Using SAS/AF® as a Front End for Developing Report Program Skeletons

Jeff Hamilton, BS, and Jennifer Lester, MA, STATPROBE, Inc., Ann Arbor, MI

Abstract

With the growing demands being placed on information system departments, the opportunity to off load "simple" programming tasks to non-SAS® programmers adds efficiencies into the program development life cycle. Using a SAS/AF application as a front-end entry tool can build such efficiencies into program development. Standardized reporting tools developed at STATPROBE, Inc. have been designed to allow the programmer to make table shell modifications very easily. These modifications have been automated via SAS/AF allowing a non-SAS programmer to create table shells with minimal involvement of the programming staff. Standard report program "skeletons" are created by the user through AF Frames. Essential program modules are then combined to create usable SAS code. The SAS program created using SAS/AF produces a table shell which can later be modified by a SAS programmer to populate the table shell.

Introduction

STATPROBE is a contract research organization (CRO) that currently serves the pharmaceutical, biotechnology, and medical device industries and academia. Most of our clients employ STATPROBE to produce either a Research Report or a Statistical Report, both of which contain SAS-generated data listings, summaries, and analyses appendices. Upon initiation of a project, a statistician, medical writer, and programmer collaborate to produce a "mock" report which includes a computer-generated "table shell" for each data listing and summary appendix. These shells are presented to the client as a first look at how their data will be displayed. In the past, the process of developing table shells began with a statistician or medical writer designing by hand each listing and summary table shell. These specifications were then given to a SAS programmer who would write the skeleton code necessary to generate the table shells. This same code was later utilized in the actual data-populated listing or summary program. Typically, two or three review cycles between the designer and the programmer were necessary to produce the final report table shells. There were three obvious inefficiencies in this process.

- It is difficult to ensure that the designer had effectively communicated all of his or her specifications to the programmer.
- For many types of data, the designer must see the shell as SAS output rather than as hand sketches, in order to develop the most appropriate presentation of the data.
- Due to the STATPROBE standard report tools, writing the SAS code for table shells is a very low-level programming task, and it is not the most efficient use of an advanced programmer's time.

With these three items in mind, it was clear that the most expedient way to produce a table shell was to remove the programmer from the process. Unfortunately, most medical writers cannot program in SAS, and, like the time of most SAS programmers, the time of a statistician and a medical writer is better used in other capacities. This led to the development of a graphical user interface (GUI) front end using SAS/AF.

Rather than the user sketching out the table by hand, the GUI prompts the user for all of the necessary specifications for each table shell. The data entered into the user interface is structured as a set of SCL lists and then saved to disk as SLIST entries. A program-generator engine then utilizes the SLIST entries to produce the SAS code for each table shell. These steps are transparent to the user, and, with a simple mouse click, the user can preview the table shell on the screen or execute the program to print out the table shell. In the time it used to take the designer to sketch out the shell by hand, he or she can enter into the user interface all of the necessary information required and the SAS table shell program can be generated.

STATPROBE Standard Report Tools

STATPROBE's report programming system has evolved into an extensive compilation of macros. A standard macro library exist from which all projects can draw for pre-programmed and pre-validated code. This macro library has in turn built many standards into our programming techniques. These standards have resulted in efficient, maintainable code that has the same modular look and feel from program to program and project to project. Two of the primary macros that attribute to the modular look and feel are LISTTOOL and SUMTOOL. The framework of the DATA _NULL_ as a reporting tool has been abstracted from the individual reporting programs, standardized, and stored in the LISTTOOL and SUMTOOL macros. This abstraction of the DATA _NULL_ to a macro has reduced the individual programs to three basic components:

- Data Preparation
- Report-Specific Macro Definitions
  - %Subhead
  - %Body
  - %Lefttext
  - %Foot
- Call to the LISTTOOL or SUMTOOL for Actual Report Generation

It is this modular programming style that the user interface was structured toward.

User Interface

The SHELL GENERATOR's Initialization Frame (Figure 1) serves two primary purposes. The first is to allow the user to select a current client and project. Under this scenario the frame acts as a navigational tool which, for the specified client and project, guides the user to a more detailed frame where one can either view existing shells, create new shells, or print shells. The second function of this frame is an entry point for project title lines. A project title is the specific three line title that appears at the top of each program-generated appendix (see Appendix 1). These project title lines are stored as SLIST entries and referenced at the time of program generation.

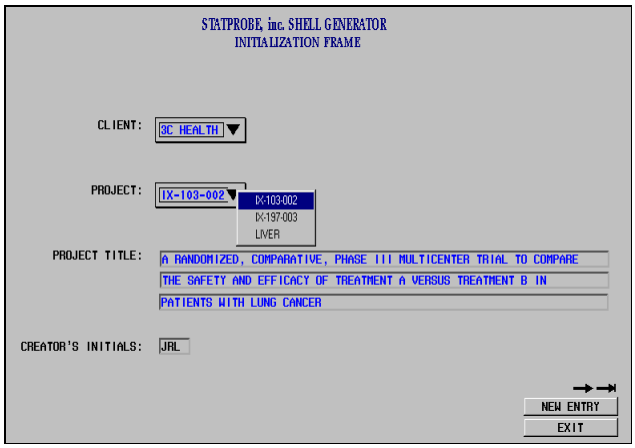

Figure 1. Initialization Frame

Once a client and project have been selected from the Initialization Frame the natural progression leads the user to the Program Specification Frame (Figure 2). The primary function of this frame is to act as an entry point for report specific title and reference lines. Similar to the Initialization Frame where project specific titles are entered, this frame allows entry of specific report titles that belong to a given client and project. A hierarchical relationship exists between the project titles and the report specific titles; one client and project will have many reports. This frame, as the title indicates, is for program specific detail. Thus, the program name and purpose are captured to a SLIST for future reference by the program-generator engine.

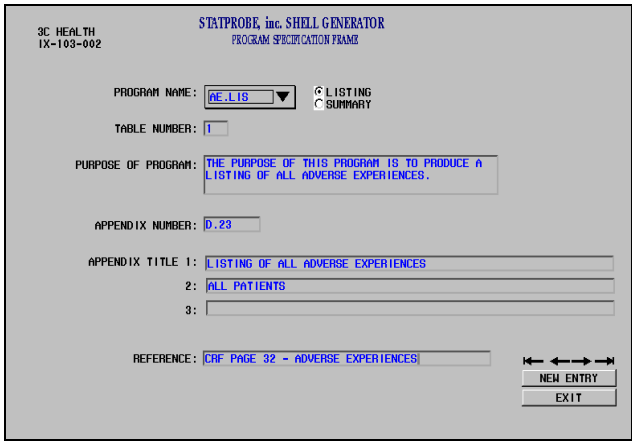

Figure 2. Program Specification Frame

The Table Shell Design Frame (Figure 3) is used for the detail design of all projects' table shells. This frame has replaced the paper and pencil table specification design technique of yesterday. A data table object has been placed on the frame as an entry point for the user to design each report table shell. The data table object functions similarly to a spreadsheet application in allowing the user to design in a free form environment with the only limitation being preset system defaults. The top down flow of this frame follows the same sequence as the report table shell appearance. First the user may enter an optional pre-header text entry. This field is solely for cosmetic purposes during the report table shell generation. Following the pre-header text the user will enter the column widths and the column headers into the data table object. This gives the user a rudimentary display of the report table shell. The same data table object is used for entry of the body of the table shell. This proves to be used primarily on the summary tables, but the functionality exists for listing tables as well. This frame finishes the top down design by allowing the user to enter additional footnote lines. These footnote lines will be listed on the final table shell output below the reference entry, captured on the Program Specification Frame.

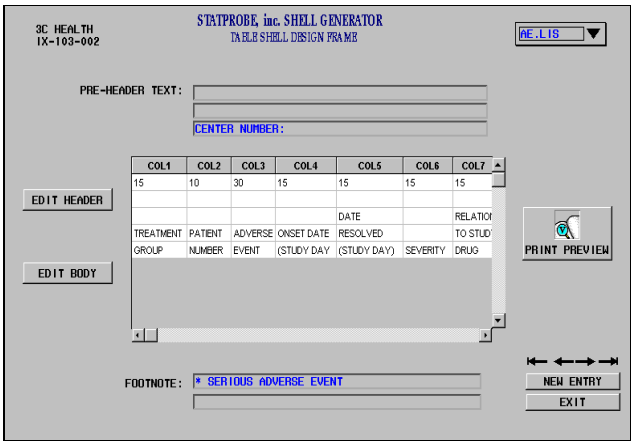

Figure 3. Table Shell Design Frame

The Shell Design Preview Frame (Figure 4) allows the user to view the table shell design without the added overhead of the program-generator engine. A visual sample of the table shell is constructed in the extended text entry object by abstracting the SLIST entries captured on the previous frames. From this frame the user can quickly view all program shells for the currently specified project. The user also has the option of printing the contents of the viewer window by clicking on the 'PRINT' icon or returning to the Design Frame for further modifications or new table shell designs.

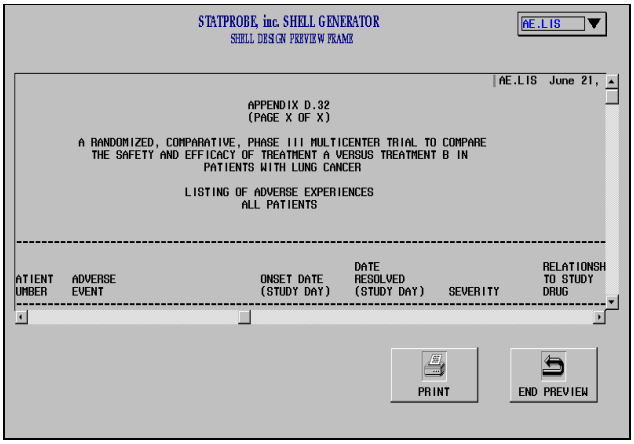

Figure 4. Shell Design Preview Frame

The final frame to the SHELL GENERATOR system is used for printed report generation (Figure 5). This frame allows the user to select program entries to print via a mouse click in the listbox object

or via the 'SELECT ALL' icon object. Once the selection is made a mouse click on the 'PRINT' icon object activates the program-generator engine and issues the submit statements for each of the selected programs. While the programs are being generated a dialog box is superimposed on the frame indicating the event. A similar dialog box is used to indicate the printing process. Once printing completes, the system prompts the user to 'Select additional reports for printing' or 'Select a new project'.

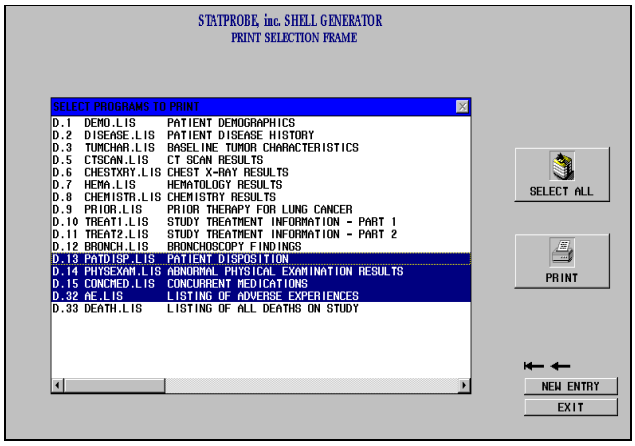

Figure 5. Print Selection Frame

Program-Generator Engine

The launch of the program-generator engine executes a block of SCL code which in turn creates the usable SAS program skeleton code. These program skeletons are simply text files built from SLIST entries captured by the user interface. Once a program has been created it is stored in the project specific program directory and can be submitted to produce the table shell output. An example of the system generated program skeleton is displayed in Listing 1 (dynamic text in bold). The text files created by the engine are actual SAS programs that, when submitted, will produce table shell output similar to that displayed in Appendix 1. These programs can later be modified to populate the table shell with data.

Aside from the obvious time savings in program development and initial system setup, a few of the value added benefits of the program-generator engine are as follows:

- Header comments are properly documented.
- New style standards can easily be built in.
- Standard macros are automatically included.
- All programs have the same look and feel.
- Writers can change table shell programs without programmer involvement.

```
*******************************************************************************;
*** Specification Section                                                     ***;
*******************************************************************************;
*** Physical Specifications                                                   ***;
***   Project ID:       3C HEALTH  IX-103-002                                ***;
***   Program Name:     AE.LIS                                               ***;
***   Created:          09/30/96                                             ***;
***   By:               JRL                                                  ***;
***   Derived From:     NONE                                                 ***;
***   Input Data:       NONE                                                 ***;
***   Output Data:      NONE                                                 ***;
***   Operating Sys:    OS/2,  version 3                                     ***;
***   Language:         SAS, version 6.11                                    ***;
*** Logical Specifications                                                   ***;
***   Purpose:  The purpose of this program is to produce                  ***;
***             a listing of all adverse experiences.                      ***;
*******************************************************************************;
*******************************************************************************;
%Include "h:\3chealth\IX103002\ix103pgm\environ.mac";
%Include "h:\3chealth\IX103002\ix103ml\listtool.mac";

*** Retrieve report title lines and header information.                      ***;
%Let pgmname=ae.lis;
%datapgm(&pgmname);

*** Assign column widths.  Creates &c1 - &cn                                 ***;
%colwids(15,10,30,15,15,15,15);

***  Program specific by variable controls                                  ***;
%Macro header;
   Put @&c1 'CENTER NUMBER:';
%Mend header;

*** Program specific column header controls                                  ***;
%Macro colhead;
   Put @&c5 'DATE'
       @&c7 'RELATIONSHIP'
     / @&c1 'TREATMENT'
       @&c2 'PATIENT'
       @&c3 'ADVERSE'
       @&c4 'ONSET DATE'
       @&c5 'RESOLVED'
       @&c7 'TO STUDY'
     / @&c1 'GROUP'
       @&c2 'NUMBER'
       @&c3 'EVENT'
       @&c4 '(STUDY DAY)'
       @&c5 '(STUDY DAY)'
       @&c6 'SEVERITY'
       @&c7 'DRUG';
%Mend colhead;

*** Program specific body text controls                                      ***;
%Macro bodytxt;
%Mend bodytxt;

*** Program specific footnote text controls                                  ***;
%Macro footer;
    Put @&c1 '* SERIOUS ADVERSE EVENT';
%Mend footer;

%listtool(dsn=,sortvars=, prehead=header, subhead=colhead,
          body=bodytxt, foot=footer, tblnum=1);
Run;
```

Listing 1. Sample skeleton program

Conclusion

In its current state, this application has added several efficiencies into the production of a "mock" Statistical or Research Report. SAS programmers have been freed from the low-level task of programming table shells, resulting in increased availability for more complicated tasks. By the time the programmer is asked to produce the data-populated appendices, the skeleton code for each program is already in place. Also, the statisticians and medical writers have complete control over table shell development and ultimately spend less time on design and modification.

References

SAS Institute, Inc. (1993), *SAS/AF Software: FRAME Entry, Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute, Inc. (1994), *SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute, Inc. (1995), *SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

Savage, Jennifer (1995), *Object-Oriented Reporting Tools*, Proceedings of MWSUG '95, 98-100, Cleveland, OH.

Acknowledgments

SAS is a registered trademark of SAS Institute, Inc. in the USA and other countries. ® indicates USA registrations.

The author's would like to thank Matt Becker and Carl Haske for their creative influence and support.

Authors' Address

Jeff Hamilton and Jennifer Lester
STATPROBE, Inc.
3885 Research Park Drive
Ann Arbor, MI 48108
(313) 769-5000

E-mail:
- 72700,1075@compuserve.com
- statprob@oeonline.com

Appendix 1. Example of a standard report table shell

AE.LIS **June 21, 1996**

APPENDIX D.32
(PAGE X OF X)

A RANDOMIZED, COMPARATIVE, PHASE III MULTICENTER TRIAL TO COMPARE THE SAFETY AND EFFICACY OF TREATMENT A VERSUS TREATMENT B IN PATIENTS WITH LUNG CANCER

LISTING OF ALL ADVERSE EXPERIENCES
ALL PATIENTS

CENTER NUMBER:

TREATMENT GROUP	PATIENT NUMBER	ADVERSE EVENT	DATE ONSET DATE (STUDY DAY)	DATE RESOLVED (STUDY DAY)	SEVERITY	RELATIONSHIP TO STUDY DRUG

REFERENCE: CRF PAGE 32 - ADVERSE EXPERIENCES
* SERIOUS ADVERSE EVENT