This paper also discussed various ways in
which SAS/AF can be used to create even more flexible and
customized extensions to your EIS.

REFERENCES

Additional information on these and other SAS/EIS related topics
can be found in:
User-Written Methods and Objects in SAS/EIS Software, Cary,
NC: SAS Institute Inc.
Inc.

ACKNOWLEDGMENTS

The authors would like to thank Jamie Robbins of SAS Institute
Inc., the staff of SAS Consulting Services Inc., and the staff of
SAS Institute Europe for some of the ideas presented in this
paper.

TRADEMARKS

SAS, SAS/AF, SAS/ASSIST, SAS/GRAPH, SAS/EDS, SAS/ETRS,
SAS/INSIGHT, SAS/LAB, SAS/OR, SAS/STAT, and SAS/STAT are
trademarks or registered trademarks of SAS Institute Inc. in the
USA and other countries.