# Creating Flexible Reporting Applications Using SAS/AF® Software

Candy R. Habich, Eastman Chemical Company

## INTRODUCTION

All of the information contained in this paper was gleaned from real-life SAS/AF applications at our site. Since I am the site consultant, it is my job to help developers figure out how to make their applications do what their users want them to do. This is sometimes challenging but always rewarding. The details of the particular application are not important. The techniques that we developed will be useful in any situation where you want to provide an extremely easy and flexible report generator. The techniques described in this paper address the following specific requests from the user and the SAS components used to satisfy them.

### REQUEST #1: LIST VALUES FROM ACTUAL DATA

Selection list values are derived from the current data values of a variable with all other selection criteria applied. The values presented to the user might NOT include all possible valid values for a variable as would be done in a data-entry application. For example, if a sales file contained sales to customers in only 30 of a possible 50 countries in which the company did business, the user would get a selection list with only 30 entries instead of 50. Furthermore, if the user has already specified other selection criteria such as customers who purchased a particular product, the country list might be reduced to even fewer values. The beauty of this is that the user can not select things which don't make any sense. An added benefit from using this approach is that the act of selecting data values for a report can become a useful inquiry process in and of itself.

This reqeust is satisfied through the components of PROC SQL views and SCL lists. This is implemented using several "joined views" which make all the possible selection list variables available in one place. There is some response time tradeoff involved in gaining this added flexibility when large numbers of variables and/or observations are involved so this approach may not be practical for every application.

Keeping up with all the user's selection criteria in this flexible environment is another challenge. But, SCL lists (new in release 6.07) simplify the task considerably. They allow us to "collect" everything and hang on to it until we are ready to actually process the data. (Note: I have successfully used the same concept with temporary SAS data sets using release 6.06 under OS/2 but I like SCL lists much better.)

### REQUEST #2: REFRESH LISTS WITH PRIOR CHOICES

When selecting qualifying data for a report (for example, selecting from a list of countries), the user might want to make some selections from one list, then another list, and then perhaps go back to modify selections from the first list. The values he/she selected the prior time should appear selected when the user returns to that list.

This was satisfied through the use of SCL extended tables. Using standard SCL functions such as "datalistc", the user's initial set of selections can not be "pre-selected". However, using extended tables, you can accomplish the pre-selection without much extra code. This little bit of extra code goes a long way toward making users want to use an application. This technique is incorporated within the general-purpose selection lists in the sample application shown.

The primary elements of a sample application which incorporates these techniques are described below.

## SELECTION LIST "VIEWS"

A SAS program shown below creates "selection list views" for the application. When more than one data set is involved, they must be joined together so that all of the possible variables for which the user could make selections appear as though they are all in one file. The ability to create "in-line views" using PROC SQL greatly simplifies this task.

The example views created below use data from the sample files which come with the SAS/ACCESS® interface for SQL/DS. Notice that three "levels" of views are created. The sample application discussed in this paper will use only the SUGILIST.ALL view. However, when dealing with large amounts of data, it may be of benefit to use only the "minimum" view needed to select the required variables.

```
proc sql;

   /* only what's needed for  */
   /* customer selections     */
   create view SUGILIST.CUSONLY as
     select customer,
            country,
            state,
            city
     from sugivw.custom
   ;

   /* only what's needed for  */
   /* order/invoice selections*/
   create view SUGILIST.ORDRINV as
     select a.*,
            b.ordernum,
            b.stocknum,
            b.invoice
     from sugilist.cusonly a left join
        (select c.shipto,
                c.ordernum,
                c.stocknum,
                d.billedto,
                d.invoice
           from      sugivw.orders  c
           left join sugivw.invoice d
           on c.shipto   = d.billedto and
              c.ordernum = d.invoice
        ) b
     on a.customer = b.shipto
   ;

   /* contains vars necessary  */
   /* for all selections       */
   create view SUGILIST.ALL as
      select a.*,
             b.fibname
      from sugilist.ordrinv a left join
           sugivw.product   b
      on a.stocknum = b.prodid
   ;
```

Please keep in mind that these views could be created in exactly the same way if the data were stored in SAS data sets. PROC SQL is an extremely powerful tool no matter where your data reside.

## PROGRAM DISPLAYING SELECTION VARIABLES

A SAS/AF program presents the user with a list of variables from which to make selections. This is an extended table used as a menu. The variables listed on this screen are taken from an application

parameter file which contains a description for each variable as well as its type (character or numeric). In addition, there is a multi-station choice group which allows the user to indicate whether or not he would like to subset the data by selecting specific values of the selected variable.

## MAIN.PROGRAM

```
+-----------------------------------------------------------+
|                                                           |
| Tab to desired choice and press enter:                    |
|                                                           |
|^^^                                                        |
|                                                           |
| &SELVRDSC__________ ( &SELVAR_ ) &C1_ &C2_ &LISTID &T     |
|                                                           |
+-----------------------------------------------------------+

***** ATTRIBUTES *****
  System Options: EXTENDED TABLE
  Alias for T is: SELVRTYP
  C1 list value is "0", initial text is "All",
     choice group is "choice"
  C2 list value is "1", initial text is "Some",
     choice group is "choice"

***** SOURCE *****
init:
  control enter;
  submit continue sql;
    create table vars as
    select distinct category, /* variables are read    */
           selvrdsc,          /* from the application */
           selvar,            /* parameter file       */
           selvrtyp,
           '0' as choice,     /* selection choice     */
           .   as listid      /* initial value is ALL.*/
      from  SUGILIST.PARMS
      order by category, selvrdsc
      ;
    quit;
  endsubmit;
  nrows = symgetn('sqlobs');
  dsid = open('vars','i');
  call setrow(nrows);      /* extended table does NOT  */
  call set(dsid);          /* act as selection list    */
return;

main:
return;

term:
  if dsid > 0 then rc = close(dsid);
  call display('subview.scl');/* subset data once more */
  call fsview('work.list');   /* report would go here  */
return;

putrow:
  if choice = '1' then do;          /* "Some" selected   */
     if selvrtyp = 'C'
        then pgm = 'charlist.program';/* generic list  */
        else pgm = 'numlist.program'; /* for char/num  */
     call display(pgm,listid,selvar);
     if listid > 0 then choice = '1'; /* no selections */
                   else choice = '0'; /* reset to "All"*/
  end;
  else do;                          /* "All" selected*/
     if listid > 0
       then rc = dellist(listid);    /* delete list   */
     listid = .;                     /* & list id     */
  end;
  dsid2 = open('work.vars','U');  /* Update parms with */
  if dsid2 > 0 then do;           /* value of list id  */
     rcw = where(dsid2,"selvar  = '"||selvar||"'");
     rcf = fetch(dsid2,'NOSET');
     call set(dsid2);
     rcu = update(dsid2);
     rcc = close(dsid2);
  end;
return;

getrow:
  rc = fetchobs(dsid,_currow_);
return;
```

## GENERAL PURPOSE SELECTION LIST

The next component is a SAS/AF program entry which presents a selection list using an extended table. This general purpose selection list can be used to display values for any character variable up to 35 characters in length. The General Attributes (GATTR) window defines this to be an extended table. There is a companion selection list program for displaying numeric variables which is not shown.

Please note that this selection list only displays the value of a single variable. In many applications, it would probably be desirable to display values of a companion variable as well. For instance, you might identify customers using a customer code but it would be useful to display the customer name with it. However, the basic concepts for refreshing the user's selections would not change.

This program performs the tasks described below. The source code corresponding to each task is identified in a comment block.

A. Creates a temporary SAS data set which is used to populate the extended table
B. Opens the data set and establishes the connection to the extended table
C. Determines whether or not the user has previously made any selections
D. Notifies the user if a "repeat find" command wrapped to the top of the file
E. Saves any selections in an SCL list upon termination of the program

## CHARLIST.PROGRAM

```
***** DISPLAY *****

+-----------------------------------------------+
|                                               |
|   ***  Tab to desired choice and         ***  |
|   ***  press <Enter> to select/deselect.  ***  |
|^^^                                            |
|      &CHARVAR_____________________            |
|                                               |
+-----------------------------------------------+

***** SOURCE *****
 entry listid 1 passvar $;
 init:

   control always;
   /* Task A: - Starts Here                                  */
   /* This step would be optional if you had a permanent     */
   /* data step in which the valid country values were       */
   /* stored.                                                */
   call display('subview.scl',passvar);
   submit continue sql;
     create table temp as
        select distinct &passvar as charvar
        from list
           where &passvar ^= ' '
           order by charvar
     ;
     quit;
   endsubmit;
   /* Task A - Ends Here                                     */

   /* Task B - Starts Here                                   */
   /* Number of rows in table determined from automatic      */
   /* PROC SQL macro var &SQLOBS.  Could also be found       */
   /* using ATTRN function after data set is opened.         */
   /* Use standard extended table routine SETROW (it is      */
   /* (NOT a dynamic extended table).  Use SET routine       */
   /* to establish connection between SCL data vector        */
   /* and data set data vector.                              */
   nrows = symgetn('sqlobs');
   dsid = open('temp','i');
   call setrow(nrows,nrows,'N','N');
   call set(dsid);
   /* Task B - Ends Here                                     */
```

```
  /* Task C - Starts Here                                    */
  /* If the value of the list identifier is greater          */
  /* than zero, then previous selections have been           */
  /* made.  In that case, read each row from the data        */
  /* set and look for its value in the list.  If in the      */
  /* list, select it.                                        */
  if listid > 0 then do crow = 1 to nrows;
    rcfetch = fetchobs(dsid,crow);
    lgposin = searchc(listid,charvar);
    if lgposin > 0 then rcsel = select(crow);
  end;
  /* Task C - Ends Here                                      */
return;

main:
return;

term:
  /* Task E - Starts Here                                    */
  /* If the user issues END command, list is deleted         */
  /* (if it exists) and re-created.  Current selections      */
  /* are then stored in the new list.                        */
  if _status_ = 'E' then do;
    if listid > 0 then do;
      rc = dellist(listid);
      listid = .;
    end;
    if nselect() > 0 then listid = makelist();
    do nsel = 1 to nselect();
      crow = selected(nsel);
      rcfetch = fetchobs(dsid,crow);
      listid = insertc(listid,charvar,-1);
    end;
  end;
  /* Task E - Ends Here                                      */
  if (dsid > 0) then rc=close(dsid);
return;

getrow:
  /* Task D - Starts Here                                    */
  /* This is just something nice to do since, by             */
  /* default, you do not get a message when an RFIND         */
  /* searches to the end of the list and back to the         */
  /* top.  You could press RFIND for a long time before      */
  /* realizing you had "wrapped" around!                     */
  if word(1,'U') = 'RFIND' and _currow_ = 1
     then _msg_ = 'Wrapped to top of file.';
  /* Task D - Ends Here                                      */
  rcfetch = fetchobs(dsid,_currow_);
return;

putrow:
return;
```

## SCL ENTRY TO CREATE SELECTION LIST VIEW

A SAS/AF SCL entry applies all the current selection criteria to the "selection list view". This is the workhorse program of the application. It is called in the "init" section of the selection list programs and creates a view to which all prior subsetting criteria have been applied.

A very important item to note here is that the entire query, including the "where clause" is being submitted in pieces. From a programming standpoint, it is usually easier to build something like a "where clause" as a string and then to submit it as "&where" in the middle of a query. However, when giving the user of an application unlimited flexibility as to the number of variables on which to subset and the number of subsetting values, the 200 character limit on a string becomes a serious limitation. One could manipulate multiple strings and submit the "where clause" in pieces. Personally, I find it easier to just submit the pieces as I go.

### SUBVIEW.SCL

```
entry optional = passvar $ 8 ;
length selvar $ 8 selvrtyp $ 1 listid 8
       table $ 8 tblprior 8;

term:
  if passvar ^= ''
     then curwhere = ' and selvar ^= '''||passvar||'''';
     else curwhere = '';

  submit continue sql;
    create table sels as /* reads variable selections  */
    select selvar,       /* and checks to see if any   */
           selvrtyp,     /* subsetting has been done   */
           listid        /* (listid > 0)               */
    from vars            /* vars is created in         */
    where listid > 0     /* MAIN.PROGRAM above         */
      &curwhere
    ;
    quit;
  endsubmit;

  nsel = symgetn('sqlobs');  /* "sqlobs" will hold no. */
  dsid = open('sels','i');   /* of vars selected       */

  submit sql;
    create view list as  /* all rows will be selected  */
    select *             /* from selection list view   */
    from sugilist.all    /* if no subsetting is done   */
  endsubmit;

  if nsel > 0 then do;   /* if any subsetting, submit  */
    submit sql;  where  endsubmit; /* "where" in parts */
    /* process each var for which values will be subset*/
    /* using "sublist.scl" which is described next     */
    do i = 1 to nsel;
      call set(dsid);
      rcfetch = fetchobs(dsid,i);
      if selvrtyp = 'C' then method = 'inlistc';
                        else method = 'inlistn';
      if i > 1 then submit sql;  and  endsubmit;
      call method('sublist.scl',method,listid,selvar,rc);
      if rc ^= 0
        then put 'Error calling method for: ' selvar=;
    end;
  end;

  submit sql;  ;  quit;   endsubmit; /* finish the query */
* rcprev = preview('edit');          /* can verify query */
  submit continue sql; endsubmit;    /* submit the query */
  if dsid > 0 then rc = close(dsid);
return;
```

## METHOD TO SUBMIT WHERE CLAUSE "IN" LIST

A SAS/AF SCL entry contains methods used for building components of a "where clause". The specific components illustrated in this paper will be those which generate "in lists". For example, "COUNTRY IN('FRANCE', 'GERMANY', 'SPAIN')". The parameters passed to this program must be a list identifier (in which the subsetting values are stored), the variable name to be subset, and a return code variable. The "inlistc" method is used for building character "in lists" and the "inlistn" method is used for building numeric "in lists".

### SUBLIST.SCL

```
length _tvarc $ 200 _tvarn 8;
inlistc:
method _listid 8 _varname $ 17 _rc 8;
   _rc = 0;
   if itemtype(_listid,1) ^= 'C' then do;
     put 'ERROR: List must be character';
     put 'ERROR: type! No statements';
     put 'ERROR: will be submitted.';
     _rc = 4;
   end;
   else do;
     do i = 1 to listlen(_listid);
       _tvarc = getitemc(_listid,i);
       if i = 1 then do;
          submit sql;
            &_varname in ('&_tvarc'
          endsubmit;
```

```
      end;
      else do;
        submit sql;
          ,'&_tvarc'
        endsubmit;
      end;
    end;
    submit sql; ) endsubmit;
  end;
endmethod;

inlistn:
method _listid 8 _varname $ 17 _rc 8 ;
  _rc = 0;
  if itemtype(_listid,1) ^= 'N' then do;
    put 'ERROR: List must be numeric';
    put 'ERROR: type! No statements';
    put 'ERROR: will be submitted.';
    _rc = 4;
  end;
  else do;
    do i = 1 to listlen(_listid);
      _tvarn = getitemn(_listid,i);
      if i = 1 then do;
        submit sql;
          &_varname in (&_tvarn
        endsubmit;
      end;
      else do;
        submit sql;
          ,&_tvarn
        endsubmit;
      end;
    end;
    submit sql; ) endsubmit;
  end;
endmethod;
```

## SAMPLE DATA SETS USED

### DATA FROM CUSTOMER FILE - SELECTED OBS

| CUSTOMER | COUNTRY | STATE | CITY |
|---|---|---|---|
| 14324742 | USA | CA | SAN JOSE |
| 15432147 | USA | MI | KALAMAZOO |
| 29834248 | Britain | | LONDON, SW7 1PU |
| 39045213 | Brazil | | SAO PAULO |

| Variable | Type | Format | Label |
|---|---|---|---|
| ADDRESS | Char | $40. | STREETADDRESS |
| CITY | Char | $25. | CITY |
| CONTACT | Char | $30. | CONTACT |
| COUNTRY | Char | $20. | COUNTRY |
| CUSTOMER | Char | $8. | CUSTOMER |
| FIRSTORD | Num | DATE7. | FIRSTORDERDATE |
| NAME | Char | $60. | NAME |
| STATE | Char | $2. | STATE |
| TELEPHON | Char | $12. | TELEPHONE |
| ZIPCODE | Num | 11. | ZIPCODE |

### DATA FROM ORDERS FILE - SELECTED OBS

| SHIPTO | ORDERNUM | STOCKNUM |
|---|---|---|
| 14324742 | 11276 | 1279 |
| 14324742 | 11288 | 9870 |
| 14324742 | 12466 | 1279 |
| 14324742 | 12479 | 9870 |
| 15432147 | 11274 | 4789 |
| 15432147 | 11287 | 2567 |
| 15432147 | 12464 | 4789 |
| 15432147 | 12478 | 2567 |
| 29834248 | 11272 | 3478 |
| 29834248 | 11275 | 3478 |
| 29834248 | 12160 | 3478 |
| 29834248 | 12465 | 3478 |
| 39045213 | 11270 | 1279 |
| 39045213 | 11280 | 1279 |
| 39045213 | 12051 | 1279 |
| 39045213 | 12471 | 1279 |

| Variable | Type | Format | Label |
|---|---|---|---|
| FABCHRGS | Num | 17.2 | FABRICCHARGES |
| LENGTH | Num | 11. | LENGTH |
| ORDDATE | Num | DATE7. | DATEORDERED |
| ORDERNUM | Num | 11. | ORDERNUM |
| PROCBY | Num | 11. | PROCESSEDBY |
| SHIPPED | Num | DATE7. | SHIPPED |
| SHIPTO | Char | $8. | SHIPTO |
| SPECINST | Char | $200. | SPECIALINSTRUCTION |
| STOCKNUM | Num | 6. | STOCKNUM |
| TAKENBY | Num | 11. | TAKENBY |

### DATA FROM INVOICE FILE - SELECTED OBS

| BILLEDTO | INVOICE |
|---|---|
| 14324742 | 11276 |
| 15432147 | 11287 |
| 15432147 | 12478 |
| 39045213 | 11270 |
| 39045213 | 11280 |
| 39045213 | 12051 |
| 39045213 | 12471 |

| Variable | Type | Format | Label |
|---|---|---|---|
| AMOUNT | Num | 17.2 | AMTBILLED |
| AMTUS | Num | 17.2 | AMOUNTINUS |
| BILLEDBY | Num | 11. | BILLEDBY |
| BILLEDON | Num | DATE7. | BILLEDON |
| BILLEDTO | Char | $8. | BILLEDTO |
| COUNTRY | Char | $20. | COUNTRY |
| EXCHANGE | Num | TIME8. | COMPUTEDEXCHANGE |
| INVOICE | Num | 11. | INVOICENUM |
| PAIDON | Num | DATE7. | PAIDON |

### DATA FROM PRODUCT FILE

| PRODID | FIBNAME |
|---|---|
| 1279 | asbestos |
| 2356 | nylon |
| 2567 | fiberglass |
| 3456 | silk |
| 3478 | olefin |
| 4789 | dacron |
| 8934 | gold |
| 9678 | cotton |
| 9870 | polyester |

| Variable | Type | Format | Label |
|---|---|---|---|
| COST | Num | 11.2 | COST |
| FIBNAME | Char | $12. | FIBERNAME |
| FIBSIZE | Num | E12. | FIBERSIZE |
| PERUNIT | Char | $5. | PERUNIT |
| PRODID | Num | 6. | PRODUCTID |
| WEIGHT | Num | E12. | WEIGHT |
| WIDTH | Num | E12. | WIDTH |

**DATA VALUES STORED IN SUGILIST.PARMS**

| CATEGORY | TABLE | TBLPR IOR | SELVAR | SELVR TYP | SELVRDSC |
|---|---|---|---|---|---|
| Customer Info | CUSONLY | 1 | CUSTOMER | C | Customer |
| Customer Info | CUSONLY | 1 | COUNTRY | C | Country |
| Customer Info | CUSONLY | 1 | STATE | C | State |
| Customer Info | CUSONLY | 1 | CITY | C | City |
| Billing/Shipping | ORDRINV | 2 | ORDERNUM | N | Order No |
| Billing/Shipping | ORDRINV | 2 | INVOICE | N | Invoice No |
| Product Info | ALL | 3 | FIBNAME | C | Fiber Name |

| Variable | Type | Len | Pos |
|---|---|---|---|
| CATEGORY | Char | 20 | 0 |
| SELVAR | Char | 8 | 36 |
| SELVRDSC | Char | 20 | 45 |
| SELVRTYP | Char | 1 | 44 |
| TABLE | Char | 8 | 20 |
| TBLPRIOR | Num | 8 | 28 |

## SAMPLE APPLICATION SCREENS

When the initial MAIN.PROGRAM screen is displayed, "ALL" is the default selection for each variable in the parameter file.

```
Sample SUGI Reporting - Main Menu

    Tab to desired choice and press enter:


Invoice Number      ( INVOICE  )  All     Some
Order Number        ( ORDERNUM )  All     Some
City                ( CITY     )  All     Some
Country             ( COUNTRY  )  All     Some
Customer            ( CUSTOMER )  All     Some
State               ( STATE    )  All     Some
Fiber Name          ( FIBNAME  )  All     Some
```

If the user tabs to "SOME" for the country variable and presses the enter key, the following list is displayed. The list is shown the way it would appear after the user selects Argentina, Australia, Austria, and Canada.

```
***  Tab to desired choice and          ***
***  press <Enter> to select/deselect.  ***

          Argentina
          Australia
          Austria
          Belgium
          Brazil
          Britain
          Canada
          France
          Japan
          Netherlands
```

When the user returns to MAIN.PROGRAM, the country variable is indicated as having subset values. The user might then select "SOME" for the cities variable.

```
Sample SUGI Reporting - Main Menu

    Tab to desired choice and press enter:


Invoice Number      ( INVOICE  )  All     Some
Order Number        ( ORDERNUM )  All     Some
City                ( CITY     )  All     Some
Country             ( COUNTRY  )  All     Some
Customer            ( CUSTOMER )  All     Some
State               ( STATE    )  All     Some
Fiber Name          ( FIBNAME  )  All     Some
```

Only cities for the previously selected countries are available for selection.

```
***  Tab to desired choice and          ***
***  press <Enter> to select/deselect.  ***

          BUENOS AIRES
          PRAHRAN, VICTORIA
          VANCOUVER, V5T 1L2
          VIENNA
```

## CONCLUSION

If you are new to SAS/AF, these examples may look a little complicated at first, but don't get discouraged. Be sure to look up the functions you haven't used before so you understand what they do. Once you understand the functions and techniques used in these programs, you will have the foundation you need to go on to building your own flexible applications.