Using PROC SQL Dictionary Tables to Generate Lists for SAS/AF Extended Tables

David Freese, Fireman's Fund Insurance Co's., Novato, CA

ABSTRACT

SAS/AF® software provides the capability of easily retrieving information about SAS® libraries, directories, catalogs and datasets. Starting with SAS 6.07, PROC SQL provided support for a series of new functions. Using SAS data sets generated from various PROC SQL dictionary tables you can quickly fill and customize extended tables which display data-driven selections. This increases programming flexibility, uses current information to drive user selections, and reduces the time and effort to adjust or enhance SAS/AF and Screen Control Language® (SCL) source code. For instance, if you store a set of PROC REPORT templates in a catalog and you want the user to be able to choose from among the various reports, you can retrieve all the information about this catalog (i.e. libname, catalog, report name, extension, label, date created, etc.) by using the SQL procedure dictionary table. Variables in the data set can then be displayed as a selection list in a SAS/AF® extended table or processed like any other SAS dataset.

INTRODUCTION

One of the objectives of software development is to discover and use techniques which enhance an application's flexibility in meeting user requests without programming intervention. Using actual data sets and meta-data sets to drive an application's presentation of information has become very popular in this respect. In applications developed through SAS/AF utilizing SCL, data descriptors associated with the LIBNAMES and FILENAMES allocated during the session can be accessed and managed for the same purpose. The various SCL lists (i.e. LIBLIST, VARLIST, DIRLIST, etc.) take advantage of this technique, reducing development time and making programming code more generalized.

The technique I would like to describe provides the same application flexibility, parameter passing and decreased maintenance time. In addition it provides the ability for customizing your screens using AF extended table logic. This can be very helpful when trying to create an application with a consistent look and feel, as well as use PMENUs for screen navigation.

USING PROC SQL DICTIONARY TABLES

One of new functions in the SQL procedure is the capability of selecting data from among a variety of dictionary tables. Dictionary tables hold information about your SAS session (i.e. allocated libnames, catalogs, datasets, external files, indexes, tables, views and system options). When used in a PROC SQL select statement this information is available in a table, view or dataset and can be, subsequently, used to fill values in a SAS AF extended table.

The example below demonstrates how the ***dictionary.catalogs*** descriptor is used to retrieve data about the members of specific SAS catalogs. Figure 1 lists the variables available to the AF program once the SQL procedure runs.

FIGURE 1 - Dictionary.Catalogs

```
DICTIONARY.CATALOGS

"Columns specific to MEMTYPE=CATALOG"
(
libname  char(8)   label='Library Name',
memname  char(8)   label='Member Name',
memtype  char(8)   label='Member Type',
objname  char(8)   label='Object Name',
objtype  char(8)   label='Object Type',
objdesc  char(40)  label='Object Description',
modified char(8)   label='Date Modified',
alias    char(8)   label='Object Alias'
 );
```

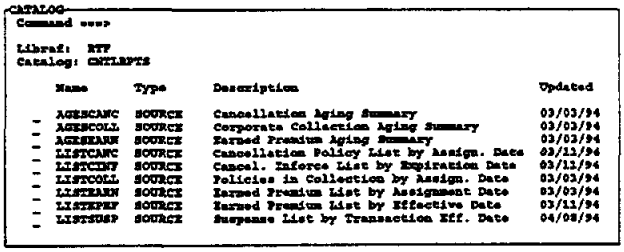
SAS/AF and SCL allow SQL submit blocks within the programs source code and, thus, easy reference to the SAS variables retrieved from such a process. In addition, SCL screen variables can be linked to the same SAS variables.

STORED PROGRAMS OR REPORT TEMPLATES

SAS source code or PROC REPORT templates are often stored in SAS catalogs to be used for future processing or by multiple users. The information about each entry in the catalog is available throughout the SAS session and can be managed using various SAS AF or SCL techniques.

Figure 2 shows the Catalog Window of the specific SAS Catalog that will be used in the example below.

FIGURE 2 - Stored Programs



Some of these programs will be used during the AF session.

SOURCE CODE

This technique shown in Figure 3 uses an SQL submit block in the INIT section to generate a table (or SAS data set) holding data about a specific SAS catalog of stored source code. Information about this catalog is retrieved using the dictionary table option of the SQL code.

FIGURE 3 - SCL Source Code

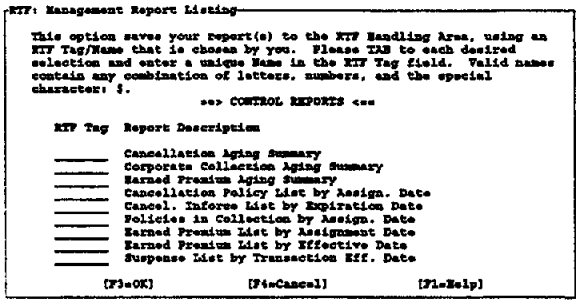
```
init:
   control always ;
   rpttype= symgetc('rpttype');
   lib = 'RTF' ;
   cat = trimn(rpttype)||'RPTS';
   typ = 'SOURCE';
submit sql continue ;
  create table lookout as
  select *
  from dictionary.catalogs
       where libname = "&lib"
         and memname = "&cat"
         and objtype = "&typ" ;
  alter table lookout
       modify luctag char(7) ;
  quit;
endsubmit ;

  outlist = open('work.lookout','U');
  call set(outlist) ;
  numobs = attrn(outlist,'NOBS');
  call setrow(0,numobs,'N','Y');
return;
```

Notice that the SCL variable "RPTTYPE" is assigned using the SYMGETC function call to a macro variable of the same name. The report type was previously specified by the end user, and here it provides a focusing tool to identify which specific catalog to select in the SQL submit block.

In addition, the SQL procedure is using the ALTER command to create a new variable on the table (dataset), which will be used to hold the value of a user-created name. The entire AF program can be found at the end of this article.

The result of the INIT section of this extended table program can be seen in Figure 4.

FIGURE 4 - Resulting AF Extended Table



If a new entry is added to or deleted from the original catalog (shown in Figure 2), or an existing entry's name or description is changed, it will automatically reflect the change in the resulting extended table and the SCL program running behind the scenes.

In addition, this extended table allows the user to enter a name or tag of their choice for each of the selections they want to process, allowing greater flexibility and customization. There is, also, enhanced control over placement of such things as the text, screen variables and the PMENU's.

REFERENCES / ACKNOWLEDGEMENTS

SAS Institute Inc. (1991), *SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07*, 286-291, Cary, NC: SAS Institute Inc.

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

AUTHOR CONTACT INFORMATION

David Freese
Supervisor/Lead Business Programmer Analyst
Management Information, Commercial Division
Fireman's Fund Insurance
777 San Marin Drive
Novato, CA 94998
TEL: (415) 899-3056, E-mail: usffckk3@ibmmail.com

APPENDIX - Entire SCL Program

```
ENTRY: RPTFRNT.PROGRAM   REPORTS: Selection of Reports to Store

***** DISPLAY *****

+-----------------------------------------------------------------------+
|  This option saves your report(s) to the RTF Handling Area, using an  |
|  RTF Tag/Name that is chosen by you.  Please TAB to each desired      |
|  selection and enter a unique Name in the RTF Tag field.  Valid names |
|  contain any combination of letters, numbers, and the special         |
|  character: $.                                                        |
|                                                                       |
|                &REPORTS_____________________________                  |
|     RTF Tag  Report Description                                       |
|~~~                                                                    |
|     &LUCTAG  &OBJDESC_____________________________________ &OBJNAME  |
+-----------------------------------------------------------------------+

  ***** GENERAL ATTRIBUTES *****

     Window name: RTF: Management Report Listing
       Start row: 1          col: 1
  Number of rows: 24       cols: 80        Banner: NONE

    General help: REPORT.CBT
   Function keys: RPTLIST
 Lookup data set:
    Command menu: OK3
     Prompt char: ?
 System Options:  EXTENDED TABLE
          Parent:
     Window Type: DIALOG
     Scroll Bars:


  ***** ATTRIBUTES *****

-------------------------------------------------------------------------
  Field name:  REPORTS    Frame: 1    Row: 7       Col: 17      Length: 40
       Alias:  REPORTS      Choice group:                 Pad:
        Type:  CHAR                Protect: YES
      Format:                       Just: CENTER
    Informat:
 Error color:  RED          attr:  REVERSE        Help:
        List:
     Initial:
     Replace:
     Options: CAPS AUTOSKIP
-------------------------------------------------------------------------
  Field name:  LUCTAG     Frame: 2    Row: 1       Col: 6       Length: 7
       Alias:  LUCTAG       Choice group:                 Pad: _
        Type:  CHAR                Protect: NO
      Format:                       Just: LEFT
    Informat:
 Error color:  RED          attr:  REVERSE        Help:
        List:
     Initial:
     Replace:
     Options: CAPS CURSOR AUTOSKIP
-------------------------------------------------------------------------
  Field name:  OBJDESC    Frame: 2    Row: 1       Col: 15      Length: 43
       Alias:  OBJDESC      Choice group:                 Pad:
        Type:  CHAR                Protect: YES
      Format:                       Just: LEFT
    Informat:
 Error color:  YELLOW       attr:  REVERSE        Help:
        List:
     Initial:
     Replace:
     Options: AUTOSKIP
-------------------------------------------------------------------------
  Field name:  OBJNAME    Frame: 2    Row: 1       Col: 59      Length: 8
       Alias:  OBJNAME      Choice group:                 Pad:
        Type:  CHAR                Protect: YES
      Format:                       Just: LEFT
    Informat:
 Error color:  BLUE         attr:  NONE           Help:
        List:
     Initial:
     Replace:
     Options: AUTOSKIP NON-DISPLAY
-------------------------------------------------------------------------
```

```
  ***** SOURCE *****

length rpttype $4 ;
length rc 8 loclvl $1 bchr $ 1 prtlist $90 taglist $80;

init:
control ALWAYS ;
rpttype= symgetc('rpttype');
loclvl = symgetc('loclvl');

select(rpttype) ;
  WHEN('SUMM') do; hlp=4; REPORTS='==> Summary Reports <==';        end;
  WHEN('DETL') do; hlp=5; REPORTS='==> Detail Reports <==';         end;
  WHEN('ALRT') do; hlp=6; REPORTS='==> Alert Reports <==';          end;
  WHEN('CNTL') do; hlp=7; REPORTS='==> Control Reports <==';        end;
  WHEN('TREN') do; hlp=8; REPORTS='==> Trend Reports <==';          end;
  WHEN('ANAL') do; hlp=9; REPORTS='==> Analysis Reports <==';       end;
  WHEN('TDCR') do; hlp=11; REPORTS='==> Time Driven Collections <==';end;
  WHEN('AUDT') do; hlp=13; REPORTS='==> Audit Receivables Rpts <==';end;
  otherwise    do; hlp=4; REPORTS='==> Summary Reports <==';        end;
  end;

lib = 'RTF' ;
cat = trimn(rpttype)||'RPTS';
typ = 'SOURCE';

submit sql continue ;
create table lookout as
select *
from dictionary.catalogs
    where libname = "&lib"
      and memname = "&cat"
      and objtype = "&typ" ;
alter table lookout
  modify luctag char(7) ;
quit;
endsubmit ;

halt = 'N';
outlist = open('work.lookout','U');
call set(outlist) ;
numobs = attrn(outlist,'NOBS');
call setrow(0,numobs,'N','Y');
return;

main:
cmdword=upcase(word(1));
if cmdword=='' ' then do;
   call nextword();
   if lookupc(cmdword,"TEACH") = 1 then
   call cbt('report',hlp);
end;
if _status_ = 'C' then do;
   _status_ = 'N' ;
   halt = 'Y';
   return;
end;
return;

term:
if _status_ = 'N' or
    halt    = 'Y' or
   _status_ = 'C' then do;
   rc=close(outlist) ;
   return;
   end;
prtlist = ' ';
taglist = ' ';
tagcnt  = 0;
do i = 1 to numobs;
   rc=fetchobs(outlist,i);
   if luctag gt ' ' then do;
      tagcnt + 1;
      prtlist = trimn(prtlist) || ' ' || objname ;
      taglist = trimn(taglist) || ' ' || luctag ;
      what = 'F'||substr(objname,1,7);
      call method('code.scl','userlog',what);
      end;
end;

if tagcnt = 0 then do;
   halt = 'Y';
   rc=close(outlist) ;
   return ;
   end;
call symputc('prtlist',prtlist);
call symputc('taglist',taglist);
call symputn('tagcnt',tagcnt);
call display('rptfile.program',rpttype,loclvl,taglist,prtlist,tagcnt) ;
rc=close(outlist) ;
return;

getrow:
if (fetchobs(outlist,_currow_)) = -1 then call endtable();
return;

putrow:
   if luctag ne ' ' then link chktag;
   if v then return;

   rc = fetchobs(outlist,_currow_,'NOSET');
   rc = updata(outlist) ;
   rc=unselect(_currow_);
return;

chktag:
rc=FIELD('ERROROFF','luctag');
if luctag=_blank_
then
  do;
    _msg_='You must enter a name';
    rc=FIELD('ERRORON','luctag');
    return;
  end;
v = verify(luctag,'ABCDEFGHIJKLMNOPQRSTUVWXYZ','0123456789','$');
if v
then
  do;
    bchr=substr(luctag,v,1);
    if bchr=' ' then
    _msg_='Invalid character ==> blank';
    else _msg_='Invalid character - > '||bchr||' in TAG "'||luctag||'".';
    rc=FIELD('ERRORON','luctag');
  end;
else do;
       rc=FIELD('ERROROFF','luctag');
  end;
return;
```