Controlling Sealed-Source Inventories Using the FRAME Entry in SAS/AF® Software

Ralph R. Royce, Martin Marietta Utility Services, Inc., Piketon, Ohio

Abstract

A major goal of application development is to provide a user-friendly environment. The application should have an intuitive look and feel. Navigation through the application should require only a recognition of the variable names and values. By requiring little or no understanding of the inner-workings, the application is easily accessible by all who use it. This is important because you do not need to have any knowledge of the underlying programming language.

The menu structure should follow a logical sequence, having related topics listed under each menu. The menus and controls of the various windows should supply options and buttons that are dynamically-created. By using these guidelines, you are more likely to use the application and obtain the desired information with the least difficulty.

‘PORTS - Radioactive Source Control Program’ meets these goals. This application provides a convenient interface to the sealed-source inventory and provides information about various attributes of these radioactive sources.

FRAME Entries in SAS/AF software provided the optimal tools for developing this application. The development of the application in the Windows environment, due to its graphical nature, further enhances the usability of this inventory program.

Introduction

The need for an inventory application arose because the following objectives needed to be satisfied:

1. A database of source inventory information existed and the data needed to be easily viewed and updated.
2. The user needed to have the application notify them when integrity testing and inventories needed to be performed.
3. The user needed a menu structure that would allow them to navigate easily through the application.
4. Reports needed to be generated without needing to know proper syntax, variables, or values.

The old method required a knowledge of these items and the inventories and integrity tests were not always completed on time. Therefore, user-friendliness was the primary goal of this application. After having attended a class at the SAS Institute on FRAME Entries in SAS/AF, I used this new found knowledge to develop this application. Using the BUILD command, I created the library and began creating the menu screens. The widgets were very easy to work with and made the application creation proceed smoothly. The SCL code, for the menus, was very straightforward to create. It wasn’t until I began delving deeper into the bag of widgets that I came to realize the wealth of widgets that were available and to understand the power of the FRAME Entry. This power is accessed by using SCL code to modify the widgets using various methods and functions.

The techniques that I apply here are generic and can be used by all SAS/AF users to create user-friendly applications.

Application Invocation

This application works in the Windows operating system environment. A new program group was added to the Program Manager window. Then a new program item was added to this new group. The command line was modified to start the SAS session by executing the SAS.EXE command. The Working Directory is set to where the inventory data sets and catalogs are located. To invoke the application, open the inventory group and double-click the inventory program icon.

At this point, two special files, CONFIG.SAS and AUTOEXEC.SAS are executed. Just like in DOS the config file is executed first. Six SAS system options were added to the default CONFIG.SAS file. These options are shown in the following code:

1105
search for formats. Application invocation is performed by the last line.

**Main Menu**

The Main Menu presents you with four choices: 1) Database, 2) Charting, 3) Reports, and 4) Exit. Display 1 shows the menu layout. To select an option, you need to click on the button of your choice. The SCL code then displays the selected frame. The following SCL code drives this menu:

```sas
init:
  /* create an SCL list and then insert character *'
  /* items into the list *'
  sid is the identifier for source databases
  vid is the identifier for supporting databases

  sid = makelistO;
  sid = insertc{sid, 'sources', 'sources'};
  sid = insertc{sid, 'rfd', 'rfd'};
  return;

  term:
  /* deletes the SCL lists *'
  rc = dellist{sid};
  rc = dellist{vid};
  return;

  database:
  /* set the window title bar and display the *'
  /* Database Menu if the Database button is chosen *'
  call execcmd('setwindow "Database Menu"');
  call display('source.source.database.frame');
  return;
```

The log, output, and pgm windows are turned off in the init section. This limits the number of windows that are available. If any sources need inventoried or integrity checked the Reports menu is displayed. The term section iconizes the application and then ends the SAS session. This limits you from roaming to other SAS windows.

**Database Menu**

The Database menu button takes you to the Database Modification Menu where the database can be viewed using SAS/FSP software. Another menu item views supporting databases that use SAS/VIEW to enter or modify data. The last menu item allows for updating the integrity testing information. The SCL code is shown below:

```sas
init:
  /* create an SCL list and then insert character *'
  /* items into the list *'
  sid is the identifier for source databases
  vid is the identifier for supporting databases

  sid = makelistO;
  sid = insertc{sid, 'sources', 'sources'};
  sid = insertc{sid, 'rfd', 'rfd'};
  return;

  term:
  /* deletes the SCL lists *'
  rc = dellist{sid};
  rc = dellist{vid};
  return;

  database:
  /* set the window title bar and display the *'
  /* Database Menu if the Database button is chosen *'
  call execcmd('setwindow "Database Menu"');
  call display('source.source.database.frame');
  return;
```
Reports Menu

The Reports Menu displays three buttons and the number of observations that need inventoried, integrity tested, and the total sources in the database. Display 2 shows this menu. If there are no items that need inventoried or integrity tested then their buttons are grayed and cannot be selected. Since there are no radioactive sources that need inventoried the Custodian Inventory Report button is grayed. The buttons are not grayed if an item needs inventoried or integrity tested within thirty days. Selecting a nongray button will execute a submit block and display a report. After the report is displayed, clicking the right mouse button and choosing Print from the menu will print the report. This is an example of a p-menu created with options that are only for this frame. The following details the SCL code for this frame:

```
init:
/* sets the fileref to be used for the output from */
/* the submit block */
rc = filename('out', 'C:sources/output.rpt', 
'.', recfm = 'p');
/* set length for variables */
length inventory leaktest database 4;
/* open the source database and set a where */
/* clause for sources that are inventoried */
dsid = open('source.sources');
rc = where(dsid.inv_able = 'Y');
/* count the number of sources by performing */
/* statistics on the data set */
rc = varstat(dsid, 'inv_date', 'n', database);
/* use the notify method to set the text label */
call notify('entire', 'set_text', 'database || OBS in database.');
/* apply a where clause for sources that are due */
/* to be inventoried in 30 days */
rc = where(dsid.inv_date < today()-150);
/* count the number of sources needing */
/* inventoried */
rc = varstat(dsid, 'inv_date', 'n', inventory);
/* use the notify method to set the text label */
call notify('inventn', 'set_text', 'inventory || OBS need inventoried.');
/* set a where clause for sources that are due */
/* to be integrity tested in 30 days */
rc = where(dsid.leakdate < today()-150 and leakable = 'Y');
/* count the number of sources needing */
/* integrity tested */
rc = varstat(dsid, 'leakdate', 'n', leaktest);
/* use the notify method to set the text label */
call notify('integn', 'set_text', 'leaktest || OBS need integrity tested.');
/* if the number of sources is 0 then gray the */
/* button so that it cannot be selected */
if inventory eq 0 then call notify('invent', 'gray');
if leaktest eq 0 then call notify('integ', 'gray');
if database eq 0 then call notify('entire', 'gray');
return;
```

```
invent:
/* submit the following code if the Custodian */
/* Inventory Report button is chosen */
submit continue;
/* set the usable area of the page, put date and */
/* page number starting with 1 on the report */
options linesize = 80 pagesize = 60 date number
pageno = 1;
/* print to the fileref create above */
proc printto print = out new;
title PORTS - Radioactive Source Control
Program;
title2 Custodian Inventory Report;
/* use PROC REPORT to format the output with */
/* a where clause set for only sources that need */
/* inventoried in 30 days */
proc report data = SOURCES SOURCES
report = SOURCES.CINVRPT.REPT
nowd;
where inv_date < today()-150 and
inv_able = 'Y';
by custod;
/* close the printto procedure */
proc printto;
run;
endsubmit;
```
/* place the report in the fileref rc = preview('include', 'out'); */
/* set the window title bar name call execcmd('setname "Report Output"'); */
/* set the p-menu that will be used call execcmd('setpmenu
source.pmenus.preview.pmenu;'); */
/* set the form to use to print the report call execcmd('formname
source.source.default.form;'); */
/* set the background color to blue call execcmd('color background blue;'); */
/* set the window size to the largest possible sr = winfo('startrow');
sc = winfo('startcol');
nc = winfo('numcols');
call wregion(sr,sc,nc+1,nc+1); */
/* browse the preview buffer and then clear it
rc = preview('browse');
rc = preview('clear');
return; */

entire:
/* submit the following code if the Entire */
/* Database Report button is chosen */
/* submit continue; */
/* set the usable area of the page, put date and */
/* page number starting with 1 on the report */
/* options linesize = 132 pagesize = 45 date */
/* number page = 1; */
/* print to the filer create above */
proc printto print = out new;
title PORTS - Radioactive Source Control
Program;
title2 Sources Requiring Integrity Testing
Report;
/* use PROC REPORT to format the output with */
/* a where clause set for only sources that need */
/* inventoried in 30 days */
proc report data = SOURCE.SOURCES
report = SOURCE.SOURCE.SRLT.REPT nowd;
where leakdate <today()-150 and
leakable = 'Y';
by custod;
/* close the printto procedure */
proc printto;
run;
/* end the submit block */
end submit;
/* place the report in the preview buffer */
rc = preview('include', 'out'); */
/* set the form to use to print the report */
call execcmd('formname source.source.default'); */
/* set the background color to blue */
call execcmd('color background blue;'); */
/* set the form to use to print the report */
call execcmd('setpmenu
source.pmenus.preview.pmenu;'); */
/* set the window size to the largest possible */
* sr = winfo('startrow');
sc = winfo('startcol');
nc = winfo('numcols');
call wregion(sr,sc,nc+1,nc+1); */
/* browse the preview buffer and then clear it */
rc = preview('browse');
rc = preview('clear');
return; */

term:
/* close the data set */
rc = close(dsid); */
return; */

Charts Menu

During the process of creating this application, a question
regarding the quantity of fissile sources stored in source
storage lockers needed investigated. A regular SAS program
could provide this information. It was decided that this informa-
tion could be displayed graphically in this application. Bar
charts were tried first and they did not fully satisfy the needed
look and feel. The critical success factor widget was tried next.
Using this widget, zones were set to keep track of the fissile
inventory in relation to the levels that were posted for each
repository. Display 3 shows this screen. The standard red, yellow, green range color scheme is reversed for the two levels. The two levels used on plantsite were for a maximum of 750 grams of $^{235}\text{U}$ and for a maximum 350 grams of $^{238}\text{U}$ in each repository. Using an IF/THEN structure and the NOTIFY method the range for the CSFGRAPH is set. Various widgets could display the information, but the BUTTON widgets performed the best. The NOTIFY method then set the labels on the buttons. Pop-up menus display the choices for the buildings and the rooms. The following SCL code drives this frame:

```scl
/* set length of variables that are used later */
length msum 8 chooseit u_nuc u_nuclab $30;

/* open the sources data set */
dsid = open('source.sources');

/* locate where the building and room buttons are located */
call notify('b_button', '_column', bldgcol);
call notify('b_button', '_row', bldgrow);
call notify('r_button', '_column', roomcol);
call notify('r_button', '_row', roomrow);

/* create a list to contain the buildings and rooms */
bltid = makelist();
roomid = makelist();

/* no return statement for init section */

/* which isotope radio box is selected */
call notify('uranium', '_is_active', station);

/* get the label for that radio box */
call notify('uranium', '_get_label', station, chooseit);

/* set the variable chooseit to be used in a later where clause */
if chooseit ne 'All' then chooseit = '1';
else chooseit = blank;
umval = 0;

/* apply where clause */
rc = where(dsid, u_nuc | chooseit);

/* are there any inventoried observations? */
numobs = attrn(dsid, 'ANY');

/* if not then select the 'All' radio box */
undo the where clause and start over
if numobs = 0 then do;
rc = activate('choose', 2);
rc = where(dsid);
goto uranium;
end;
/* fill an SCL list with unique values for */
/* building number from the source data set */
rc = lvarlevel(dsid, 'bldg', numval, bltid);

/* sort the list */
bltid = sortlist(bltid);

/* select the first item in the list */
bldgchoice = 1;
bldg = getitemc(bltid, bldgchoice);
/* gray the selected item */
rc = setlattr(bldid, 'INACTIVE', bldgchoice);

/* no return statement for choose section */

bldg: /* apply where clause */
rc = where(dsid, bldg = '1' | chooseit);

/* are there any observations? */
numobs = attrn(dsid, 'ANY');

/* if no new building is selected then */
if numobs = 0 then do;
oldbldg = bldg;
set bldg to missing
bldg = blank;
remove all where clauses
rc = where(dsid);
do while(bldg = blank);
end;
/* display building SCL list in a pop-up menu */
newbldgchoice = popmenu(bldid, listlen(bltid), bldgrow + 5, bldgcol - 2);
/* if no new building is selected then */
/* reset bldg to original value */
if newbldgchoice = 0 then do;
bldg = oldbldg;
return;
end;
bldg = getitemc(bldid, newbldgchoice);
/* ungray the old building choice in the SCL list */
rc = setlattr(bldid, 'ACTIVE', bldgchoice);
/* gray the new building choice in the SCL list */
rc = setlattr(bldid, 'INACTIVE', newbldgchoice);
/* make the new choice the current choice */
bldgchoice = newbldgchoice;
end;
/* no return statement for bldg section */

1109
Conclusions

The SCL programs used with their respective FRAME entries that have been presented here show the power and flexibility that are available for making user-friendly applications. A large arsenal of tools for application development, contained in SAS/AF, provides the programmer with a wealth of widgets and methods that can manipulate these widgets. This is done with SCL functions. FRAME Entries make menu navigation and use extremely easy because of the graphical nature of the Windows environment.

Use of dynamically-created menus ensures that the retrieval of correct information requires only a click of the mouse button. The alternative, keyboard entry, can be a nuisance as one typographical error can cause an application to operate incorrectly. The concepts presented here are only a sampling of the capabilities of FRAME entries in SAS/AF software and can be modified to be used in other areas.

SAS, SAS/AF, SAS/EP, and SAS/VIEW 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.

The submitted manuscript has been authored under contract USECHA-93-C-001 with the United States Environement Corporation, an agency of the U.S. Government. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of the contribution, or allow others to do so, for U.S. Government purposes.