SAS/EIS® Software: A Tutorial
Scott Isaacs, SAS Institute Inc., Cary, NC
Gail Kramer, SAS Institute Inc., Cary, NC

INTRODUCTION
This session is aimed at users interested in learning how SAS/EIS® software can be used to develop executive information systems (EIS). Applications will be developed that illustrate features of the objects provided with SAS/EIS software. Additionally, applications will be developed that require customization. FRAME entries in SAS/AF® software and Screen Control Language (SCL) programs will be introduced as tools for customizing and extending SAS/EIS applications.

INFORMATION DELIVERY USING SAS/EIS SOFTWARE
Developing strategies for information delivery is a key goal of many of today's organizations. It is becoming imperative that you organize all of your sources of data and seamlessly transform the data into information. Delivery of the information should be quick, efficient, and timely, in addition to being visually impressive.

When this is accomplished, you will have a system that enhances the ability of employees to make key decisions based on critical information. Providing this type of solution is the job of application development personnel in your organization; the centerpiece for quickly developing an executive information system using the SAS® System is SAS/EIS software.

THE DEVELOPMENT ENVIRONMENT IN SAS/EIS SOFTWARE
SAS/EIS software provides a robust collection of tools that increase the productivity of application developers. Importantly, you can rely on the powerful data processing, analytical, and report writing routines in the SAS System to drive many of your SAS/EIS applications. The SAS System has provided its users with all of this functionality for years. Now, the application developer can call on the rich development environment of SAS/EIS software to organize old, current, and new applications. The development process benefits from the object-oriented techniques inherent in SAS/EIS software. Also, the ability to fully expand the development environment's 'toolbox' will be of vital importance to most EIS developers. This topic will be discussed later as we explore some of the application development tools available in SAS/AF software and in SCL.

An intuitive development interface is presented to the application developer when SAS/EIS software is invoked. From a Display Manager command menu, select

![Display 2 SAS/EIS Main Menu](image)

Some of the choices on the menu include:

**Metabase**

The metabase system organizes your data so that you can specify how you intend for the data to be used by various applications. It is quite possible that one variable in a data set may be used differently in two separate SAS/EIS applications. The metabase system ensures that these usages are defined appropriately.

**Build EIS**

The Build EIS window displays the current application database that you are building. You can add applications that display reports, graphs, critical success indicators, and so on. As a developer, you may choose to use the predefined objects that are delivered with SAS/EIS software or use your own unique objects that address a specific task your users want implemented. Several applications will be built during the tutorial.

**Object Manager**

The Object Manager window displays the current object database setting. An object database is a storage location for custom, user-defined objects written by an application developer. A listing of custom objects appears in this window, and additional windows prompt the developer to specify methods (programs) that control the behavior of the object. An example will be illustrated later in the tutorial.

**Creating a Metalist and a Metabase**

As defined earlier, the metabase system manages and organizes your data. You register data sets and variables so that your applications use the correct types of variables and data for specific tasks. Hierarchically, the metabase system is designed as 'smart lists' of information, recording how you intend to use your data. The integral components include

**Metastats**

A list of metabases that are related. For
example, a department within your company might be assigned a dedicated metadata, which stores a list of metabases. The metabases would record what data sets this department uses and how it intends to use the data.

A list of SAS data sets (memtype DATA or VIEW) that are registered in order to supply data to applications. The metabase records descriptive information about each data set and about each variable within each data set. It also records what specifications the application developer makes for each variable. In other words, what the intended usage(s) of a variable are.

A list of attributes available to the application developer. These attributes are assigned to variables in registered data sets and define what the usage(s) of the variables are. You can create custom attributes in addition to the default attributes.

Suppose that you have a SAS data library that contains five data sets. You might have three primary divisions at your company, and each division may need to use all or some of the five data sets. Each division has its own projects, tasks, and usages of the data. Therefore, hierarchically, you might set up a metabase system as follows:

To create a metadata from the SAS/EIS Main Menu, select

**Metadata → Metadata → Add → Library**

Select a libref from the selection list, type in the five-character name of your metadata and a description, and then select **OK**.

Next, to create a metadata return to the Metadata System menu. First, make sure that your new metadata is active by selecting

**Metabases → Metadata → Select a metadata → your new metadata → OK**

Now, from the SAS/EIS Metabases window, select

**Add → Library**

Select a libref from the selection list, type in the five-character name

Finally, you need to assign attributes to variables contained in the data sets. This can be done at this point or later. It might be that, as you later develop applications, you will realize the need to assign specific attributes to variables. To assign attributes to variables, edit a registered data set by either typing an "e" next to a data set name or using the popmenu. When the popmenu appears, select

**Registered variables**

When a list of variables appears, you can edit the variables’ assigned attributes. This action should produce a list of attributes that can be assigned to the variable. If the variable is character, you can choose the following extended attributes by default:

- CLASS
- DRILL

If the variable is numeric, you can choose the following extended attributes by default:

- ACTUAL
- ANALYSIS
- BUDGET
- CLASS
- CSF
- DATE
- DATETIME
- DISCRETE
- DRILL
- FORECAST
- TIME
- X
- Y

A thorough understanding of the application types that you can create using SAS/EIS software and what registered attributes are required for each application will aid you in determining how to register your variables. Refer to *SAS/EIS® Software: Reference, Version 6, First Edition* for more information on metadatas, metabases, and attributes.

**CREATING APPLICATIONS USING OBJECTS**

A comprehensive set of objects is supplied with SAS/EIS software. You can use these objects to create applications for your
information delivery system. An object is a collection or package of methods, and a method is simply a program or routine. A given method would control some feature or task of the object. An object's methods work in conjunction to provide some type of functionality when you supply data or information to the object. This is referred to as "instantiating" the object, which means that you have created an application using the object.

Figure 2 Creating Applications Using Objects

You can use any of the 27 objects supplied with SAS/EIS software to create your applications. You can also write methods that work in conjunction with an existing SAS/EIS object that meets most of your task's needs, thus you are extending and customizing the object by overriding some default feature. Another choice is to call existing SAS or SAS/AF applications to use as part of your EIS. Lastly, you could consider writing your own object entirely from scratch using SAS/AF software and SCL to provide a certain functionality based on your users' needs. This requires writing your own methods, designing your own development and execution screens, and debugging your own object. You would then use SAS/EIS to create applications using your custom object as many times as needed.

The point is that you have great flexibility as an application developer because you have the power of SAS/EIS software and the power of the entire SAS System surrounding it. The object-oriented approach provides you the ability to re-use these packaged routines over and over just by simply supplying data or information to them to create applications. This idea allows us to explore the difference in objects that require data versus those that don't.

Creating an Application Database

You can create an application database by making the following selections from the SAS/EIS Main Menu:

**Build EIS → Application Database → Create a database →**

Libref

Select a libref and then provide a name. Return to the Build EIS window by selecting **OK**.

You are now ready to add an application to your application database. To see a list of the predefined application types that you can use to develop applications, select **Add**.

Display 3 Default SAS/EIS Add Window

You would next select an application type that you wish to build. The following examples will illustrate graphic-oriented tasks that require using several different objects to create specific applications.

Viewing Dynamic Results

The first example is a business graph application. When you select **Business graphs...**

in the SAS/EIS Add window, you will see a popmenu that prompts you for the type of business graph.

Display 4 Defining a CHART Application

A CHART application will be built for this example. Once you have made this selection, the SAS/EIS Charts window will be displayed. This is referred to as the **build method** of the CHART object.
of objects that require data to build applications. One is dynamic. After you have provided the required data to the object's build method, you have created an application. There are no categories run-time data. The other is pre-processed data from which a stored report or graph was produced.

The requirement for a CHART application is that you register, at a minimum, one variable of a data set with the attribute DRILL in your metabase. Additionally, you could register other categorical variables with the DRILL attribute, which builds what is referred to as a drill hierarchy. Thus, the concept of drill down would be implemented. Also, you may consider registering other variables in the same data set with the attribute ANALYSIS. This applies to numeric data and registers the variable as a candidate for statistical computations.

You may have correctly registered your data earlier in the development phase or you may need to return to the metabase system to update your registrations. Once you have met the requirements, you may test your new application.

The results of the CHART application may be perfectly adequate for your users. Or, you may decide that you have existing SAS/GRAPH® programs that produce custom output using many features of the software. The next section addresses this situation. Remember, though, that the CHART application in conjunction with the metabase system and the attribute dictionary provides dynamic reporting.

Viewing Static, Customized Results

If you want to develop an application that views static output or results from your favorite tried, true, and tested programs, then one simple solution is to learn how to implement the SCRIPT object. This object falls into the ‘no data required’ category because all that you have to do is sequence the execution of other existing SAS/EIS applications. This presumes, of course, that you have already developed and tested the applications that you will include in the SCRIPT application.

You can create a SCRIPT application by selecting in the Add window. Next, simply sequence the application in the order that they should execute. You can type a '?' in the fields to display a selection list containing entries for your current application database. For example, you may want to script an EXECUTE application with a GRAPH application that follows in order to execute one of your SAS/GRAPH programs and then display the stored graph. This type of example is similar to something that we will focus on in a later section when selection methods are discussed.

Calling a SAS/AF Application

Now, let's assume that you have some end-users who would benefit from or are already accustomed to the High Resolution Graphics Menu in SAS/ASSIST® software. It is a very simple task to develop an application that fires up a SAS/AF application, which is all that the SAS/ASSIST screens really are. Therefore, once you determine which menu screen to call, you can easily create an application using the AF object in SAS/EIS software that calls a SAS/ASSIST screen.

For example, you may want to provide users the capability to branch to a screen that will build simple vertical bar charts. This can easily be done once you determine the location of the program entry for this type of application resides in the SASHELP library. In this example, we will use the entry named GCART1 in the SASHELP.APPLICATION catalog. In order to build an AF application using the AF object, make the following selections from the SAS/EIS Main Menu:

| Build EIS | Add | SAS/AF application ...
|

Next, provide a name and description for the application and then designate the target SAS/AF application by selecting SAS/AF application:

in the SAS/AF Application window. Type or select the four-level entry name and then return to the previous window. You can test the application at this point.
Using Your Methods to Modify Object Behavior

If you determine that you need to perform tasks which require functionality that is not provided directly by the pre-defined SAS/EIS objects, then you can create your own programs that will augment the pre-defined objects. These programs would be SCL programs that you would use to implement some task or tasks in addition to the default tasks that an object performs. Of course, this type of customization would require sound knowledge of SAS/AF software and SCL.

Suppose that you want to display a map for users and allow selection of graphic elements on the map. These graphic elements might be symbols representing different regions or cities. Once a user has selected a symbol, the target application would display a bar chart or pie chart for the region or city that was selected. Now, thinking about what we have discussed so far, it would seem logical that you could provide a target CHART application for each region or city on the map. Actually, this would be a viable solution in many situations. On the other hand, let’s suppose that the map displays 50 or 100 different regions or cities. In this case, you would be facing a lot of repetitious work to build all of the individual target applications. Since you want to remain within the framework of SAS/EIS software and use a pre-defined object to display the map, one solution to the problem is to write an SCL program that modifies the default behavior of the selection method provided with the DESKTOP object. Thus, you could get by with only one application that does the work to create a dynamic, user-specified graph. A selection method is a program that controls object behavior when a user is executing or running an application. The DESKTOP object enables you to display stored graphs in a SAS/AF frame entry, which provides hot-spotting functionality for symbols on the graph. A DESKTOP application can be built by selecting

Graphics menu builder...

in the Add window. Keep in mind that many other objects in SAS/EIS software provide the capability to specify your own selection methods; we have chosen the DESKTOP object to illustrate this idea.

The SCL program needs to be developed using the BUILD procedure of SAS/AF software. When completed, it might look like the following partial program:

This program will determine what hotspot the user selected and then generate a graph for the respective city, subsetting the data based on the city corresponding to the hotspot. The graph will always be stored in the same temporary catalog entry so that the program can manage the contents of the catalog. Finally, the graph will be referenced in a GRAPH application stored in the EIS application database. A GRAPH application simply displays a previously stored, static graph. Only one GRAPH application needs to be created since all hotspots (cities in this case) can call the same target application. Thus, when the DESKTOP application executes, after the existing graph in the catalog is deleted a new graph will be created each time a hotspot is selected. The new graph will always be stored by the same name, which the GRAPH application references.

Once you have compiled and debugged the SCL program, you can reference the program in the DESKTOP application. The DESKTOP application needs to be designed so that the master region is filled with the SAS/GRAPH map picture. Any area that should be a hotspot needs to be defined using the same GRAPH application as a target (assuming the GRAPH application was already created). After this is done, you can return to the Graphics Menu Builder window and select Method: Selection action: to specify the four-level name that contains your SCL program. Now, you can test your application.
Writing an Object to Handle Your Tasks

One other alternative for the developer is to write an entire object from scratch using SAS/AF software, SAS/AF frame entries, and SCL programs. This choice again would require an extensive comprehension of the aforementioned tools and would be considered an advanced task. For more information on creating SAS/AF frame entries, refer to SAS/AF Software: Frame Entry Usage and Reference, Version 6, First Edition. The flexibility provided by SAS/AF software and frame entries gives you the opportunity to develop custom objects that perform the tasks that you want. You can design these objects with the same 'look and feel' as the pre-defined objects in SAS/EIS software, or you can make them look entirely different. You will need to write all of the associated methods (SCL programs) for the object and design what the object will look like to both the developer and the end-user. Once you have tested and debugged the object, you can include it in the Add window selection list that is displayed when a developer adds a new application to their application database. It will appear in its respective alphabetical location in the list.

This talk will use, as an example, an object that provides the capability to scroll through stored graphs using up and down arrows. The graph names are actually stored as values of a variable in a SAS data set. Thus, a type of 'briefing book' could be created, using dynamically updated SAS data sets that contain information needed by a particular user or group of users. The object could control what information is available by reading or updating specific data sets that track the most current reports or graphs. Once all of the tools are developed, the tools can be combined to form a stand-alone object. Or, the tools could be defined as sub-classes, which would allow a developer to fill regions of a frame with each type of tool. The finished product, as a stand-alone object, is displayed below.

In order to add the object to the default list of objects provided with SAS/EIS software, you need to use several features in the Object Manager window. This window can be invoked by selecting Object Manager on the SAS/EIS Main Menu. For information about using the Object Manager refer to SAS/EIS Software: Reference, Version 6, First Edition. You also need to update the catalog search path using the Setup window.

Once you have accomplished adding your object to the Add window list, you can 'instantiate' the object and build applications just as you have done with the pre-defined objects that come with SAS/EIS software.

ORGANIZING YOUR APPLICATIONS - THE DESKTOP OBJECT

The final goal in this talk is to pull together all of the different pieces that have been discussed earlier. In most executive information systems, you want to provide the end-user with a powerful, friendly, and attractive graphical menu so that the user can make intuitive selections. Thus, you need to organize all of the individual applications and decide how to present them as choices on a menu or several menus.

The DESKTOP object is used to create the menu for this example. Many of the same features of the DESKTOP object that were used earlier can be used to design a graphical menu. Simply, it comes down to deciding which stored graphs you would like to display in different subregions on a frame entry. Once you have made these decisions, you can specify what the target application is for each choice. A sample DESKTOP application for a main menu was displayed earlier.


SUMMARY

This tutorial has demonstrated how easily simple, intermediate, and advanced applications can be developed using SAS/EIS software. You can use the pre-defined objects that come with the software or customize your development environment by designing your own tools to enhance the software's already powerful capabilities.