

Customizing EIS Objects: Approaches to Solving the Most Common Requests

Yvonne Selby and Ivy G. Parker



Yvonne Selby is a senior technical support analyst at SAS Institute. Her areas of expertise include SAS/AF® and SAS/EIS® software, FRAME technology, and Screen Control Language. Yvonne has a BA degree in computer science from East Carolina University and has been a SAS® user for nine years.



Ivy G. Parker is a technical support analyst at SAS Institute. Ivy has a BS degree in computer science from North Carolina State University and has been a SAS user for five years. Her areas of expertise are SAS/AF software, FRAME technology, and SAS/EIS, SAS/CALC®, and SAS/TRADER® software.

Abstract

SAS/EIS software offers many objects for creating your Executive Information System. Many of the objects provide you with simple dynamic tools for generating your business graphs, business reports, and menus. However, you may want to make simple customizations to some of the objects or to your EIS in general. This article answers some of the most common questions regarding customizing standard EIS objects and the application catalog. Approaches to solving the problems vary between objects and also between releases.

Introduction

Standard EIS objects that provide a Methods button or an Advanced button from the initial window of the build task allow you to override basic functionality of the object in question. Whether the object has a Methods button or an Advanced button determines the approach you need to take when customizing the default behavior of the object.

Objects with a Methods button have a Display initialization, Selection action, and a Command menu method. The Display initialization method executes before the object displays. The Selection action method executes when you make a selection

within the displayed object. These methods are available only in objects that are available in Release 6.10 of the SAS System and prior. For more information regarding these two methods, refer to "INITIALIZE and SELECT Methods" in Chapter 3, "SAS/EIS Software Method Templates and Macros," in *Extending SAS/EIS Software Capabilities, Version 6, First Edition*.

Objects added in Release 6.11 that have an Advanced button provide you with methods that can be overridden or executed before or after the default method, such as `_FILL_`, `_OBJECT_LABEL_`, `_POSTINIT_`, `_TERM_`, `_DRILLDOWN_`, `_EXEC_CMD_`, `_GET_ACTIONS_`, `_NAVIGATE_`, `_ROTATE_`, and `_SELECT_`. The method you need to override varies depending upon the object. For more information regarding these methods, refer to *SAS/EIS Software: Reference, Version 6, Second Edition*.

This article provides approaches for

- altering titles or displaying information in the title bars
- invoking an EIS object from a SAS/AF application
- defining and querying commands in an object
- subsetting data prior to executing or displaying an object
- displaying custom information within the Desktop object
- defining hotspots to allow subsetted data to be presented
- determining if entries can be deleted from the application catalog.

For implementing any of these examples, knowledge of SAS/AF software (including FRAME entries), Screen Control Language, and SAS/EIS software is assumed.

Altering Titles or Displaying Descriptions

Any SAS/EIS object that has a Methods button or an Advanced button can either alter the title or display descriptive information in the title bar of the window.

For objects available prior to Release 6.11, all approaches involve writing your own Display initialization method. For objects available beginning with Release 6.11, a different approach must be used. Note that in all cases you can use macro variable substitution as well as query information from the data set to display in the title.

Altering Titles for Objects Prior to Release 6.11

To display information on the title bar of the window, edit your own SCL entry and use the EXEC CMD routine to issue either the SETWNAME or SETWDATA commands. For example, the following code displays the current date in the title bar. Place this

continued on next page

code in an SCL entry. Select **Methods** and specify this SCL entry as the Display initialization method:

```
%eis; %eisentry applst erc 8;
INIT:
  date=put(today(), worddate18.);
  call execcmd("setwname 'Business Report for '||
    left(date)||'");
return;
```

Beginning with Release 6.10, information about the displayed objects is passed via an SCL list to the overriding methods. Many of the objects available prior to Release 6.11, such as the Business graphs (Chart or Plot) objects, allow you to override TITLE1. To accomplish this, you could code the following in your SCL entry for the Display initialization method:

```
%eis; %eisentry applst erc 8;
INIT:
  erc=1;
  date=put(today(), worddate18.);
  applst=setntemc(applst, 'Business Report for '||
    left(date), 'TITLE1', 1);
return;
```

For more information about the %EIS and %EISENTRY statements, refer to *Extending SAS/EIS Software Capabilities*.

Altering Titles for Objects Available with Release 6.11

For objects available beginning with Release 6.11, the Display initialization method is not used. Depending on the object, you may need to override various methods such as `_FILL_`, `_DRILLDOWN_`, or `_NAVIGATE_`. Not all objects implement titles in the same manner. In the following examples, titles are altered for a 3D Business Graphs object and a Multidimensional report object.

Altering Titles for 3D Business Graphs

For the 3D Business Graphs object, you can write your own `_FILL_` method to execute before the default `_FILL_` method of the object. To do this, edit the object and select **Advanced** followed by **Methods** while editing the object. Then select `_FILL_` and specify the entry name and the label to execute for your own `_FILL_` method. Select **Before** for **When to Run**. This example overrides the `_FILL_` method for the 3D Business Graphs (Graphs) object and sets a title for it. The entry specified is `SASUSER.OVERRIDE.G3DTITLE.SCL`, and the label is `FILL`. Edit the SCL entry and enter the following code:

```
FILL:
  method applst erc 8;
  date=put(today(), worddate18.);
  call send(-sel f-, '-SET-TITLE-', 1,
    'Business Report for '||
    left(date));
endmethod;
```

You can also use the `EXECCMD` routine to issue the `SETWNAME` command. You can add additional programming to retrieve information from data sets, external files, macro variables, or SCL lists for display in the text of the title.

Altering Titles for the Multidimensional Report

The Multidimensional Report (Multi) object does not automatically display the selected drill values as you drill down. This is not a problem if you have customized the object to select `Expand for Double Mouse Click` under **Customize**. However, if you have selected `Drill down for Double Mouse Click`, then you need to override the `_NAVIGATE_` and `_DRILLDOWN_` methods in order to display the selected drill values in the title. When activated, the title can display two title lines. To activate the title, specify a default title in the `Report Title` box under **Customize**.

Now you need to write your `_DRILLDOWN_` method. To do this, edit the object, and select **Advanced** followed by **Methods**. Then select `_DRILLDOWN_`, and specify the entry name and the label to execute for your own `_DRILLDOWN_` method. Select **After** for **When to Run**. The following example executes the specified `DRILL` method after the default `_DRILLDOWN_` method. The entry specified is `SASUSER.OVERRIDE.DRILL.SCL`, and the label is `DRILL`. Edit the SCL entry, enter the following code, compile, and `END` back to the **Methods** window:

```
length title $ 200;
drill:
  method;
  frameid=getntem(-sel f-, 'frameid');
  call send(frameid, '-get-widget-', 'obj1', objectid);
  titleid=getntem(objectid, 'title');
  call send(titleid, '-is-hidden-', hidden);
  if hidden then return;
  drills=getntem(-sel f-, 'dimsubset');
  do i=1 to listlen(drills);
    name=nameitem(drills, i);
    list=getitem(drills, i);
    val=getitem(list, 1);
    title=title||' '||name||'='||val;
  end;
  call send(titleid, '-set-line-', title, 2);
endmethod;
```

The `DRILL` method determines the object identifier of the title object and stores it in `TITLEID`. It loops through the `DRILLS` subset list and builds the title. The `SEND` routine updates line two of `TITLEID` by specifying the new value of `TITLE` in the `_SET_LINE_` method.

Now you need to write your own `_NAVIGATE_` method. The `_NAVIGATE_` method executes whenever you select a direction to drill such as `UP` or `DOWN` in your `MULTI` object. To do this, select `_NAVIGATE_` from the **Methods** box. Select **After** for **When to Run**. This example executes the specified

NAVIGATE method after the default `_NAVIGATE_` method. The entry specified is `SASUSER.OVERRIDE.NAVIGATE.SCL`, and the label is `NAVIGATE`. Edit the `SCL` entry, enter the following code, and compile:

```
length title $ 200 name $ 8;
navigate:
  method direction $ optional =table-name
    active-var next-var hierarchy $
    drills-path 8 mb $;
  frameid=getni temn(-sel f-, 'frameid');
  call send(frameid, '-get-widget-', 'obj1', objectid);
  titleid=getni temn(objectid, 'title');
  call send(titleid, '-is-hidden-', hidden);
  if hidden then return;
  activevar=getni temc(-sel f-, 'activevar');
  drills=getni teml (-sel f-, 'drillsubset');
stopit:
do;
  title='';
  do i=1 to listlen(drills) until (name=activevar);
    name=namei tem(drills, i);
    if name=activevar then leave stopit;
    list=geti teml (drills, i);
    val=geti temc(list, 1);
    title=title||' '||name||'='||val;
  end;
  call send(titleid, '-set-line-', title, 2);
end;
endmethod;
```

The `NAVIGATE` method again determines the identifier of the title object and stores it in `TITLEID`. The current drill level is returned to `ACTIVEVAR`. It loops through the `DRILLS` subset to build the `TITLE`. If the `NAME` in the `DRILLS` subset is the same as `ACTIVEVAR`, then you don't want that information placed in the title. After the title is built, the `SEND` routine updates line two of `TITLEID` by specifying the new value of `TITLE` in the `_SET_LINE_` method.

Invoking an EIS Object from within Your Own AF Application

It is very simple to invoke an EIS object from within your own AF application if your AF application is invoked as a `SAS/AF` Application object from within your EIS application. By creating a `SAS/AF` Application object to point to your AF application and selecting to `USE DEFAULT EIS PARAMETERS`, most of the work is done for you. All you have to code in your AF application is the following macro invocations. These can be placed prior to the `INIT` section:

```
%EIS; %EISENTRY;
```

Now, to invoke the EIS object, place the following `CALL DISPLAY` routine in your `SCL` code. The second parameter specifies

the four-level name of your EIS object. In this example, `SASUSER.OVERRIDE.G3DCHRT.GRAPHS` is invoked.

```
call display('sasuser.eis.runeis.program', &eparmlst,
  'sasuser.override.g3dchrt.graphs', rc);
```

The `RUNEIS` program invokes the specified object. The `EPARMLST` macro variable resolves to a list of EIS parameters that the object needs in order to execute.

Defining and Querying Commands

It is possible to write your own menus for use with some of the objects. To do this, you must know the commands issued for the existing `pmenus` and duplicate those items in your own `pmenu` code. You must also know how to query the commands being executed in order to implement them in your object.

For example, the `Graphics Menu Builder (Desktop)` object's menu definition consists of the following commands: `HOTSPOT`, `NOTES`, `BOOKMARK`, `GOBACK`, `HELP`. If you want to implement these commands along with your own, you must use the `PMENU` procedure to first define the menu. If the item for your menu executes the same command as the item name, then no `SELECTION` option or statement is necessary. If the item executes a command different than the item description, then use a `SELECTION` option and statement to define the command. This example adds the `REPORT` item to the menu. In this case, the command issued is `REPORT` so no `SELECTION` option or statement is needed.

```
proc pmenu c=sasuser.override;
  menu mymenu;
  item HOTSPOT;
  item NOTES;
  item BOOKMARK;
  item REPORT;
  item GOBACK;
  item HELP;
run;
```

Edit your `Desktop` object and select [Methods](#) followed by `Command` menu to specify your menu.

Now you must write a `Selection` action method for the `Desktop` object that will query the `REPORT` command and execute code conditionally. In this case, the `REPORT` command could open an `FSVIEW` window or it could invoke another EIS object using the same `DISPLAY` routine as shown previously. The following `SCL` code could be placed in `SASUSER.OVERRIDE.REPORT.SCL` and then specified as the `Selection` action under [Methods](#) in your `DESKTOP` object:

```
%eis; %eisenry applist erc 8;
INIT:
  erc=1;
  cmd=getni temc(applst, 'COMMAND');
  if upcase(cmd)='REPORT' then
    call fsvew('sasuser.class');
return;
```

continued on next page

Here an FSVIEW window opens to display the SASUSER.CLASS data set.

For objects available beginning with Release 6.11, the Selection action method is not used. Instead, you can write your own `_EXEC_CMD_` method to query your command and execute code based upon the command. For example, you may want your 3D Business Graphs (Graphs) object to query the REPORT command and invoke an External File Viewer object called EXTVIEW. To do this, select [Advanced](#) followed by [Methods](#) while editing the object. Then select `_EXEC_CMD_` and specify the entry name and the label to execute for your own `_EXEC_CMD_` method. The entry specified is SASUSER.OVERRIDE.G3DREPT.SCL, and the label is GETCMD. Edit the SCL entry and enter the following code:

```
%eis;
length kname $ 17 kbdsid 8 applds $ 17 appldsid 8
      text1 text2 $ 200 num1 num2 8;
GETCMD:
  method command $ erc 8 ;
  if upcase(command)=' REPORT' then
    call display(' sashelp.eis.runeis.program', &parmlst,
                ' sasuser.override.extview.fileviewer', rc);
  endmethod;
```

Make sure you specify SASUSER.OVERRIDE.MYMENU.PMENU for Menu under [Advanced](#).

Now when the REPORT command is selected, it will invoke the External File Viewer object. The %EIS and LENGTH statements are only necessary when invoking other EIS objects.

Subsetting Data Prior to Executing the Object

If you have a master data set that you want to subset but you do not want to have to create a separate object for each subset, then you could write a Display initialization method for objects available prior to Release 6.11 that would prompt for the subset criteria, subset the data set, and execute the object. For objects available with Release 6.11, you must use a different approach.

For example, you may want to display a separate Business graphs (Chart or Plot) object or each house style in the SASUSER.HOUSES data set. Instead of creating a separate Chart object for each style and hardcoding a subset, you can write an AF application to accomplish this. Here are the steps.

First, register a temporary data set in the metabase. This can be a WORK data set and it should contain the same variables as the master data set. For example, create WORK.HOUSES from SASUSER.HOUSES as follows and register WORK.HOUSES in the metabase.

```
data work.houses;
  set sasuser.houses (obs=0);
run;
```

Second, you need to write your own AF application that will prompt the end users for the style of home. Edit SASUSER.OVERRIDE.SUBSET.FRAME, and place the following objects in the display window:

Object	Name	Attributes
PushButton	Select	Label: Select Style
Text Entry	Style	Type: Char, Protect: YES
PushButton	OK	Label: Continue

Edit the SCL source and enter the following:

```
%eis: %eisentry applst erc 8;
INIT:
  dsid=open(' sasuser.houses');
  stylelist=makelist();
  nlevel=0;
  rc=level(dsid, 'style', nlevel, stylelist);
  rc=sortlist(stylelist, 'ascending');
  dsid=close(dsid);
return;
SELECT:
  item=popmenu(stylelist);
  if item then style=getitemc(stylelist, item);
return;
OK:
  if style ne _blank_ then do;
    submit continue;
      data work.houses;
        set sasuser.houses;
        where style=&style";
      run;
    endsubmit;
  call execcmd('end');
end;
else _msg_=' Please select a style.';
return;
```

After you have compiled and tested your FRAME application, you can specify it for Display initialization under [Methods](#) in the Chart object. Make sure the data set specified for the object is the temporary data set that you registered.

Now when you invoke your Business graphs Chart object, the FRAME application displays first and prompts you to select a style. After you click on [CONTINUE](#), WORK.HOUSES is created as a subset of SASUSER.HOUSES. The Business Chart displays using the subsetted data set. You should also be aware that the LVARLEVEL function creates an SCL list containing the formatted values of the variable. Some additional coding may be necessary if your variable is formatted.

Again, for objects available beginning with Release 6.11, the Display initialization method is not used, and therefore you do not need the %EIS and %EISENTRY statement in your SUBSET application. Instead, you can execute your own `_FILL_` method before the object displays. To do this, select [Advanced](#) followed by [Methods](#) while editing the object. Then select `_FILL_` and specify the entry name and the label to execute for your own `_FILL_` method. For When to Run, select Before. This example overrides the `_FILL_` method for the 3D Business Graphs (Graphs) object and invokes the AF application to subset the data set. The entry specified is

SASUSER.OVERRIDE.G3DSUBST.SCL, and the label is FILL. Edit the SCL entry, and enter the following code:

```
FILL:
  method appl1st rc 8;
  call display('sasuser.override.subset.frame');
endmethod;
```

When you invoke your Graphs object, this FRAME application executes and prompts you for subset criteria. The WORK.HOUSES data set is generated, and the 3D chart displays.

Displaying Custom Information in a Desktop Object

The Display initialization method is not the only way to initialize an EIS Desktop object with custom information. By using SAS/AF, you can subclass Desktop objects that you want customized and add them to the MAKE selection list when you are in build mode within the Graphics Menu Builder.

For example, suppose you would like to create a graphics text object that you could select from the MAKE list that would automatically display the current date. To do this, you would need to invoke the BUILD procedure and edit a CLASS entry. In Description, enter whatever text you would like to see displayed in the MAKE selection list. Enter SASHELP.FSP.GTEXT for the Parent class. The Additional Attributes will then be ungrayed, and you can begin defining customizations to this object.

Under Additional Attributes, select Edit attributes. . . to display the Graphic Text Attributes window, fill in the following information, and select :

```
Name: TODAY
Text: <set to blank>
Color: <choose a color>
Font: <choose desired font>
```

Next, choose Set custom attributes if you would like to disable the radio button for Display attributes upon fill/make. If you disable this, then you can select this object from the MAKE list and position it without displaying the attribute window. Select to return.

You also need to specify where the code to carry out your date initialization is located and which label to execute, so select Methods to open the Methods window. This will allow you to specify the SCL entry to run to initialize this object with today's date. Scroll through the list of methods and select _INIT_, specify SASUSER.OVERRIDE.TODAY.SCL as the entry to execute, and specify INIT as the label. Select twice to exit the Methods window and the CLASS entry.

Now you can edit the SCL entry you specified for the _INIT_ method and enter and compile the following SCL code. The code sets the automatic instance variable _VALUE_ to the current date.

```
INIT:
  call super(-self-, '_INIT-');
  _value_=trim(putn(today(), 'weekdate. '));
return;
```

The next step is to copy SASHELP.EIS.GRAPH.RESOURCE to your catalog and add this new object to the resource list. This is the object resource used by the MAKE selection list under the Desktop object. After you have copied the resource entry, edit it and choose to add your new class entry to the resource list. Select to exit the RESOURCE entry.

The last step is to invoke EIS and choose Setup from the EIS main menu. Under Setup, select Resource management, then to select your modified GRAPH.RESOURCE entry. This will let EIS know where your modified resource entry resides.

You should now be able to go into Build EIS, edit a Desktop object, and see this custom graphics text object listed at the bottom of your MAKE selection list. When you test the Desktop object, today's date will be displayed.

Using Hotspots to Allow Subsets of Data to Be Presented

You can define either segment or overlay hotspots within EIS that enable you to invoke a target application that has been specified within the hotspot attributes. You can also override the Selection action method to execute your own SCL code when a hotspot is selected.

For example, suppose you created a Desktop object that contains a SAS/GRAPH® Output object to display a map of the United States. Suppose also that you also have defined segment hotspots for several states to allow you to click on a state to display some sales data. In this case, the hotspot names correspond to the two letter postal code for each state.

After the Desktop object is designed, you can choose within the Desktop object to specify the Selection action method to process each selected hotspot. The following code could be placed in SASUSER.OVERRIDE.USHOTSP.T.SCL to process the selected hotspot:

```
%eis;
%eientry appl1st rc 8;

length state $2;

MAIN:
  state=getntemp(appl1st, 'HOTSPOT');
  if state in ('WA' 'CA' 'TX' 'FL' 'NC' 'NY') then
    call fsvi ew("sasuser.sales(wher=(state="||
      quote(state)||")");
return;
```

Now, when a hotspot is selected for one of the defined states, the data are displayed for that state.

Deleting Entries in the Application Catalog

As you add entries to your application database, entries are added to the corresponding catalog. For instance, many objects will create a corresponding entry of type EIS in the application catalog. Graphics Menu Builder objects will create FRAME entries in the

continued on next page

application catalog. No entry should be deleted from the catalog randomly. If you want to delete an object, do so from the Build EIS window.

Program Availability

The SCL code (`custmei s. scl`) is available through Anonymous FTP. To use this facility, connect to `ftp.sas.com`. Once connected, enter the following responses as prompted:

```
Name (ftp.sas.com:userid): anonymous
Password: your e-mail address
```

You then enter

```
cd/observati ons/1q96/sel by
get custmei s. scl
```

Note that after you enter your password, all statements are case sensitive. If you want to rename a file before downloading, you can enter a target filename after the filename specified. To disconnect from Anonymous FTP, enter `quit`.

If you do not have a direct Internet connection, this file is available from SAS Institute's Bulletin Board System (SIBBS), which you can access via modem at (919) 677-8155. To use the SIBBS service you must establish an account with SAS Institute. Once you have registered, from the SIBBS main menu select (D) for

download files. From the Download Files Area select (F) for *Observations*[®] article source code. After viewing the list of journal files select (D) to download. When prompted, enter the filename, `custmei s. scl`. After downloading is complete, press return to exit, then enter `!` to logoff.

Summary

As you can see, it is possible to customize the basic functionality of many EIS objects simply by overriding methods, using command menus, and creating subclasses. These approaches are simple and should provide you with a basis for making more advanced customizations. For more information regarding methods and subclassing, refer to *SAS/AF Software: FRAME Application Development Concepts, Version 6, First Edition* and *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition*. Also refer to *SAS/EIS Software: Reference, Version 6, Second Edition* and *Extending SAS/EIS Software Capabilities, Version 6, First Edition*.

SAS, SAS/AF, SAS/CALC, SAS/EIS, SAS/GRAPH and SAS/TRADER are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.