Helping the HELPLINE - SAS/AF® to the Rescue
A Step by Step Guide

Verna C. Brantley, Blue Cross and Blue Shield of South Carolina, Columbia, South Carolina
Helen L. Lintern, Blue Cross and Blue Shield of South Carolina, Columbia, South Carolina

Abstract

This paper focuses on the how to's for developing an online information system for querying data using SAS/AF® software (frame entries), SAS/FSP® software and SCL (Screen Control Language). The system was created under SAS 6.09 on a UNIX platform but is not limited to UNIX. This application was developed for telephone representatives in the health care arena who needed quick and efficient ways to respond to customers. The techniques used to build the system can be easily modified to suit any application. As newcomers to the applications development arena, the authors provide a gentle step-by-step guide of the 'behind the scenes' techniques used in development.

Introduction

Blue Cross and Blue Shield of South Carolina administers a number of government health care contracts under the name Palmetto Government Benefits Administrators. Among these contracts is the Statistical Analysis Durable Medical Equipment Regional Carrier (SADMER), a support organization for Medicare’s Durable Medical Equipment Regional processors.

One duty of the SADMER is to support a HCPCS (HCFA Common Procedure Coding System) Helpline with phone representatives who provide proper procedure coding guidance to manufacturers and suppliers of durable medical equipment. The phone representatives listen to product descriptions and provide the caller with the most appropriate procedure code for billing the item to the regional carrier. In addition, the Helpline responds to pricing questions for items (procedure codes) priced through a fee schedule.

In an effort to provide customers with accurate, up-to-date, and timely information, an on-line information system was developed using SAS/AF, SAS/FSP, and SCL on the UNIX platform (SAS 6.09). Prior to the on-line system, the phone representatives were burdened with paper documentation listing over 3,500 procedure codes and descriptions. Separate documentation existed for the fee schedule data which includes individual pricing for each state and territory of the United States. Relieved of this paper chase, the Helpline representatives can now quickly query data via an on-line database.

The HCPCS Information System currently consists of two components accessed through a Main Menu. In the Fee Schedule component, developed using SAS/FSP, you enter the procedure code in question and optionally enter other parameters which may determine the price. In the HCPCS Procedure Lookup component, developed using a SAS/AF frame entry, you may look up specific procedure codes or enter any keyword that describes the code, and all applicable codes will appear on the screen.

Hardware/software

At our site, the SAS system resides on a RISC/6000 which utilizes AIX UNIX. The end-users access the system through personal computers connected to the RISC/6000 by a token ring using TCP/IP and Hummingbird software.

How to develop it

I. Main Menu

Access to the SAS/AF build screen can be granted by submitting a PROC BUILD statement along with the appropriate libname statement(s) or issuing the build command at the command prompt.

Once in the build environment, create or choose an existing catalog. We named our catalog HCPCS.

File => New => Catalog

The first frame (MAIN.FRAME) of the application can be created with the following command selections.

File => New => Entry

A dialog box pops up requesting an entry name (MAIN) and entry type (FRAME). The application we developed is comprised of several frames or windows with graphical objects that are linked together. These frames allow you to access and query databases with little or no training. The Main Menu is a frame entry comprised of three types of widgets - two graphic text objects, two icon objects and two push-button objects.
The **graphic text objects** "SADMERC" and "HCPCS Information System" in Figure 1 can each be created with the following command selections.

**Actions => Make => Graphic Text**

Position the box where you want text displayed. Once you have positioned the box, a Graphics Text Attributes window pops up. Enter the text you want displayed and other features such as font and font size. The default name of this widget, OBJN, can be left unchanged. The Actions menu can also be accessed through a pop-up menu by clicking on the right mouse button.

The **icon objects** "Fee Schedule" and "HCPCS Lookup" in Figure 1 can similarly be created by selecting

**Actions => Make => Icon**

and positioning the box where you want the icon displayed. (The first one named FEES will access a customized FSEDIT screen, and the other named PROC will access another frame entry.) Once you have positioned the icon box, an Icon Attributes window pops up so that you can enter the name of the icon (FEES and PROC) and the label you want displayed (Fee Schedule and HCPCS Lookup, respectively). The names of the icons, FEES and PROC, will refer to corresponding labeled sections of the SCL program. By selecting the current icon, you can browse through the SAS-supplied icons.

The **push button objects** "Help" and "Exit" in Figure 1 can be created by selecting

**Actions => Make => Pushbutton**

and positioning the box where you want the pushbutton displayed. (The pushbutton named HELP will call a frame entry with instructions on where you can call for assistance, and the one named EXIT will access a frame entry that prompts you before exiting the system.) Once you have positioned the pushbutton box, a Pushbutton Attributes window pops up where you enter the name of the pushbutton (HELP and EXIT) which will refer to labeled sections in the SCL program and the label you want displayed (HELP and EXIT, respectively).

**Screen Control Language**

According to *Building SCL Applications Using Frame Entries Course Notes* (p.81), SCL programs contain several labeled sections. The INIT, MAIN, and TERM sections are reserved (but not required) labels. A section begins with an object name followed by a list of instructions and ends with a return statement.

From MAIN FRAME, select the following commands to access the screen where the SCL code is stored.

**Locals => Edit SCL source**

The INIT section executes before the application window is displayed. The MAIN section executes if any object is activated (this section is not needed in this SCL program). The TERM section executes when the application window is closed by an END or CANCEL command.

The other sections (FEES, PROC, EXIT and HELP) refer to objects on the main menu. Selecting one of these causes the corresponding labeled sections to execute.

**Behind the scenes SCL for the initial frame (MAIN.SCL)**

```
INIT:
    _msg_="*** Welcome to the SADMERC HCPCS Information System ***";
    return;

FEES:
    call fsedit('in.demos1','in.fee','browse');
    return;

PROC:
    call display('procs.frame');
    return;

EXIT:
    call wregion(18,20,10,40);
    call display('exit.frame');
    return;

TERM:
    link exit;
    _status_='R';
    return;

HELP:
    call display('helpdesk.help');
    return;
```
The INIT Section

The INIT section will display the message “Welcome to the SADMERC HCPCS Information System” on the message line every time the application is invoked.

FEES Section

Once you click on the FEES icon, the FEES section will execute. An FSEDIT screen will appear that accesses the DMEPOS database, using a customized screen called FEE. *Browse only access will be allowed to the end user.*

PROC Section

When you click on the PROCs icon, the PROCs section will call another frame entitled PROC.FRAME. Once this frame is initiated, control will be passed to the SCL associated with the PROC.FRAME window. PROC.FRAME is described in Section III (HCPCS Procedure Lookup Component).

HELP Section

Selecting the HELP pushbutton will call a Help entry called HELPDESK.HELP which will instruct you where to call for assistance. See *Building SCL Applications Using FRAME Entries Course Notes.*

EXIT Section

Selecting EXIT will call another frame with the following dimensions. The frame will start in row 18 and column 20 and be 10 rows long and 40 columns wide. EXIT.FRAME entry will appear in this customized frame with the message “Are you sure you want to exit?”. If you choose not to exit the program, the “_STATUS_="R";” statement will resume execution of the application.

Order of Execution

The order in which this program runs should help clarify the SCL code. As previously mentioned, the INIT section executes before the frame (MAIN.FRAME) is displayed to you. When you initially see MAIN.FRAME (figure 1), the text that appears on the message line is a result of the execution of the INIT section. When you select the icon labeled Fee Schedule and named FEES, the FEES section of code executes. Likewise, when you select the icon labeled HCPCS Lookup and named PROCs, the PROCs section of code executes. If you select the pushbutton labeled and named HELP, the HELP section will execute. Finally, when you choose to exit the program, the TERM section will execute, which links to the EXIT section before control is returned back to the TERM section.

Compiling and Testing

Sections of code can be compiled and tested at any point. "During the compilation phase, the BUILD procedure checks syntax of each SCL statement and builds the SCL vector for the program. If no errors are detected, then the program is compiled" (SAS Screen Control Language Reference, p. 47). Compiling SCL code is easily done by issuing the following command selections.

Locals ⇒ Compile

"Once the code is compiled successfully, you can test the entry by issuing the TESTAF command."(SAS Screen Control Language, p47).

Locals ⇒ Testaf

Debugging

To actually step through the execution of the program or to locate errors, turn on the Debug facility.

While in the SCL program, use the following series of command selections.

Locals ⇒ Debug ⇒Debug on
Locals ⇒ Compile
Locals ⇒ Testaf

The SCL Source Language Debugger window appears with your SCL code, and a MESSAGE window appears. You can step through the execution of the program by using the ENTER key to execute SCL code a line at a time. To exit the Debugger, go to the Message window and issue the following command selections

Run ⇒ Quit Debugger

and from the SCL Source Language Debugger window, enter

Locals ⇒ Debug ⇒ Debug off.

II. Fee Schedule Component

The Fee Schedule component allows you to access pricing information for the procedure codes. When the Fee Schedule Icon is selected, the SCL program behind the Main Menu executes the section labeled FEES. A customized FSEDIT screen appears which accesses the DMEPOS1 data set and a saved customized screen named FEE. The customized screen FEE was created using
Posters

SAS/FSP: A customized FEDIT screen can be created by submitting the following statement in the SAS Program Editor window:

```
libname in '/your directory';
proc fssedit data=in.dmeapos1 screen=in.fee;
run;
```

A default FEDIT screen will appear with the variables that exist in the DMEPOS1 data set. To make the custom screen, enter:

Locals ⇒ Modify Screen...

You may either enter a password for security purposes or bypass this option by selecting the OK pushbutton. The FEDIT Menu appears. Screen Modifications can be made by selecting option 2 - Screen Modifications and Field Identification. In the FEDIT MODIFY window, you can move variables around using text editor commands, add titles and/or more descriptive text, and modify the colors of the display through the VIEW menu. Once you END the FEDIT MODIFY window, SAS will ask you if you created any computational or repeated fields. None of these were created in this particular application. The FEDIT IDENTIFY window appears next. According to the SAS/FSP Software Manual (p. 290), when the FEDIT IDENTIFY window is opened, the status of each field is determined to either be opened, unidentified or unwanted. Before you can exit the field identification step, all fields must either be identified or defined as unwanted. SAS will prompt you with a message once it knows the location of all variable fields.

Control is then passed to the SCL behind PROCS.FRAME.

The second frame of the application (PROCS.FRAME) can be created similarly to MAIN.FRAME with the following command selection:

File ⇒ New ⇒ Entry

and entering an entry name (PROCS) and entry type (FRAME). The Procedure Lookup is a frame entry comprised of an extended table, two text entries (plus one additional optional text entry), text labels, a graphic text object and a control object.

![Figure 3](image-url)

In figure 3, the graphic text, PROCEDURE LOOKUP, is a graphic text object which can be created by selecting:

Actions ⇒ Make ⇒ Graphic Text

and positioning the box where you want text displayed. Once you have positioned the box, a Graphics Text Attributes window pops up. Enter the text you want displayed and other features such as font and font size. The default name of this widget, OB2In, can be left unchanged. The same menu can also be accessed through a pop-up menu by clicking on the right mouse button.

The text entry objects can similarly be created by selecting:

Actions ⇒ Make ⇒ Text Entry Object

and positioning the box where you want the text entry object displayed. Three text entry objects will need to be created: one named VARNAME, which will allow you to enter the name of the variable, and another named VALUE, which will allow you to enter a procedure code value. The third text entry is named NOTE and will display different messages as you progress through the application. To accommodate the length of the messages...
that NOTE will display, the text entry object needs
to have a length of 30 and be 2 rows long. Text labels are
similarly created to label the text entry fields on the
display.

The extended table object can be created by selecting

Actions ⇒ Make ⇒ Extended Table

and positioning the box where you want the extended
table displayed. An extended table attribute window pops
up. Assign the fields in your extended table using the
attribute values in Figure 4.

![Extended Table Attributes](image)

**Figure 4**

Inside the extended table box is a container box. This is
where text entry objects need to be created for each
variable that will be displayed in the extended table. The
text entries, PROC (procedure) and DESC (procedure
description), were created inside the container box. The
steps to create text entries were described earlier.

**Behind the scenes - SCL for the Procedure Lookup (PROCS.SCL)**

```scl
INIT:
call notify('note', '_set_text_','
   'Please Select a VARIABLE NAME and enter a
   SEARCH VALUE');
proc = open('in.lproc');
call set(proc);
return;

MAIN:
call notify('note', '_set_text_','Enter another SEARCH
   VALUE or VARIABLE/SEARCH
   COMBINATION');
if (value = '_blank_') then
do:
   temp2 = varname = ^null;
   if (where (procs, temp2)) then _msg_ = sysmsg();
   link count;
end;
call notify('OBJ1', '_need_refresh_');
_msg_ = put(num,4.),' records found. Use F3 key if
you wish to exit;'
return;

COUNT:
   num=0;
do while (fetch(procs) ne -1);
   num=1;
end;
return;

TERM:
close(procs);
return;

GET1:
   rec = fetchobs(procs, _current_);
   if rec=1 then call notify('OBJ1', '_endtable_');
return;
```

**The INIT Section**

This section displays an initial message in the text entry
called NOTE which prompts you to enter a variable name
(VARNAME) and search value (VALUE).

The SAS data set "in.lproc" is opened with a data set id
called PROCS. The data set contains two variables,
PROC (procedure code) and DESC (procedure
description). The CALL SET routine associates all data
set variables with SCL variables of like name and type.
The text entries, PROC and DESC, positioned within the
container box of the extended table are now associated
with the data set variables PROC and DESC from the data
set id PROCS.

**The MAIN Section**

The CALL NOTIFY routine prompts you to enter a
search value (VALUE) or variable and search value
combination (VARNAME and VALUE). If the first IF
statement will clear the WHERE function if you have not
specified a VARNAME or VALUE.

The second IF statement executes if the variable name
(VARNAME) or value (VALUE) is modified by you. A
WHERE statement is created, and the data is subset to
meet the condition in the WHERE statement. Control is
passed to the COUNT section where the number of
observations in the requested subset is counted. Control
returns to the MAIN section, where a message is displayed giving you the number of records meeting your criteria and giving you the opportunity to exit the application using the F3 key if you wish.

The TERM Section

The CLOSE routine closes the data set id. PROCS.

The GET1 Section

This section is the GETROW section for the extended table. GET1 executes repeatedly until either a full window of the table is displayed or until the final row in the table is displayed. For a dynamic table, such as this, the final row occurs when the extended table method ENDTABLE is called.

Order of Execution

The order in which this program runs should help clarify the SCL code. The INIT section executes prior to the frame (PROCS.FRAME) being displayed. Control is then passed to the GET1 (GETROW) section, which fills the extended table. Once the table is filled and the frame is displayed to you, the MAIN section executes after you enter a request. The COUNT section will execute next and control passes back to MAIN, where the extended table is updated and the number of records meeting the requested criteria is displayed on the message line. The TERM section executes when the F3 key is hit and closes the PROCS data set.

Conclusion

SAS/AF with Screen Control Language is a powerful and flexible tool for developing applications. The authors hope that sharing their experiences will help promote more creative application development.

References


Authors

Verna C. Brantley
Blue Cross and Blue Shield of South Carolina
SADMERC 9th Floor
I-20 at Alpine Road
Columbia, South Carolina
(803) 788 - 0222 (ext. 41122)

Helen L. Lintern
Blue Cross and Blue Shield of South Carolina
SADMERC 9th Floor
I-20 at Alpine Road
Columbia, South Carolina
(803) 788 - 0222 (ext. 41129)

Acknowledgments

The following employees of Blue Cross and Blue Shield of South Carolina contributed to the preparation of this paper.

Mike Edge Gail Ward-Besser
Julie Flowers Jeannie Wingard

SAS, SAS/AF, SAS/FSP are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.