Introduction to Version 6 SAS/AF® software

Brenda Erickson
Linda Wharton
SAS Institute Inc., Cary, NC

ABSTRACT
SAS/AF software for personal computers combines a powerful programming language with the ability to easily create customized menu-driven applications for end users. This paper introduces the new concepts in Version 6 SAS/AF software, including new options in the BUILD procedure statement, the panels that make up an entry, the program screen language, and conversion of applications developed with Version 5 SAS/AF software to Version 6 format.

INTRODUCTION
Version 6 SAS/AF software provides the control and programming capability needed to write powerful applications for end users. The control is provided with a programming language similar to the SAS data step language. The program screen language uses special functions to perform interactive query and handle multiple screens. Prototype applications can be easily and quickly designed, and production applications can be implemented using the SAS/AF software tools.

Applications designed with SAS/AF software can eliminate keying errors and the frustration that can be experienced by new users. Applications designers have complete control over the fields and the logic of the program, allowing them to trap error messages and help the end user before code is submitted to the SAS System for execution. A well-designed application built with SAS/AF can be the next best thing to having a SAS expert at your user’s side.

THE BUILD PROCEDURE
The BUILD procedure is a facility for creating and managing the screens that make up an application. The four primary screen types are MENU, HELP, CBT, and PROGRAM. MENU screens provide selections; HELP screens provide information about an application, a screen, or a field; CBT screens contain the frames that make up computer-based training; and PROGRAM screens prompt the user and drive the application. Screens are stored as entries in a SAS catalog; catalogs are members of SAS data libraries. The catalog directory provides an index or listing of the entries stored in a catalog and allows you to edit, delete, rename and copy them.

PROC BUILD syntax in Version 6
The BUILD procedure syntax for initiating a BUILD session in Version 6 provides more options for merging and printing the entries or selected entries. The Version 6 syntax is:

PROC BUILD CATALOG= BATCH I BROWSE NOCMENU NODIR
PADCHAR="" TEXTLENGTH=n;
PRINT DISPLAY|FIELDS SHOWFILL SOURCE ATTR XREF LISTD;
MERGE CATALOG= NOSOURCE NOEDIT REPLACE UPCASE
MEMTYPE= SELECT= EXCLUDE=;
COMPILE SELECT= EXCLUDE=;

The PRINT statement allows you to print individual members of a catalog, the attributes of an individual member of a catalog, or the program that drives the screen. The options determine what part of the entry is printed. For example, the SHOWFILL option prints the fields with the fill characters instead of with the Xfieldname displayed. The XREF option prints a table showing the entries called by any MENU entries specified in the SELECT or EXCLUDE option. The MERGE statement allows you to merge another catalog’s entries into the current catalog. The COMPILE statement refers only to PROGRAM entries and allows you to compile the source programs for specified entries.

PROGRAM ENTRIES
Program entries are versatile and powerful tools for building applications. Program entries are made up of the following elements: a screen or display panel, a general attributes panel, an attributes panel, and a source panel. The screen is the portion of the entry that is used to communicate with the end user. It is the portion of the entry that can be used to prompt the end user for information. A typical screen consists of text and fields that prompt the user for information and provide a place for the user’s response. The attributes panel for a program entry is actually a field attributes panel where options for each field defined on the screen can be specified. The general attributes panel allows general options, such as a help or keys entry to associate with the entire entry. The source panel is used for entering programs that control the fields and drive an application. Now let’s look at the details of creating a PROGRAM entry. On the command line of the catalog directory, use the EDIT command to specify the name of the new entry you want to create. The display panel
portion of the program entry is displayed. On this panel, you can design a screen that will be used by the program to communicate with the end user.

---

**Data set name: &ds__________**

---

After designing the display screen, you may assign special attributes to the user fields using the ATTR command. The replacement field is a new attribute field. You can use the replacement field to define a special substitution string for a field. The replacement is made only if the field is not left blank on the screen. You can see how this attribute can make your code easier to read in the sample program later in this paper. In Version 6 program entries, the source program is stored on a special panel. While editing the entry, simply type the command SOURCE on the command line to display the source panel. The source panel appears as a blank screen. You can use text editor commands to edit your program. Using the SOURCE command, you can access the source panel from any other panel associated with a program screen. In summary, some of the new features of the BUILD procedure in SAS/AF software for Version 6.03 are:

- options in the BUILD procedure statement to allow printing, merging, and compiling
- program entries with an optional display panel, plus a place for field attributes (ATTR), general attributes (GATTR), and a source program (SOURCE) which drives an application.

**The SOURCE Program: Program Screen Language**

The most significant feature in SAS/AF software for the PC is the program screen language, PSL. The program screen language replaces the Version 5 ### control macro language, used "below the line" in Version 5 program screens. This screen control language is provided to help those applications programmers who need more control over the screen input and output. The applications developer can cross-validate user fields, perform computations, process specific requests for an application, generate SAS source statements and interface to other SAS procedures. The syntax of this language is very similar to the syntax of the SAS DATA step. With few exceptions, the statements and functions available in the DATA step have been implemented in the program screen language. Many additional functions and statements provide the functionality necessary to communicate between screens and to design and execute an interactive application.

**Features of the PSL**

Two features of the new program screen language that are essential for developers to understand are SUBMIT-ENDSUBMIT and the reserved labels; INIT, MAIN, and TERM. SAS source code that is generated for submission to the SAS system is enclosed in a submit block. The submit block distinguishes PSL statements from the statements that you want to submit to the SAS System for execution. A submit block begins with SUBMIT; and ends with ENDSUBMIT;. You can specify in the SUBMIT statement that the SAS source be executed immediately (while the user waits) or held and executed after the application is ended. Within the block, substitution is made for specified replacement strings. The reserved labels INIT, MAIN, and TERM, are used to control when sections of the program are executed. These reserved labels clearly define three tasks to be performed:

- The INIT task is executed just before the screen is displayed to the user.
- The MAIN task is executed after a field on the screen is modified by the user and the ENTER key is pressed.
- The TERM task is executed when the user issues the END or CANCEL command or when the applications programmer forces an END within the program.

Below is a sample program that could be associated with the display panel used earlier. The INIT section displays a message on the message line. The MAIN section checks the user field and links to the INIT section if no name has been entered by the user; if a name has been entered a PROC PRINT statement is submitted to the SAS System. No action is performed in the TERM section, so the label is closed with a RETURN statement.

```plaintext
init: _msg= 'Please enter data set name. '
return;
main: if &ds=' then link init;
submit;
proc print data=&ds;
run;
endsubmit;
return;
term: return;
```
Replacement Strings In the PSL, statement execution can be controlled by the traditional if-then/else construct and by using replacement strings. When a replacement string is used to associate a string with a field, the string is substituted in the submit block. The replacement string is used only if the field is not blank. If the user field is blank, the user field is ignored, or dropped, from the generated code. Thus, a replacement string is an implicit IF clause.

For example, for the &ds field, which is the field for the data set name, we could assign a replacement string of

\[
data = \&ds
\]

on the attribute panel. Then in the SOURCE panel, the program can use the implicit IF and replacement features for submitting statements.

```plaintext
init: return;
main: submit;
   proc print &ds;
   run;
   endsubmit;
term: return;
```

During execution of the application, if the user does not supply a data set name in the &ds field, the generated SAS statements are

```plaintext
PROC PRINT;
RUN;
```

However, if the user supplies the data set name, in.sales2, the generated SAS statements are

```plaintext
PROC PRINT DATA=IN.SALES2;
RUN;
```

The equivalent version 5 SAS/AF program would be:

```plaintext
PROC PRINT #DS DATA=&DS #;
RUN;
```

With Version 6 you have an entire programming language available to control the fields and values. Version 6 SAS/AF software provides features that let you develop more powerful interactive applications. PSL functions provide the tools to control error checking, open data sets, verify data, and communicate between screens. Some of the special functions of the PSL are:

- Functions to interact with SAS data sets: OPEN, CLOSE, EXIST, FETCH, UPDATE, APPEND, SET, DELETE, PUTVAR, GETVAR
- Functions to access variables stored in SAS data sets: NVAR, VAREXIST, VARLABEL, VARNAMES, VARTYPE, VARLEN, VARNUM, VARFMT, VARIMFMT
- Functions to set error flags for fields: ERROR, MODIFIED, ERRORON, ERROROFF
- Functions to display windows: DISPLAY, CATLIST, DIRLIST, FILELIST, VARLIST, WDEF, LEGEND, TABLE, FSEDIT, FSPRINT
- Macro variables and functions: SYMGET, SYMPUT, SYMGETN, SYMPUTN

For example, the VARLIST function allows the programmer to display a list of variables from a data set so that the end user simply chooses from a list of variables. Using VARLIST, the developer can display special types of variables for selection. For example, if only numeric variables can be used, the list can be restricted to display only the numeric variables in the data set. The user simply positions the cursor and presses ENTER to select a variable.

Using the LEGEND function, you can define a window and provide instructions to the user during an application. You can build small windows for prompting users and provide special instructions in a LEGEND window.

An important feature in any application is the ability to perform data validation. The program screen language provides the functions required to check for a specific value or compare many values in a SAS data set.

For example, to verify a specific value, use either IF-THEN/ELSE

```plaintext
if ds = 'SASWORK.JAN' then ... ;
```

or SELECT/WHEN

```plaintext
select (ds);
when ('SASWORK.JAN') ...;
when ('WORK.JAN') ...;
otherwise msg = 'Invalid data set name, please reenter';
end;
```

To check a user-supplied data value against the values of a specific variable in a specific data set, use the PSL functions. For instance, in this program we are comparing an integer value...
entered by the user with the values stored in the variable X in a data set.

| Enter an integer value: &num___ |

init:  
x = 0; /*set field type to numeric, init to 0*/  
return;

main:  
found = 0;  
dsnnum = open('dataset',I,'');  
call set(dsnnum);  
getx = fetch(dsnnum);  
do while ( (getx ^= -1) and (found = 0) );  
    if num = x then found = 1;  
    else getx = fetch(dsnnum);  
    end;  
if found = 0 then _msg_ = 'Value not found in data set';  
rc = close(dsnnum);  
return;

term:

You can also easily calculate and display values based on user-supplied information. In the following example, total regular and overtime hours for several work projects are calculated based on the user's reported hours.

| Enter hours worked for each project.  
Landau project: &L__  
Peters project: &P___  
Cary project: &C___  
| Total hours: $TOTAL__  Overtime hours: $OVER__ |

init:  
l = 0;  
p = 0;  
s = 0;  
total = 0;  
over = 0;  
return;

main:  
if total > 40 then over = total - 40;  
else over = 0;

CONVERTING VERSION 5 APPLICATIONS

If you have Version 5 applications, you can use the CPORT and CIMPORT procedures, which are now a part of base SAS software, to move and convert your Version 5 applications to Version 6 format. A conversion option in PROC CIMPORT allows users to easily convert applications written with the V5 # language to the V6 format. Any operating-system-dependent changes can be made by the applications programmer.
CONCLUSION

Version 6 SAS/AF software has features that simplify and accelerate the design and implementation of production application systems. When prototyping and developing applications, a variety of needs must be met. SAS/AF software combines programming power with easy screen design capabilities, CBT capabilities for your training needs, help system capabilities for assisting your users, and easy menuing design capabilities to let you create intelligent applications. The versatility of SAS/AF software combined with the power of base SAS software provide extremely powerful tools for the applications developer using the SAS System.