Techniques for Sharing Screen Control Language Programs in a SAS/AF® Application

Derek Drummond, Jeff Phillips, and Veronica Walgamotte
ARC Professional Services Group

Introduction

During application development, we often need to use identical Screen Control Language (SCL) statements in multiple locations of a program or even multiple programs. The most straightforward way to accomplish this is to replicate statements and place copies where needed. However, if modifications need to be made, all copies may need to be corrected; this can be a maintenance nightmare.

A better solution is to make one copy of the statements and share that copy. This way there is only one copy to maintain. Many programming languages provide tools for users to create their own subroutines and subfunctions to hold shared programming statements. The statements in the subroutines or subfunctions can be used by one or more programs with a simple reference to the routine. The question is "How can we best share code in an SAS/AF® or SAS/FSP® application?"

Version 6.07 of the SAS software, Screen Control Language provides three ways to use shared programming statements in SAS/AF applications: link, macro and method blocks. This paper investigates the advantages and disadvantages of using these ways to develop SCL statements that can be shared.

Link Blocks

A link block is a group of SCL statements that begins with a label (example, 'CHECKREC:'), end with an unconditional RETURN statement and is invoked by a LINK statement.

Positive Aspects:

1. can contain any executable SCL statement
2. can be invoked by another link block within the same program
3. can contain another link block
4. can be used to move code out-of-line and thus, break-up the program into smaller (hopefully functional) sections
5. can be used with SAS/FSP products
6. can include SUBMIT blocks
7. can contain macro blocks
8. can be placed in macro blocks
9. can contain statements to read, write or update screen variables
10. has access to any variable created and maintained by the program (except those variables contained in method blocks)
11. all variables created and maintained in the link block can be accessed by any other section of the program (except method blocks)

Negative Aspects:

1. can not be referenced from another program entry
2. out-of-line may be more difficult to understand
3. link blocks of SAS/AF program entries can be maintained only by the BUILD Facility (invoked by the BUILD command, PROC BUILD step, or the EDIT/SELECT command while on the CATALOG window of a SAS/AF catalog)
4. link blocks of SAS/FSP programs can be maintained only by the SAS/FSP Editing Facility (FSEDIT or FSVIEW Edit Mod)
5. nested link blocks can not be part of a conditional block of statements
6. nested link blocks terminate via the same RETURN statement

Link Example:

```
      Init:
        libref = 'DATABASE';
        disp = 'SHR';
        link alloclib;
             :
        sysrc = libname('database');
      Return;
      Main:
        libref = 'DATABASE';
        disp = 'OLD';
        link alloclib;
      Return;
      Term: Return;
      Alloclib:
        /* Parameters:
           LIBREF Name of SAS library
                  to be allocated
           DISP   OLD or SHR  */
        failure = 1;
        do while (failure);
          _msg_ =
'Attempting to allocate ' || libref;
          refresh;
          sysrc =
      libname(trim(symget('prefix'))||
      libref,'v607','disp="'||
      disp||'"');
          if sysrc = 0 then failure = 0;
          if failure then do;
            _msg_ =
'Allocation failure.    Waiting  20
seconds'||
' before retrying';
            refresh;
            start = time();
            do i = start to start + 20;
            end; /* delay loop */
          end;
        end;
      Return;
```

Macro Blocks

A SCL macro block is a group of macro statements that will generate SCL statements.

Positive Aspects:

1. can contain macro statements to conditionally generate any SCL statement (including executable and/or non-executable statements)
2. generated SCL statements will be placed in-line
3. can be used with SAS/FSP products
4. can generate statements to read, write or update screen variables
5. can generate SUBMIT blocks
6. can be "shared" by multiple SCL programs, SAS/FSP programs and base SAS programs
7. can be placed in an external file (e.g. a PDS) and maintained outside the SAS environment
8. can contain link blocks
9. can be part of link blocks
10. can contain another macro block
11. can contain method blocks
12. can be part of method blocks

Negative Aspects:

1. SCL statements are generated during compilation and not during the execution of the SCL program
2. modifying the macro requires programs to be re-compiled (Note: SAS/FSP programs must be compiled one at a time - SAS/AF programs can be compiled as a group by using the COMPILE statement of a PROC BUILD step)
3. the macro must be re-compiled before the program can be re-compiled (this can be a difficult task, often requiring extensive knowledge of the macro facility, system options associated with macros and the SASMACR catalog)
4. macro code is out-of-program and will be more difficult to understand
5. can not view the SCL code generated by macro block
6. messages about errors involving the SCL statements generated by a macro reference the line number at the location of the macro prior to macro substitutions
7. DEBUG can not reference SCL statements generated by macros

Macro Example:

```
%macroalloclib(libref=,disp=shr);
  /* Parameters:
     LIBREF Name of SAS library
            to be allocated
     DISP   OLD or SHR   */
  failure = 1;
  do while (failure);
     _msg_ =
'Attempting to allocate ' || "&libref";
     refresh;
     sysrc =
libname(trim(symget('prefix'))||
"&libref",'v607',"disp='"||
"&disp"||"'");
     if sysrc = 0 then failure = 0;
     if failure then do;
       _msg_ =
'Allocation failure.      Waiting   20
seconds'||
' before retrying';
       refresh;
       start = time();
       do i = start to start + 20;
       end; /* delay loop */
     end;
  end;
%mend alloclib;
```

SCL Program 1:

```
Init: Return;
Main:
 %alloclib(libref=database,disp=old);
Return;
Term: Return;
```

SCL Program 2:

```
Init:
 %alloclib(libref=lookup,disp=shr);
Return;
Main:
          :
 %alloclib(libref=master,disp=old);
Return;
Term: Return;
```

Method Blocks

A method block is a group of SCL statements that begin with a METHOD statement and ends with an ENDMETHOD statement. Using SAS version 6.07, a method block is invoked by the CALL METHOD statement (version 6.08 includes a NOTIFY statement that can invoke method blocks). A method block is treated as an external function and thus communicates via a parameter list. A method block must have at least one parameter.

Positive Aspects:

1. can contain some executable SCL statements (see negatives below)
2. can be invoked by another method block
3. can be invoked by multiple SCL and/or SAS/FSP programs
4. can be used to move code out-of-line and thus, break-up program into smaller (hopefully functional) sections
5. can contain SUBMIT blocks
6. can contain macro blocks
7. can be part of macro blocks
8. modifications can be made to method blocks without having to re-compile all programs (this is especially good for SAS/FSP applications which are cumbersome to re-compile)
9. multiple method blocks can be grouped together in a single program entry and can reduce storage requirements by sharing global variables

Negative Aspects:

1. must be version 6.07 or higher
2. can not reference screen variables, screen functions (e.g. MODIFIED is a screen function) or screen statements (e.g. CURSOR is a screen statement)
3. can not invoke PF key functions (i.e. PF keys are defined as screen functions)
4. can not contain non-executable statements (e.g. ARRAY, ENTRY, LENGTH, etc. - such statements can be placed elsewhere in the entry)
5. code is out-of-line, possible out-of-program and can be more difficult to understand
6. can not contain a link block
7. can not contain another method block

8. method blocks are SAS/AF program entries and can be maintained only by the BUILD Facility (invoked by the BUILD command, PROC BUILD step, or the EDIT/SELECT command while on the CATALOG window of a SAS/AF catalog)

Method Example:

```
alloclib:
  method libref $ 8 disp $ 3
         failure 8;
  /* Parameters:
     LIBREF Name of SAS library
            to be allocated
     DISP   OLD or SHR    */
  failure = 1;
  attempts = 0;
  do while (failure);
    attempts + 1;
    if attempts > 5 then return;
    sysrc =
libname(trim(symget('prefix'))||
"&libref",'v607',"disp='"||
"&disp"||"'");
    if sysrc = 0 then failure = 0;
    if failure then do;
      start = time();
      do i = start to start + 20;
      end; /* delay loop */
    end;
  end;
endmethod;
```

SCL Program 1:

```
Init: Return;
Main:
  call method('alloclib',
              'database','old',rc);
  if rc > 0 then do;
     _msg_ =
'Allocation of DATABASE failed after'||
' 5 attempts';
     refresh;
  end;
Return;
Term: Return;
```

SCL Program 2:

```
Init:
  call method('alloclib',
              'lookup','shr',rc);
  if rc > 0 then do;
     _msg_ =
'Allocation of LOOKUP failed after'||
' 5 attempts';
     refresh;
  end;
Return;
Main:
        :
  call method('alloclib',
              'database','old',rc);
  if rc > 0 then do;
     _msg_ =
'Allocation of DATABASE failed after'||
' 5 attempts';
     refresh;
  end;
Return;
Term: Return;
```

Conclusion:

As shown, each method has good points. A typical solution to most complex problems would be a combination of methods. Understanding each method's short comings is the first step in a total solution. Below is a table of pros and cons for each method.

The authors can be contacted at:

Atlantic Research Corporation,
Information Systems Division
1301 Piccard Drive, 2nd Floor
Rockville, MD 20850
(301) 258-5300

Comparison of Selected features of Link, Macro and Method Blocks

	LINK	MACRO	METHOD BLOCK
Types of SAS Statements	Executable SCL statements only	Any statement(s) (Base SAS or SCL)	Some executable SCL statements
Invocation	Can be invoked only from within a single SCL program	Can be invoked from within multiple SCL programs and/or any base SAS program	Can be invoked from within multiple SCL programs
Storage	Must be stored with the SCL program	Should be stored outside of SCL program to ensure proper macro definition	Can be stored with the SCL program or stored as separate program entry in the same or a different SAS library
Maintenance	Calling program must be re-compiled whenever changes are made	All programs that invoke the macro should be re-compiled when the macro is changed - macro can be maintained outside of SAS environment	If stored independent of calling programs, calling programs do not have to be re-compiled after changes are made to the method block
Can be invoked by multiple programs?	NO (local to resident program)	YES	YES
Can access screen variables?	YES	YES	NO
Can contain a link block?	YES	YES	NO
'In-line' or 'out-of-line' statements	Statements moved 'out-of-line'	Statements will be placed 'in-line'	Statements moved 'out-of-line'
Can contain a method block?	NO	YES	NO