Paper 35

TAP Into SAS/FRAME® Technology

David M. Hartman, Boehringer Ingelheim Pharmaceuticals, Ridgefield, CT

Abstract

TAP is an acronym for Table Acquisition Program which enables users to select from a list of programs, various summary tables and listings they would like to produce for a report. The application begins by displaying a FRAME which enables the user to select a clinical trial (database) from which the data will be summarized. Once the trial is selected, the next FRAME appears which provides the selection list of programs. In addition to producing tables from this frame, selected tables may be viewed on the screen and/or sent to the printer. Except for the selection of a study and the production of the selected tables, there is virtually no SCL code. From this application, additional functionality can be added such as screens that would enable the user to control the display of the tables including table numbers, titles, footnotes, special sub-setting of the data, etc. The FRAME entries, objects, and SCL code described in this paper can open the door to a wide variety of easy to use "point and click" applications.

Introduction

In Clinical Research, data is collected in a variety of different ways. This results in many different programs to generate the tables and listings needed to analyze the data. For the user, he/she has to know where the programs are located, and have some programming knowledge. In an effort to provide easier access and execution of the programs (even for those who have very little computer skills) the Table Acquisition Program (TAP) was created. This Graphical User Interface (GUI) enables multiple users to select from a list of programs, various summary tables and listings they would like to produce for a clinical trial report.

TAP is a simple application consisting of only three FRAME entries. The first FRAME is primarily navigational in that the users select a Therapeutic Area, Drug Project, and Clinical Trial. This information is use by TAP to locate the programs and the source of the data to produce the desired tables and listings. The second FRAME, which is the "work horse" of this application, creates tables (in batch) with the .LOG and .LST files for each selection written to a folder. From this FRAME the users are then able to view output on the screen or send the output (.LST) to a printer. The third FRAME is the table-viewing screen that appears when the view button on the second FRAME is selected. The only maintenance required is a SAS® dataset, which contains the list of programs for each trial as well as a dataset for each of the drop down lists of Therapeutic Areas, Projects (specific drug undergoing testing), and Protocols. All the user has to do is "point and click" and TAP does the rest. Now, you are about to embark on a trip through TAP. When you are finished, you will have acquired enough knowledge about SAS/FRAME to create your own application.

Environmental Description

TAP is built to run on a PC/Network server under Microsoft® Windows95. The directory structure has four primary levels. The highest is the SYSTEM level, which is where the SAS/FRAME catalog and the datasets that contain the list of therapeutic areas, projects, and studies are stored. Beneath that is the Therapeutic Area level, then the Project level, and finally the study level which is where the programs that appear in the list are stored. TAP is currently running under version 6.12 of SAS.

Main Frame (primary frame for this application)

For those of you who have never built a SAS/AF® application, the first step is to write a program to create a SAS/AF catalog. The following is an example of such a program:

```
libname in 'g:\sastools';
proc build c=in.tap;
run;
```

Execute this program and you are ready to start building your application. To create a FRAME, click on FILE and NEW. A window will appear where you type in the name you want to call your FRAME and click on the FRAME option. A blank FRAME will appear from which you can add a variety of different "objects" depending upon the functionality you want for that FRAME. To create an object, you can either "right click" anywhere on the FRAME or click on "Actions". On the pop-up menu that appears, click on "make". Another pop-up menu appears from which you can make your selection of the type of object you would like to add to your FRAME. Again, the type of object you select depends on the functionality you require for your application.

The first FRAME that appears for TAP is called "MAIN"(see Appendix I). This FRAME has 9 objects. The first object that appears on MAIN.FRAME is Graphic Text, which displays the text "TAP" at the top of the frame. This object has no Screen Control Language (SCL) behind it. The next three objects are Push Button. These objects provide drop down selection lists (for selecting therapeutic area, drug project, and clinical trial). Once a selection is made from each of the drop down selection lists, the Text Entry objects under each of the three Push Buttons display the selection the user made. The following is a section of the SCL that deals with the Therapy button and the Text Entry object below it:

```
THERAPY:
thpds=open('tap.therapy');
call execcmd('setwname "Select
                  Therapy"');
thplist=datalistc(thpds,'therapy',
                        ' ','Y',1);
call notify('thplist','_update_');
rc=close(thpds);
select(compress(thplist));
      when ('GeneralMedicine')
             thpy='GENMED';
      otherwise;
   end;
RETURN;
```

The OPEN function opens up a SAS dataset containing the list of therapeutic areas. DATALISTC displays the list, and the CALL NOTIFY function displays the selection on the screen. The rest of this code takes the selection and stores it in a macro (THPY) for use later when defining a library reference.

The remaining two objects on the Main FRAME are Push Button. One button allows the user to proceed to the next FRAME and the other button allows the user to exit from the application. Neither of these buttons have SCL code behind them. The "Continue" button has the command, "afappl c=in1.tap.tabsel.frame" while the "Exit TAP" button has the command, "end;" which you will find under Object Attributes/Command Processing.

Table Selection Frame

Once the desired study is selected and the "Continue" button is pushed, a new FRAME appears. This FRAME provides the user with all of the functionality to create, view, and print tables (See Appendix II). The INIT section of the SCL for this FRAME contains, amongst other things, a macro program embedded in a Submit block that checks to see if a dataset called LISTTAB exists. If it does, then it is deleted. The LISTTAB dataset gets populated with the user clicks on the "selected" list of tables. This is an appending process. Therefore, if a user clicks on the Summary button and clicks on the list of tables that appear, the selection will end up in LISTTAB. Later, if a user clicks on the Listing button, LISTTAB starts clean and then gets populated with the set of selections.

The largest of the objects on this FRAME is called List Box. Its purpose is to display a list of programs and allow the user to select from that list, the programs he/she would like to run. The following is SCL code, which supports this object:

```
TABLIST:
 call notify('tablist',
      '_get_value_',prjds);
 call notify('tablist',
      '_get_nselect_',numselected);
 call notify('tablist',
  '_selected_',numselected,row_num);

/*put selections into temp dataset*/
  submit continue;
     data gettab(drop=n);
     set &dsname;
     retain n 0;
       n=n+1;
       if n eq &row_num then output;
     run;

     proc append base=listtab
                 data=gettab;
     run;

     data gettab;
     set  gettab;
         delete;
     run;
  endsubmit;
  selid=getiteml(prjds,1);
  call putlist(selid);
RETURN;
```

Initially, this object doesn't display anything. However, there are three objects (Push Button located below the List Box), one for Summary tables, one for Ad-hoc tables, and one for Listings. By pushing any one of them, a different list of programs will appear in the List Box. The following is SCL code for the Summary button:

```
SUMTAB: /*act on "Summary" button*/
  submit continue;
  /* if dataest exists, delete it */
     %checkds;
  endsubmit;
  prjds=open('saftab');
   call notify('TABLIST',
               '_update_');
  dsname='saftab';
  pgm='STATSUM';
RETURN;
```

The %CHECKDS macro is defined in the INIT section and is used to delete the LISTTAB dataset if it exists as mentioned above. When selected, a dataset containing a list of programs that produce Safety tables will be opened and that information displayed in the List Box object. Also, two variables (dsname and pgm) are defined and used in the SCL behind the CREATE icon.

From the selection list, a user can select any one or all of the programs that appear. To the right of the List Box are four ICON objects. The first ICON (Create) allows the user to run the selected tables. The following is a section of SCL code taken from the TABSEL FRAME which gets executed when the CREATE icon selected. Due to the size of this section of SCL, only a portion appears in this paper:

```
/*create selected tables in batch */
CREATE:
submit continue;
options mprint;
%macro outsafe;

/*********************************/
 section of code removed which
 assigns the following macros:
 BATCHFIL,PGMFOLD,LSTFOLD,LOGFIL,
 LOGDIR, and CONFIG
/*********************************/

%global protlist prtlst viewlib
datalib comfile;

options nodate nocenter;

proc printto print="&batchfil" new;
/* begin outputting 'batch' file */
title;









/* write out program for */
/* BATCH processing      */
data _null;
set &listtab end=eof;
file print noprint;
     if _n_ eq 1 then do;
        .
```

```
        .
        .
    end;
run;

* use these options to launch a  *
* a remote session without tying *
* up current session             *;
options noxwait noxsync;

* X command runs external command, *
* in this case a SAS job (in batch)*
x "l:\sas612\sas\sas -sysin
   &batchfil
   -log &logfil
   -config &config";

proc printto;
run;
%mend outsafe;

%outsafe;

endsubmit;

call
notify('tablist','_UNSELECT_ALL_');
RETURN;
```

When the desired programs have been selected and the CREATE button clicked, a series of events occur. First, a 'batch' SAS program is created and written to a study level folder. This program is nothing more than a set of %INCLUDE statements for every program selected. Next, a new SAS batch process is initiated and runs the 'batch' program. Finally, the .LOG file for the 'batch' program as well as the .LOG files for every program selected are written to a study level folder. The .LST file for every program selected is also written to a study level folder.

View Table Frame

When the VIEW icon is selected the following command is executed under OBJECT ATTRIBUTES/COMMAND PROCESSING: afappl c=in1.tap.viewtb.frame. This activates a new FRAME that enables users to view the output (See Appendix III). The largest object that is on this FRAME is called External File Viewer. This object has both vertical and horizontal scroll bars that enable users to view the entire table or listing. As shown below, the majority of the SCL to display output is located in the INIT section.

```
init:
  /* point to Table of Tables
dataset of selected protocol */
  submit continue;
    %macro getlib;
    data _null_;
    set tap.&user;
        call
symput('thpy',compress(thpy));
        call
symput('proj',compress(proj));
        call
symput('prot',compress(prot));
    run;
    %mend getlib;
    %getlib;
```

```
/* get Table of Tables data and */
/* populate screen              */
    data emptytb textdta;
    length row $140;
        row='';
        output emptytb;
        output textdta;
     run;
     filename _all_ clear;
  endsubmit;
  thpy=symget('thpy');
  proj=symget('proj');
  prot=symget('prot');
  newtabs='r:\de_&thpy\clean
            \&proj&prot\lst';

  submit continue;
       data _null_;
       set listtab;
         if _n_ eq &nexttab then do;
            call symput('rptname',
                 compress(program));
         end;
       run;
       filename files "&newtabs\
                    &&rptname..lst";
       proc datasets library=work;
            delete textdta;
       run;
  endsubmit;
  call notify('viewtab','_set_file_'
            ,'files','c');
return;
```

As you can see in the SUBMIT block, a file reference is defined. This reference is then used by the CALL NOTIFY function to view the output on the screen. At the present time, only one table can be selected at a time from the table selection FRAME to view on the External File Viewer. However, in the future, I would like to be able to give users the opportunity to select several tables, and then be able to flip from one table to another within the External File Viewer object. There are two other objects on this FRAME and they are Push Button. One button (Previous Screen) takes you back to the previous FRAME. With this object, the following SCL is executed:

```
prevscr:
   submit continue;
       proc datasets library=work;
            delete listtab;
       run;
   endsubmit;
   call execcmd("Cancel");
return;
```

Here the SUBMIT block deletes the dataset LISTTAB which contains the name of the output to be viewed, and the CALL EXECCMD function closes the FRAME. The other button simply allows the user to exit TAP. Unlike the Previous Screen button there is no SCL code for the EXIT button.

The following is SCL code, which supports the PRINT icon. This code will send to the printer every table that you select (if you select 10 tables it will send 10 tables to the printer):

```
PRINT:
  thpy=symget('thpy');
  proj=symget('proj');
```

```
  prot=symget('prot');
  newtabs='r:\de_' || thpy ||
'\clean\' || proj || prot || '\lst';

  submit continue;
      data prttab(keep=table);
      set &listtab;
      length table $30;
  table=compress(program) || '.lst';
       run;
  endsubmit;

  tablist=open('prttab');
  do while (fetch(tablist) ne -1);
     call set(tablist);

table=getvarc(tablist,varnum(tablist
,'table'));
     file=newtabs || '\' || table;
     rc=filename('table',file);
     rc=preview('clear');
     rc=preview('include','table');
     rc=preview('print');
  end;
  rc=close(tablist);
  call
notify('tablist','_UNSELECT_ALL_');
RETURN;
```

In the SUBMIT block, the .LST extension is added to the program name (the output file logs and tables have the same name as the program name).

Conclusion

TAP is a simple yet powerful application that was built with Clinical Research in mind. However, with some minor modifications (i.e. remove Push Button and Text Entry objects from MAIN.FRAME) TAP can be used for any purpose so long as your intent is to run custom designed SAS programs to fulfill your table production needs. A few of the advantages to using TAP vs the Program Editor include not having to click your way through folder after folder to get to where your programs are stored. Since programs are accessed via the mouse, there is never a worry about programs being accidentally modified and/or saved back to the wrong folder. Multiple programs can be selected and executed in BATCH verses running one program at a time. The .LOG and .LST files are immediately written out to a folder. Tables and listings can be viewed on the screen. Finally, multiple tables/listings can be sent to a printer. The bottom line is all you have to do is just point and click and let TAP do the rest.

SAS provides 13 entries and 41 objects that can be used to build an application. TAP uses just two entries (FRAME and SCL), and six objects (Graphic Text, Push Button, Text Entry, List Box, Icon, and External File Viewer). As you evaluate all of the options that are available to you, it is not always easy to make the appropriate selection, and it is sometimes difficult to put the various pieces together. For example, knowing when SCL for an object is appropriate. Hopefull, with this paper you have discovered, with just a few ENTRIES and OBJECTS, how to create a wide variety of applications quickly and easily.

References

SAS® Language: Reference 6, First Edition.

SAS/AF® Software Usage and Reference, Version 6, First Edition.

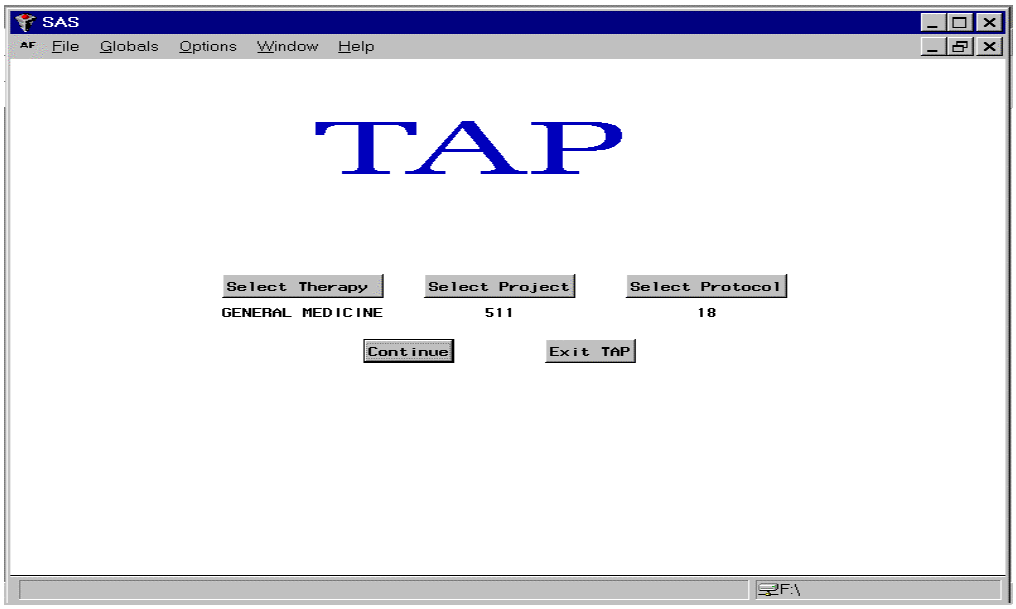
SAS® Screen Control Language, Usage and Reference, Version 6, First Edition

SAS® Technical Report P-216, SAS/AF® Software, SAS/FSP® Software, and SAS® Screen Control Language: Changes and Enhancements, Release 6.07.

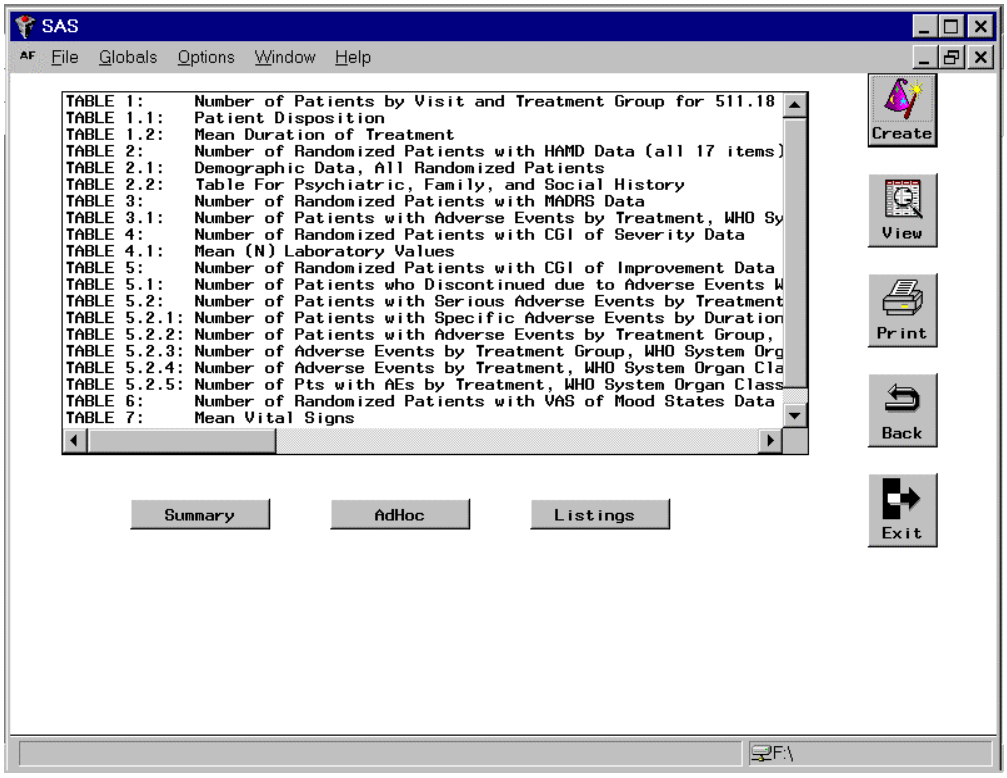
For further information contact:

David M. Hartman
Boehringer Ingelheim Pharmaceuticals, Inc.
900 Ridgebury Road
P.O. Box 368
Ridgefield, CT 06877-0368

APPENDIX I: The first FRAME to appear for TAP



APPENDIX II: The “Work Horse” of this application



Appendix III: The “VIEW Output” Frame