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 request 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 "datalist", 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.

```sql
proc sql;
/* only what's needed for */
/* customer selections */
create view SUGILIST.CUSONLY as
select a.*,
    b.ordernum,
    b.stocknum,
    b.invoice
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 sugivw.orders c
left join sugivw.invoice d
on c.shippto = d.billedto and
    c.ordernum = d.orderno;

/* contains vars necessary */
/* for all selections */
create view SUGILIST.ALL as
select a.*,
    b.fibname
from sugivw.orders c
left join sugivw.invoice d
on c.shippto = d.billedto and
    c.orderno = d.orderno;
```

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.
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 (GATT) 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." |

entry listid $ passvar $:

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
call display('subview.scl', passvar);
endsubmit;
quit;

/* End of Task A */

***** 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
call display('subview.scl', passvar);
endsubmit;
quit;

endsubmit;

/* End of Task A */

**** Display ****

| "Tab to desired choice and press <Enter> to select/deselect." |

entry listid $ 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
call display('subview.scl', passvar);
endsubmit;
quit;

/* End of Task A */

***** 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
call display('subview.scl', passvar);
endsubmit;
quit;

/* End of Task A */
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 "inif" section of the selection list 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.

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 term:

```
term:
  if passvar = 'I'
    then curwhere = '" and selvar = "("; passvar = ';'
    else curwhere = '';
  submit continue sql:
    create table _sels as /
      select curwhere
        from _sugilist
        where listid = 0
      create table sels
        as
          select selvar,
            from _sugilist
            where listid = 0
          create view list as
            select curwhere
              from _sugilist
              where listid > 0
            create table _sels
              as
                select selvar,
                  from _sugilist
                  where listid = 0
        end submit;
```
SAMPLE DATA SETS USED

DATA FROM CUSTOMER FILE - SELECTED OBS

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th>COUNTRY</th>
<th>STATE</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>14324742</td>
<td>USA</td>
<td>CA</td>
<td>SAN JOSE</td>
</tr>
<tr>
<td>15432147</td>
<td>USA</td>
<td>MI</td>
<td>KALAMAZOO</td>
</tr>
<tr>
<td>29834248</td>
<td>Britain</td>
<td>LONDON</td>
<td>SW7 1PU</td>
</tr>
<tr>
<td>39045213</td>
<td>Brazil</td>
<td></td>
<td>SAO PAULO</td>
</tr>
</tbody>
</table>

DATA FROM ORDERS FILE - SELECTED OBS

<table>
<thead>
<tr>
<th>SHIPTO</th>
<th>ORDERNUM</th>
<th>STOCKNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>15432147</td>
<td>12478</td>
<td>2567</td>
</tr>
<tr>
<td>15432147</td>
<td>1275</td>
<td>3478</td>
</tr>
<tr>
<td>15432147</td>
<td>12160</td>
<td>3478</td>
</tr>
<tr>
<td>15432147</td>
<td>12465</td>
<td>3478</td>
</tr>
<tr>
<td>15432147</td>
<td>11270</td>
<td>1279</td>
</tr>
<tr>
<td>15432147</td>
<td>11280</td>
<td>1279</td>
</tr>
<tr>
<td>15432147</td>
<td>12051</td>
<td>1279</td>
</tr>
<tr>
<td>15432147</td>
<td>12471</td>
<td>1279</td>
</tr>
</tbody>
</table>

DATA FROM INVOICE FILE - SELECTED OBS

<table>
<thead>
<tr>
<th>BILLTO</th>
<th>INVOICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>14324742</td>
<td>11276</td>
</tr>
<tr>
<td>15432147</td>
<td>11287</td>
</tr>
<tr>
<td>15432147</td>
<td>12478</td>
</tr>
<tr>
<td>39045213</td>
<td>11270</td>
</tr>
<tr>
<td>19045213</td>
<td>11280</td>
</tr>
<tr>
<td>19045213</td>
<td>12051</td>
</tr>
<tr>
<td>19045213</td>
<td>12471</td>
</tr>
</tbody>
</table>

DATA FROM PRODUCT FILE

<table>
<thead>
<tr>
<th>FIBRID</th>
<th>FIBNAME</th>
<th>COST</th>
<th>FIBSIZE</th>
<th>PERUNIT</th>
<th>WEIGHT</th>
<th>WIDTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1279</td>
<td>asbestos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2356</td>
<td>nylon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3456</td>
<td>fiberglass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3478</td>
<td>olefin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4789</td>
<td>dacron</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6934</td>
<td>silk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9870</td>
<td>cotton</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9870</td>
<td>polyester</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variable Type Format Label

<table>
<thead>
<tr>
<th>FIBRID</th>
<th>FIBNAME</th>
<th>COST</th>
<th>FIBSIZE</th>
<th>PERUNIT</th>
<th>WEIGHT</th>
<th>WIDTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1279</td>
<td>asbestos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2356</td>
<td>nylon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3456</td>
<td>fiberglass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3478</td>
<td>olefin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4789</td>
<td>dacron</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6934</td>
<td>silk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9870</td>
<td>cotton</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9870</td>
<td>polyester</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DATA VALUES STORED IN SUGILIST.PARMS

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>TABLE</th>
<th>SELVAR</th>
<th>SELVR-DSC</th>
<th>TBLPRIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Info</td>
<td>CUSONLY</td>
<td>1</td>
<td>CUSTOMER</td>
<td>C</td>
</tr>
<tr>
<td>Customer Info</td>
<td>CUSONLY</td>
<td>1</td>
<td>COUNTRY</td>
<td>C</td>
</tr>
<tr>
<td>Customer Info</td>
<td>CUSONLY</td>
<td>1</td>
<td>STATE</td>
<td>C</td>
</tr>
<tr>
<td>Customer Info</td>
<td>CUSONLY</td>
<td>1</td>
<td>CITY</td>
<td>C</td>
</tr>
<tr>
<td>Billing/Shipping</td>
<td>ORDINVR</td>
<td>2</td>
<td>ORDERNUM</td>
<td>N</td>
</tr>
<tr>
<td>Product Info</td>
<td>ALL</td>
<td>3</td>
<td>FIBNAME</td>
<td>C</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY</td>
<td>Char</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>SELVAR</td>
<td>Char</td>
<td>8</td>
<td>36</td>
</tr>
<tr>
<td>SELVR-DSC</td>
<td>Char</td>
<td>20</td>
<td>45</td>
</tr>
<tr>
<td>SELVRTYP</td>
<td>Char</td>
<td>1</td>
<td>44</td>
</tr>
<tr>
<td>TBLPRIOR</td>
<td>Num</td>
<td>8</td>
<td>28</td>
</tr>
</tbody>
</table>

 Samp le SUGI Reporting - Main Menu

Tab to desired choice and press enter:

<table>
<thead>
<tr>
<th>Invoice Number</th>
<th>ORDINVR</th>
<th>CITY</th>
<th>COUNTRY</th>
<th>CUSTOMER</th>
<th>STATE</th>
<th>FIBNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVOICE</td>
<td>All</td>
<td>Some</td>
<td>All</td>
<td>Some</td>
<td>Some</td>
<td>Some</td>
</tr>
</tbody>
</table>

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

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.

SAS, SAS/ACCESS, and SAS/AF 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.