SAS/AF Frames: List Objects

Serge Dupuis, BKD Software Consultants
Loretta Golby, ISM Alberta
Edmonton, Alberta, Canada

Abstract

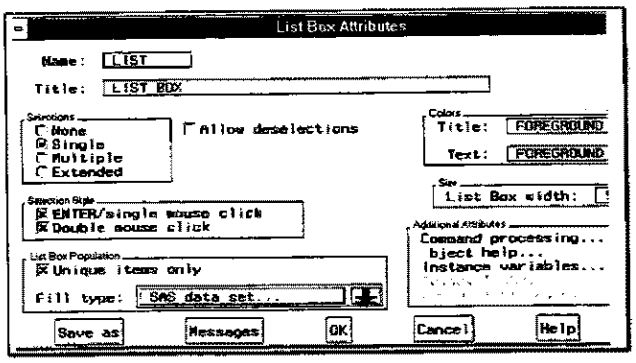
This tutorial will cover the use of simple and complex selection list objects in SAS/AF® Frame applications. Older SAS/AF programs based on program entries using DATALISTC and VARLISTC functions will be updated with equivalent GUI objects available in SAS/AF Frames.

Introduction

List box objects found in SAS/AF Frames offer the developer a method of building applications that users will find very intuitive in running their SAS® applications. Applications built with user selections tend to run without data entry errors. This tutorial will show what can be done with these list boxes. All examples use either of the SAS data sets, SASUSER.HOUSES or SASUSER.CLASS, copied to the SUGI21 data library.

List box attributes

When you create a list box in a SAS/AF Frame, a list box attributes window defines the behavior of the list box. Some attributes can be changed with specific methods available to the list box object. For example, the data used to fill the list box can be changed with a Screen Control Language program. This is documented in the SAS/AF Frames reference manual.



Of particular interest for this tutorial are the following attributes:

- Selections (none,single, multiple,extended)
- Fill type, including
  - SCL list
  - SAS data set
  - SAS data set id

Data set fill type

In this example, the data set SUGI21.HOUSES is used to fill the list box. When you choose SAS data set, you are also asked for which variable to use in that data set.

You can create a SAS/AF Frame with the Build command:

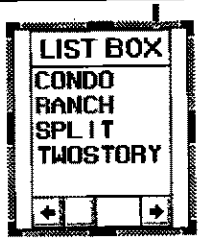
```
BUILD c=SUGI21.appl.list0.Frame
```

When you build this Frame, you will need to create a list box by choosing List Box Object from the **Actions..Make** pull down menu. When the list box is positioned on Frame, the attribute window will be displayed.

The attributes for this first object should be set as follows:

- Name: LIST
- Selections: SINGLE
- Unique items only check ON
- Fill type: SAS data set SUGI21.HOUSES, variable: STYLE

When the attributes window is closed, the Frame should display the list box with the values of STYLE:



To use the selections in a program, Screen Control Language (SCL) is used. In this example, the user clicks on a STYLE and a SQL submit block lists the data.

```
length val $8;
list:
     call notify('list',
                 '_GET_LAST_SEL_',
                 row,issel,val);
        submit sql continue;
         select * from sugi21.houses
         where style="&val";
     quit;
     endsubmit;
return;
```

This example used the same dataset for populating the list box and SQL. A smaller lookup data set would normally be preferred for the list box.

CALL NOTIFY is documented in the SAS/AF Frames reference. In this example, it is used to get the one selected value of style, which is passed on to the submit block. Since the value returned(VAL) is character, the variable must be previously declared as a character type in the LENGTH statement.

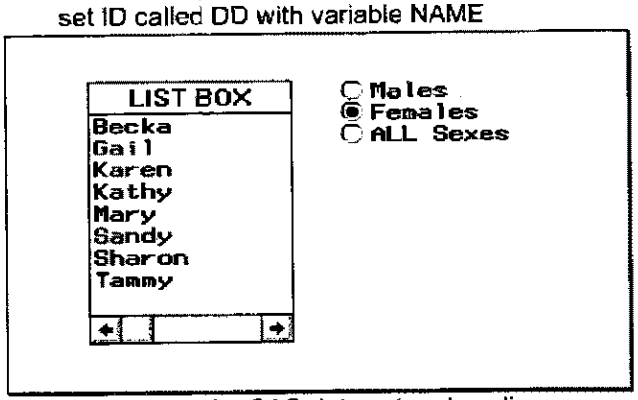
Data set ID fill type

In this example, the list box attributes are set to get data from a SAS data set id called DD. The advantages over a simple SAS data set reference is that you can use SCL to fill the content of the list box. To start, use the BUILD command:

```
BUILD C=SUGI21.APPL.LIST1.Frame
```

Two objects are created in this Frame:

- a radio button(SEX) with output 'M', 'F' or 'ALL'
- a list box object called LIST1 with fill type Data set ID called DD with variable NAME



The SCL opens the SAS data set and applies a WHERE clause with the settings of the radio button.

```
LENGTH WW $30;
init:
  dd=open('sugi21.class');
return;
sex:
 RC=WHERE(DD);
 IF SEX NE 'ALL' THEN DO;
            WW= 'SEX="' !! SEX !! '"';
            RC=WHERE(DD,WW);
            END;
 CALL NOTIFY('LIST1','_UPDATE_');
RETURN;
term:
ss=close(dd);
return;
```

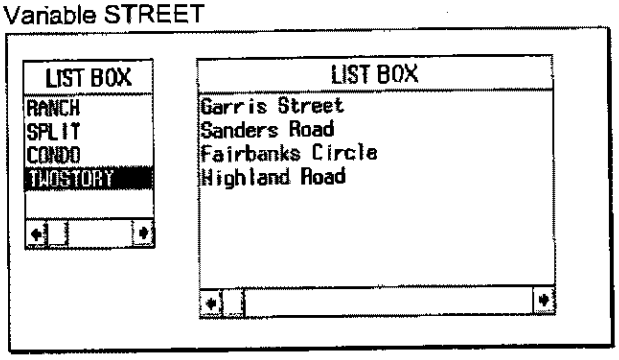
Combining two list boxes

In the following example, one list box is used to select STYLE of houses. This selection is used to subset the data set for populating a second list box. This example was used to replace two extended tables with several SCL lines of code. To start, use the BUILD command:

```
BUILD C=SUGI21.APPL.LIST2.Frame
```

Build the Frame with two objects
List1:
Selection=single
Fill type: SAS data set SUGI21.HOUSES
Variable STYLE (unique values)

List2:
Selection=none
Fill type: SAS data set ID DD
Variable STREET



```
LENGTH WW $30 val $10;
init:
   dd=open('sugi21.houses');
return;
LIST1:
  call notify
  ('list1','_GET_LAST_SEL_',row,issel,val);
  RC=WHERE(DD);     *RESET;
  IF VAL NE '' THEN DO;
       WW= 'style="' !! val !! '"';
       RC=WHERE(DD,WW);
 END;
 CALL NOTIFY('LIST2','_UPDATE_');
RETURN;
term:
dd=close(dd);
return;
```

The LIST1 block reads the user selection with a _GET_LAST_SEL_ method on object LIST1. The value returned(VAL) is character and must be previously declared in the LENGTH statement. The value of VAL is used to create a where clause in variable WW. Single and double quotes must balance!

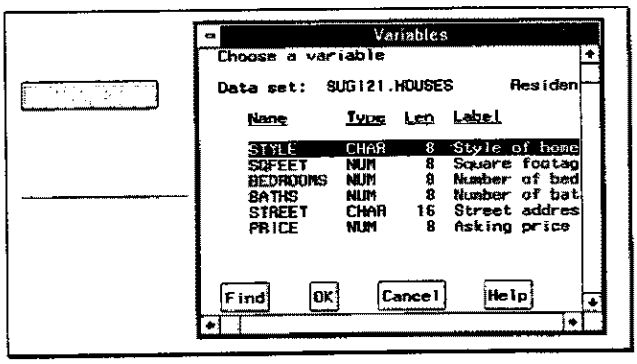
The WHERE function subsets the dataset referenced by DD, which is then passed to the second list box.

VARLIST function

This SCL function presents a selection list of variable names, type, labels given a specified data set ID. The function is used extensively in program entries, but is very much character-like in appearance. It opens on top of the calling program display.
This example shows a button called VARLIST activating a VARLIST window.

```
init:
  dd=open('sugi21.houses');
return;

varlist:
  select=varlist(dd,' ',2,'Choose a
       variable','','y');
return;

term:
  if dd then rc=close(dd);
return;
```

When the user clicks on the VARLIST button, the VARLIST function displays a selection of variables



A GUI VARLIST: Update to Frames

The VARLIST function can be replaced with a list object to present the user with a more GUI compliant application. In this first example, a crude Frames-based VARLIST is attempted.
Create a SAS/AF Frame entry with these objects:

Button called VARLIST
Button called OK
Text entry called SELECT
Text label displaying "SELECTED:"
List box called LIST1 with:

- Multiple selections
- Fill type SCL list name VLIST

The position of the OK button should be below the list box, in a same way as the OK button is in the VARLIST function window.

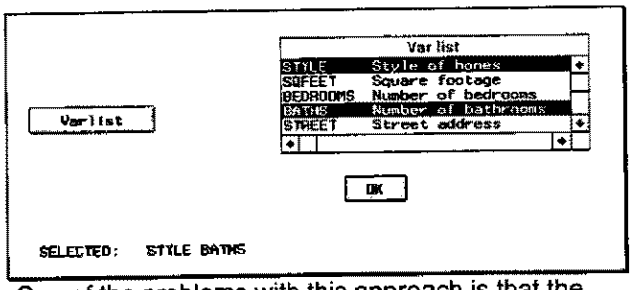
A number of aspects have to be covered:

- The variable names and labels of any given SAS data set can be found in the SASHELP.VCOLUMN view, or a temporary dataset can be created from a SQL or PROC CONTENTS block.

- The list box will show only one text string, but you need to show at least two columns, Variable NAME and LABEL. This is done by creating a new SCL variable called XX. Since variable names can be 1 to 8 characters, some string manipulation is done to present uniform column width.

```
length xx $30 name $8 label $25 fill $10;
init:
   vlist=makelist();
   dd=open('SASHELP.VCOLUMN');
   rc=where(dd,'libname="SUGI21" and
        memname="HOUSES"');
return;
varlist:
   vlist=makelist();
   * read dd dsn;
   vname=varnum(dd,'name') ;
   vlabel=varnum(dd,'label') ;
   rc=0;
     do until(rc ne 0);
        rc=fetch(dd);
        if rc=0 then do;
        name=getvarc(dd,vname);
        label=getvarc(dd,vlabel);
        L=9-length(name);
        fill=repeat(' ',L);
        xx= name || fill   || label;
        vlist=insertc(vlist,xx,-1);
        end;
     end;
call notify('list1','_update_');
return;
OK:
  * get the selections from the list;
  call notify('list1','_get_nselect_',numsel);
  if numsel > 0 then do;
   call notify('list1','_get_value_',boxid);
   selid=getniteml(boxid,'text');
   do i=1 to numsel;
     select=select || getitemc(selid,i);
   end;
  end;
return;
```

When the Frame is tested, a list box populated with variable names and labels is shown.



One of the problems with this approach is that the list box needs to use space on the calling frame. It can appear with _UNHIDE_ method, but it will not display 'on top' of the calling frame as VARLIST function would. This will be discussed in a later section.

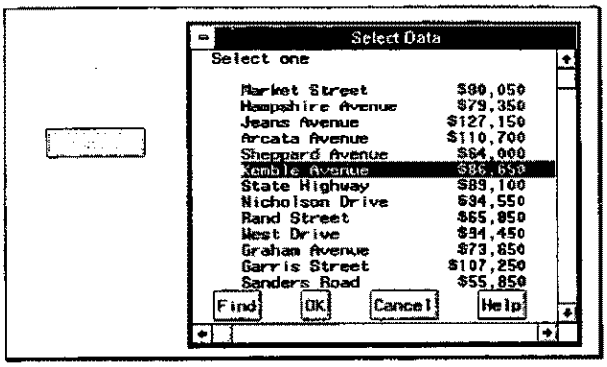
DATALISTC function

This function will display variable values to be selected, given a dataset ID. Unlike a list box it can display more than one column, but it returns only the value of the first column on selection.

In this example, a DATALISTC box displays values of STREET and PRICE. Create a Frame with push-button object called START, and write the following SCL:

```
init:
   dd=open('sugi21.houses');
return;

start:
        select=datalistc(dd,'street
                price bedrooms','Select one');
return;

term:
  if dd then rc=close(dd);
return;
```

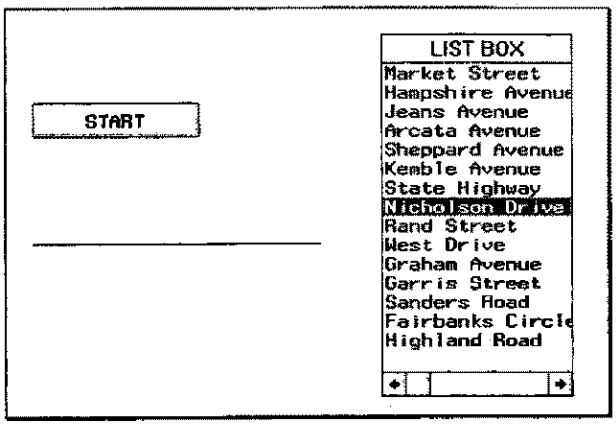
When the user clicks on button Start, the DATALISTC function displays available values of STREET and PRICE



On character based terminals the DATALISTC function works very well, but users are more demanding in windowing environments. One of the neat features of GUI compliant list boxes is the ability to get down a selection list by typing only the first character. For example, if your list box contains hundreds of entries, in alphabetical order, the user can type 'S" to get to the SMITHS without repeatedly pressing on the DOWN key. This cannot be done in character-based DATALISTC function.
In this example, build a Frame with two objects:

- A push-button called START
- A list box called LISTBOX with fill type of data set id, with id of DD and variable of STREET

```
init:
dd=open('sugi21.houses');
   call notify('listbox','_hide_')  ;
return;

start:
   call notify('listbox','_unhide_') ;
   cursor listbox;
return;

listbox:
   call notify('listbox',
          '_GET_LAST_SEL_',row,issel,val);
   call notify('listbox','_hide_')  ;
return;

term:

dd=close(dd);


return;
```



Calling list boxes in other Frames

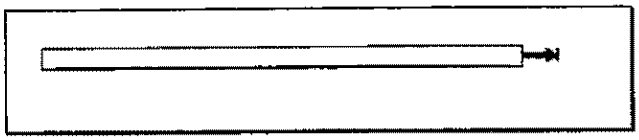
In the previous two examples, the list box objects were contained on the same Frame as the button object that called the list boxes. You can initially hide the list box and use the _UNHIDE_ method as needed, but the list box will still be using space on your Frame. What is needed is a way to have a list box appear 'on top' of the calling Frame, in much the same way as DATALISTC and VARLIST functions do.

In the following example, a main program called MAIN.Frame calls a GETLIST.Frame program. Passing parameters between these Frames can be done by lists, and a number of these were defined to make that application work.

The Main.Frame entry contains two objects:

- Text entry called VIEWBOX to display values selected
- Control object called GETAGE

The display for this MAIN is simple:



The SCL for the Main.Frame entry has three main parts:

- INIT section to initialize SCL lists. Global lists are used because data is passed between Frames. Three lists are used, a master containing two sublists, one for calling the next Frame, another to return values to this MAIN.
- GETAGE section that calls the Frame containing the list box. This will appear 'on top' of the calling program display.
- RETURNED section to process the list containing the selections in the called Frame

INIT section

```
init:
 master=makelist(0,'G');
* sublists  to master;
 listage=makelist(0,'G');
* attach the sublists to the master;
rc=insertl(master,listage,-1,'List of ages')    ;
request=makelist(0,'G');
  rc=insertc(request,'sugi21.class',-
       1,'Datasetname');
  rc=insertc(request,'age',-1,'variablename');
  rc=insertc(request,'listage',-1,'listname');
Retlist=makelist(0,'G');
* create a environment list to return data ;
  envlist=envlist('G');
  rc=insertl(envlist,master,-1,'Masterlist');
  rc=insertl(envlist,request,-1,'Requestlist');
  rc=insertl(envlist,retlist,-1,'Retlist');
return;
```

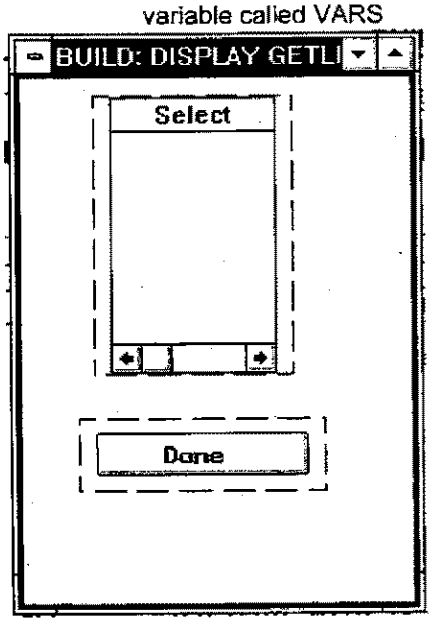
GETAGE and RETURNED sections

```
getage:
   Call display('GETLIST.Frame');
   link returned;
return;
returned:
   retlist=getniteml(envlist,'Retlist');
    selected=makelist();
    selected=getniteml(retlist,'text');
     n=listlen(selected) ;
    * seldist='';
     viewbox='';
     do i=1 to n;
     if viewbox='' then
        viewbox=getitemc(selected,i);
         else  viewbox= viewbox || ',' ||
               getitemc(selected,i);
     end;
return;
term:
     rc=dellist(envlist,"Y");
return;
```

Returned values are put into a string with commas between. This facilitates further processing into a Where clause such as where age in(1,2,3) ;

The next Frame contains two objects:

- A push-button called DONE to end the Frame
- A list box called 'listbox' with the attributes:
  - Multiple selections
  - fill type: **data set id** named DD with variable called VARS



This Frame is sized smaller than full screen and offset to the right similar to the position of a DATALISTC display. This will appear to the user as a display on top of the main program Frame.
The SCL for this Frame has two main parts:

- INIT section to read the passed SCL lists and populate the list box according to information created in the calling(MAIN) program.
- LISTBOX section to read selections from the list box and updated SCL list to be passed back to the calling program (MAIN.FRAME)

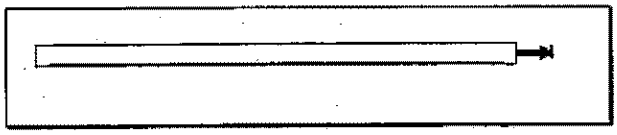
INIT Section

```
init:
    envlist=envlist('G');
    master=makelist(0,'G');
    request=makelist(0,'G');
    retlist=makelist(0,'G');

    master=getniteml(envlist,'masterlist');
    request=getniteml(envlist,'requestlist');
    retlist=getniteml(envlist,'retlist');

    dsn=getnitemc(request,'datasetname');
    varname=getnitemc(request,'variablename');
    listname=getnitemc(request,'listname');

  * set up data to populate list window;
    dd=open(dsn);
    values=makelist(0,'G');
    num=0;
    rc=lvarlevel(dd,varname,num,values);
    rc=sortlist(values);
    dd=close(dd);

return;
```

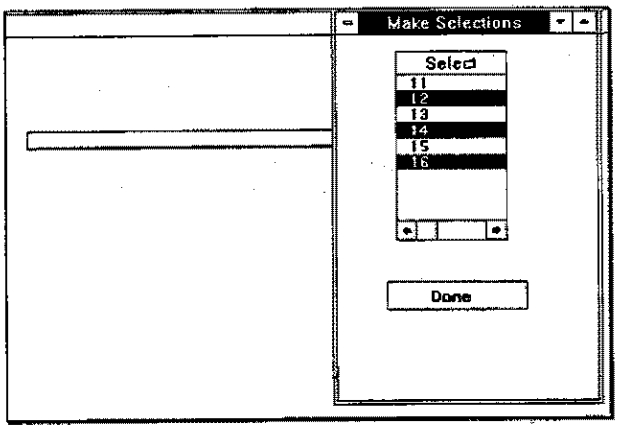
LISTBOX section

```
listbox:
call notify('listbox','_GET_VALUE_',selectlist);
 SELECTL=MAKELIST(0,'G') ;
 selectl=copylist(selectlist,'Y',selectl);

  *store the selections;
   rc=setniteml(envlist,selectl,'Retlist');
return;

DONE:
RETURN;
```

The MAIN program displays an empty text entry field, with a control object implying (to the user) that more can be obtained.

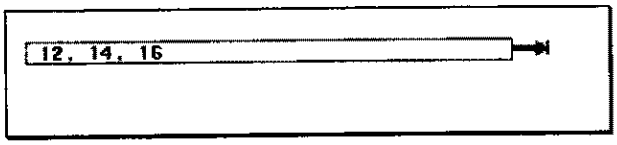


When the user presses on the control object, the second Frame(GETLIST) appears on top of the calling program. The use of smaller size for GETLIST is similar to that of the DATALISTC function.



When the user presses DONE on the selection list Frame, the Frame ends and the user is returned to the MAIN program.

The MAIN program then displays the user selection



Conclusion

Use of list objects in Frames can be useful in building selection lists of various types. They can be used to replace DATALISTC and VARLISTC SCL functions, but do require additional SCL. Further development in building custom objects and methods will enhance and simplify the amount of SCL required.

References

SAS/AF Software: Frame Entry Usage and Reference, Version 6, First Edition

SAS Screen Control Language: Reference, Version 6, Second Edition

SAS/AF Software: Frame Class Dictionary, Version 6, First Edition

Acknowledgments

The authors would like to thank Bill Benson of ISM Alberta for comments and preparation of this paper.

Please address comments or questions to:

Serge Dupuis
Internet: 75032.2240@compuserve.com

Loretta Golby
Internet: lgolby1@isma.com