The Pediatric Oncology Group Accessing Clinical Trial Data Across the Internet via a Bulletin Board System Using SAS/AF® and SAS/FSP® Software

Barbara Blum
Cynthia Hurley
Pediatric Oncology Group, Statistical Office
Gainesville, Florida (904) 392-5198

Abstract:
This paper describes how the Pediatric Oncology Group (POG) Statistical Office is utilizing SAS/AF and SAS/FSP to allow restricted access (via a UNIX-based BBS with an ASCII interface, connected to the Internet) to patient data kept in SAS data sets.

Introduction:
POG is one of two American collaborative groups dedicated to the treatment of childhood cancer. Over 100 major medical centers (located in the USA, Canada, and Europe) are clinical members of POG. These POG institutions register more than 2,500 patients per year onto more than 80 ongoing POG clinical trials. The POG Statistical Office, located at the University of Florida, serves as the data repository and reporting group for POG. The POG Statistical Office has used SAS since 1980 to maintain a clinical trial patient database and to analyze the patient data.

In order to provide more timely reporting and superior communications, the POG Statistical Office implemented a Bulletin Board System (BBS) in January 1994. The POG BBS provides more than 1,700 POG personnel access to e-mail services, news groups, document transfer capabilities, and SAS applications -- across the Internet or via modem. The POG BBS applications consist primarily of SAS/AF and SAS/FSP programs. The current applications are POG_Data, POG_Info, POG_9400, and Study_Coor. These applications use extensive Screen Control Language (SCL) programs, custom PMENUs, and macros. As a representative example, the Study_Coor application is discussed here.

Study_Coor
This application allows browse access via security codes to patient data ordered by study number. To insure patient confidentiality, security codes are provided only to POG Study Coordinators (the physicians who serve as Principal Investigators of particular POG studies) and their authorized assistants. A valid security code must be entered to display the study or studies for which the Study Coordinator or assistant has responsibility. Each valid security code is stored in a SAS data set and is associated with a maximum of ten POG studies, which, once a valid security code is entered, are displayed between the <> brackets as shown on the screen below. Portions of the SCL code for the security screen are shown below the screen and on the next page.

Study_Coor Information System

Enter Your Security Code ________

(Code will not appear.)

To CANCEL the application, press the <Tab> key to position the cursor in the next field, and enter a non-blank character: < >

array command[66] $ 80 ;
INIT:
control label;
link disable;
sc_found=0; /* initialize the flag */
.
. /* librets, file existence checks, etc. not listed */
m=1; /* initialize number of valid security code tries */
studyco=vamum(dsid,'stc_no');
message= _blank_; /* initialize the message */
cursor stc_no; /* position the cursor */
rc=activate('choice',-1); /* deactivate choice */
rc=activate('choice',-2); /* deactivate choice */
rc=activate('choice',-3); /* deactivate choice */
rc=activate('choice',-4); /* deactivate choice */
rc=activate('choice',-5); /* deactivate choice */
rc=activate('choice',-6); /* deactivate choice */
rc=activate('choice',-7); /* deactivate choice */
rc=activate('choice',-8); /* deactivate choice */
rc=activate('choice',-9); /* deactivate choice */
rc=activate('choice',-10); /* deactivate choice */
return;
STC_NO:
message=_blank_; /* increment the # of tries */

if (m=4) then /* if more than 4 tries, exit */
call execmd('endsas');
val=locaten(dsid,studyco,sc_no); /* check security code */
if (val<0) and (m=4) then do;
message='Invalid code. Try again';
cursor stc_no; /* position the cursor */
end;
if val>0 then sc_found=1; /* set flag to found */

if (m<4) and (val>0) then do; /* valid security code */
call set(dsid);
ysrc=fetchobs(dsid,val); /* place studies on-screen */
if study ne. then do; /* if study exists */
rc=activate('choice',10); /* activate pushbutton */
end;
if study9 ne. then do; /* if study exists */
rc=activate('choice',9); /* activate pushbutton */
end;
if study8 ne. then do; /* if study exists */
rc=activate('choice',8); /* activate pushbutton */
end;
if study7 ne. then do; /* if study exists */
rc=activate('choice',7); /* activate pushbutton */
end;
if study6 ne. then do; /* if study exists */
rc=activate('choice',6); /* activate pushbutton */
end;
if study5 ne. then do; /* if study exists */
rc=activate('choice',5); /* activate pushbutton */
end;
if study4 ne. then do; /* if study exists */
rc=activate('choice',4); /* activate pushbutton */
end;
.
.
if study3 ne. then do; /* if study exists */
rc=activate('choice',3); /* activate pushbutton */
end;
if study2 ne. then do; /* if study exists */
rc=activate('choice',2); /* activate pushbutton */
end;
if study1 ne. then do; /* if study exists */
cursor study1; /* position cursor on pushbutton */
rc=activate('choice',1); /* activate pushbutton */
end;
end;
return;
.
.
. Depending on which study push button is activated,
call the next program (SCMAIN) & pass the study number;
STUDY1:
call display('scmain.program',study1);
return;
STUDY2:
call display('scmain.program',study2);
return;
STUDY3:
call display('scmain.program',study3);
return;
STUDY4:
call display('scmain.program',study4);
return;
STUDY5:
call display('scmain.program',study5);
return;
STUDY6:
call display('scmain.program',study6);
return;
STUDY7:
call display('scmain.program',study7);
return;
STUDY8:
call display('scmain.program',study8);
return;
STUDY9:
call display('scmain.program',study9);
return;
STUDY10:
call display('scmain.program',study10);
return;
CANCEL:
call execmd('endsas');
return;
.
.
. if valid security code, protect the security code field */
if sc_found=1 then rc=field('protect','stc_no');
return;
.
.
. /* TERM, DISABLE, and RESTORE code not listed */
Selecting a particular study displays the following screen, where XXXX would be the actual study selected by the user.

### Study Coordinator Application

<table>
<thead>
<tr>
<th>Options for Study XXXX</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; Study Coordinator Form Data &gt;</td>
</tr>
<tr>
<td>Current as of 04/29/94</td>
</tr>
<tr>
<td>(Frozen Cutoff)</td>
</tr>
<tr>
<td>&lt; Roadmap/FlowSheet Data &gt;</td>
</tr>
<tr>
<td>Not Stored for Study XXXX</td>
</tr>
<tr>
<td>&lt; Patient Identification Information &gt;</td>
</tr>
<tr>
<td>Current as of 07/25/94</td>
</tr>
<tr>
<td>&lt; EXIT THE APPLICATION &gt;</td>
</tr>
</tbody>
</table>

The data is current POG Statistical Office data. The Study Coordinator Form Data is updated every six months while the Roadmap/FlowSheet Data and the Patient Identification Information are updated weekly. The date of the most recent weekly update is displayed via a read to a file that contains the macro below, which is manually modified every update.

```scl
%LET CUTOFF=MDY(10,20,94); /* line 6 */
%LET DB_DATE=10/20/94; /* line 7 */
%LET OLDCUT=MDY(10,17,94); /* line 8 */
%LET OLDDATE=10,17,94; /* line 9 */
```

Portions of the SCL code for the above screen follow:

```scl
/* to avoid overly long defaults, set lengths ourselves */
length clause $ 60 clause1 $ 45 cut1 cut2 cut3 $ 2;
entry study 3; /* study number from calling program */

INIT:
n=1; /* initialize */
******read current date file*******;
rc=filename('current','/dbinforfile.dbf');
******read the date of last update for display.;
fileid=fopen('current','I',0);
if fileid = 0 then do;
message='The file CURRENT could not be opened.';
do i = 1 to 1000; end; /* Had to build our own pause */
end;
else do;
if (fileid>0) then do
while((fread(fileid)=0) and n<7); /* read date on line 7 */
rc=fget(fileid,c,21); /* char var c gets the data */
length of line is 21 /*
end;
n=n+1; /* increment */
rc=fclose(fileid);
end;
****** bring in date frz ******;
studymsg='Options for Study "listudy; /* datefrz. is a format in our libraries */
/* pull the date of the frozen cutoff using this format */
cut1=put(1,datefrz.);
cut2=put(2,datefrz.);
cut3=put(3,datefrz.);
/* concatenate the date with the text on the screen */
current1='Current as of 'licut1'fllcut2'I'fllcut3;
current2='Current as of 'lisubstr(c,14,9);
current3='Current as of 'lisubstr(c,14,9);
****** activate choices ******;
rc=activate('choice',1);
****** exclude particular roadmaps ******;
if study not in(8625,8651,8653,8654,8704,8710,8725,9005,9006)
then do;
rc=gray('choice',2); /* change if add more roadmaps */
current2='Not stored for study 'listudy;
message2= _blank_;
end;
****** set up WHERE clauses for data set subsetting ******;
clause='(where=(study_no='llstudyll));
clause1='study-no='llstudy;
return;

MAIN:
if choice ne _blank_ then call execmd('pmenu on');
select(isactive('choice'));

when(1) do;
message1=NOTE: Gathering Study Coordinator Data;
refresh;
generic=open('gensc.generic','I');
sysrc=where(generic,clause1);
result=ATTRN(generic,'any');
rc=close(generic);
if result then do;
/* Study exists. */
call fsedit('gensc.generic',clause1,
'gensc.gensc.scmain.screen','browse');
message1=_blank_; refresh;
end;
```

882
else /* Study does not exist. */
if not result then do;
    message1='studyl' has no study coordinator forms';
    r=activate('choice',-1);
end;
end;

when(2) do;
message2='NOTE: Gathering Roadmap/FlowSheet Data...';
refresh;
call fsedit('gensc.roadmap'llclause,'gensc.gensc.roadmap.screen','browse');
message2= _blank_;
refresh;
end;

when(3) do;
message3='NOTE: Gathering Patient Data...';
refresh;
call fsedit('bbsdata.bbsptid'llclause,'bbsptid.bbsptid.bbsptid.screen','browse');
message3= _blank_;
refresh;
end;

when(4) /* turn off pmenu which was turned on at application start */
call execmd('pmenu off; end');
otherwise;
end;
return;

TERM:
return;

Study Coordinator Form Data

This option uses FSEDIT to allow a POG BBS user with a valid security code to browse the three screens of Study Coordinator Form data for a particular study. The information displayed is GENERIC to most of the Study Coordinator forms, as many forms are specific to particular studies and are not included in this application at this time. The second screen, as shown below, has a tricky spot in that we have three classifications of eighteen (18) variables: Type of Toxicity (Tox_C#), Degree of Toxicity (Tox_D#), and Course of Toxicity (Tox_CS#), where the # ranges from 1 to 18 for each variable.

Browse Generic Study Coordinator Data

<table>
<thead>
<tr>
<th>[NXT_OBS]</th>
<th>[PRE_OBS]</th>
<th>[NXT_SCR]</th>
<th>[PRE_SCR]</th>
<th>[WHERE]</th>
<th>[EXIT]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study_no</td>
<td>POG_no</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access_n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date_INE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protcomp</td>
<td>RespEval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tox_Eval</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type (Tox_C#)/Degree (Tox_D#)/Course (Tox_CS#)
1. / / 7. / / 13. / / 
2. / / 8. / / 14. / / 
3. / / 9. / / 15. / / 
4. / / 10. / / 16. / / 
5. / / 11. / / 17. / / 
6. / / 12. / / 18. / / 

Because our users need to view patients having any of the eighteen variables matching a certain criteria, we had to provide a method of allowing them to search those eighteen variables all at one time. The pmenu below allows users to choose which variable to search and is created using the code shown below the pmenu.

[WHERE]
[WHERE ALSO]...
[Toxicity/Degree/Course]...
[UNDO LAST WHERE]

This is provided through the PMENU code:
menu wheremenu;
item [WHERE] dialog=dbox4;
item [WHERE ALSO] dialog=dbox5;
item [Toxicity/Degree/Course] dialog=dbox6;
item [UNDO LAST WHERE] selection=undo;
Selecting the [Toxicity/Degree/Course...] option from the pmenu displays the following window, which uses the stored WBUILD macro shown below.

<table>
<thead>
<tr>
<th>[Toxicity/Degree/Course...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Toxicity</td>
</tr>
<tr>
<td>_ Degree</td>
</tr>
<tr>
<td>_ Course</td>
</tr>
</tbody>
</table>

Enter Value: ___________

<OK>  <Cancel>

WBUILD macro:

```
options sasmstore=generic mstored;
%macro wbuild(varlist,value) / store;
%let prefix=%scan(&varlist,1);
%let index=%scan(&varlist,2);
%if &value ne %then %do;
  where %do i=1 %to &index;
    &prefix.&i=&value;
  %if &i ne &index %then %do;
    or
  %end;
%end;
%mend wbuild;
run;
```

The MACRO substitution that occurs is:

```
dialog dbox6 %wbuild(%1, @1);
radiobox default=l;
rbutton #3 @5 'Toxicity' substitute='tox_c 18';
rbutton #4 @5 'Degree' substitute='tox_d 18';
rbutton #5 @5 'Course' substitute='tox_cs 18';
text #7 @1 'Enter Value:';
text #7 @25 len=4;
```

This macro uses the / STORE option to store the compiled macro as an entry into a SAS macro catalog in a permanent SAS data library; later, it can be accessed when you execute the macro in a different SAS program, saving macro-compilation overhead. The MSTORED system option must be in effect to use compiled-stored invocations. Name-style macros (those stored in the SAS macro catalog and invoked via the % followed by the macro name) offer the best performance gain for compiled stored macros. The WBUILD macro searches through each of the 18 variables for the value provided by the user. It treats the variable as though it were an array with 18 entries, where the value of 18 is passed into $index via the macro calling statement. The resulting data set includes any observation in which any of the 18 variables matched the provided value.

**Roadmap/Flowsheet Data**

This option uses FSEDIT to allow a user with a valid Security Code to browse the 12 screens of GENERIC Roadmap/Flowsheet data (if applicable to the study and stored in the Statistical Office database) for that particular study. This option does not employ the use of a macro such as shown in the option above, as Roadmaps are being phased out of use by the POG.

**Browse Roadmap/Flowsheet Data**

```
Obs xxxx
Screen 1
[NXT_OBS] [PRE_OBS] [NXT_SCR]
[PRE_SCR] [SEARCH] [REPEAT] [WHERE]
(EXIT)
Study_no _____ POG_no _____
Access_n _____ Induct_n Dis_type __
Roadmap 1 Date_RM1 ____________
Respond1 __ Date_CR1 ____________
```

<table>
<thead>
<tr>
<th>Type / Grade / Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. CompTY11_ / CompGR11_ / CompEF11_</td>
</tr>
<tr>
<td>2. CompTY12_ / CompGR12_ / CompEF12_</td>
</tr>
<tr>
<td>3. CompTY13_ / CompGR13_ / CompEF13_</td>
</tr>
<tr>
<td>4. CompTY14_ / CompGR14_ / CompEF14_</td>
</tr>
<tr>
<td>5. CompTY15_ / CompGR15_ / CompEF15_</td>
</tr>
<tr>
<td>6. CompTY16_ / CompGR16_ / CompEF16_</td>
</tr>
</tbody>
</table>
**CONCLUSION**

The SAS/AF and SAS/FSP software products have been extremely beneficial to the Pediatric Oncology Group Statistical Office. With these products, we have achieved our goals: (1) provide in-house applications served from one database; and (2) allow over 1700 users both browse and edit access to information stored in SAS data sets via applications available through an ASCII Bulletin Board System (BBS). We are currently testing these products' capability to create data entry applications - which will enable us to progress from a keypunch method of data entry to in-house data entry with immediate checks against our database. This experience will assist us in our long term goal of having all data entry done via applications on our BBS. Once we are able to provide this service to our members, we will enjoy a marked reduction in both paper usage and mailing costs.

**ACKNOWLEDGMENTS**

The authors would like to thank SAS AF/FSP Technical Support for their assistance in explaining the use of the ATTRN function and the succeeding IF statement which assisted us in establishing the necessary clause to subset the data set, as shown on page 3. The authors would also like to thank Ms. Ann Carpenter of SAS AF/FSP Technical Support and Mr. Tom Hahl of SAS Macro Technical Support for providing us with the macro and macro substitution code shown on page 5.

**REFERENCES**


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

Other brand and product names are registered trademarks or trademarks of their respective companies.