Paper 45-26

Dynamic Applications To Satisfy An Ever Changing World

Bernard R. Poisson, U.S. Air Force, San Antonio, TX

ABSTRACT

Today, business and government organizations thrive on information. They are totally dependent on their continued ability to rapidly and efficiently access information in order to function. Remove or reduce the capacity to access needed information and the organization slows to an ebb or worse, stops dead in its tracks. The changing nature of information required, and the need to leverage costs while at the same time increasing the availability of information, makes the design and coding of applications a progressively more daunting task. Using a few powerful SAS© functions and a modest change in design approach, you can create applications capable of satisfying the continually changing information needs of your organization, decrease costs, and increase information availability. Two equally important derived benefits that we cannot overlook are those of increased application portability, and quick recovery after system failures.

INTRODUCTION

The Air Force Surveys Branch is required to perform between 12 and 15 telephone-based surveys called "Computer Assisted Telephone Interviews" or CATIs per year. The average number of personnel that must be successfully contacted and interviewed for a survey project of this nature to be statistically representative of the Air Force (AF) as a whole is approximately 400. These projects normally arrive only two to three weeks before the results are required in a formal report to the Air Staff or Air Force Chief of Staff (CSAF) at the Pentagon. The ability to conduct the data collection phase efficiently is critical. This paper describes and contrasts the manual vs. automated approaches to call tracking. It then demonstrates how to achieve the same degree of flexibility in just about any application where source data continually changes.

THE MISSION

Typically, the branch chief receives a no-notice tasking via telephone from the Air Staff on a much needed survey project for the CSAF. As an example, the CSAF must know how a change in policy may or is affecting the Air Force so he is prepared when he testifies before Congress in 30 days. The table below identifies the major tasks that must be accomplished in order to meet the mission.

Task	Time Required
Receive tasking. Identify purpose. Identify theatres of interest, target population (Civilian/Military/Officer/Enlisted), and other sample stratifications	2 Days
Create sample-generation code, create sample, verify accuracy	2 Days
Collect data. Successfully contact 400 personnel ensuring critical stratification categories are satisfied	5 Days
Perform analysis on collected data and create report outline	2-4 Days
Create final report	1-2 Days
Forward report through chain-of-command for review and coordination through the Pentagon to its destination.	5-10 Days

CONTRASTING DATA COLLECTION METHODS

What occurs with the sample data set from this point is the principal topic of this discussion. The table below identifies the capabilities provided by the manual method of call tracking and the SAS©-based 'CATI Survey Manager' application.

Capabilities Provided	Manual Method	CATI Survey Mgr.
Tracking method	Paper listings	Interactive query
Standardized status indicators	No	Yes
Caller performance tracking	Vague, error prone	Automatic, non-modifiable audit-trail
Stratification tracking	Major Stratifications on listings	Any data element
Status tracking	Major Stratifications on listings	Any data element + 8 Status Controls
Caller Interchangeability	Tedious, time-consuming	Transparent
Filter/Select by:		
Time-Zone	Listing pages	8 Zones
Call Status	No	8 Controls
Rank	No	19 Grades
Gender	No	Yes
Availability Date	No	Yes
Name	No	Yes
Social Sec. #	No	Yes
Data-driven stratifications	No	Yes
Duplicate calls placed	Several	None
Resultant data directly ready for analysis	No	Yes
Data snapshots possible	No	Yes

The table above clearly demonstrates that the CATI Survey Manager provides a much more efficient, controllable, and intelligent approach to telephone-based data collection efforts. The scalability of the application (Intranet vs. Internet) is restricted only by its accessibility on the network.

'CATI SURVEY MANAGER' CONFIGURATION AND OPERATION

Required Setup Procedures: Since a data-driven application will be used to perform tracking, there are a few simple configuration items that must be performed by the analyst before the initial call can be placed. They are:

1. The analyst calls up their web-browser on their desktop system and clicks the shortcut link to 'CATI Survey Manager'. The actual call to the broker invoked by the shortcut is:
   'http://path-to-broker/broker.exe?_program=catiprg.scl'

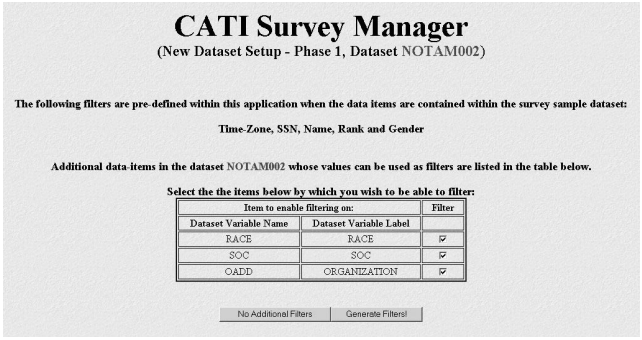
Notice that no arguments are passed to the SAS© broker except for the '_program' argument, the minimum necessary for the

application server to invoke the appropriate program.
The analyst is greeted with the following screen:

Data set Configuration Screen #1

No reference has been made to any mandatory data set specifications except for the 5 'expected' data items included by default. This is because the application can literally handle ANY SAS© data set!

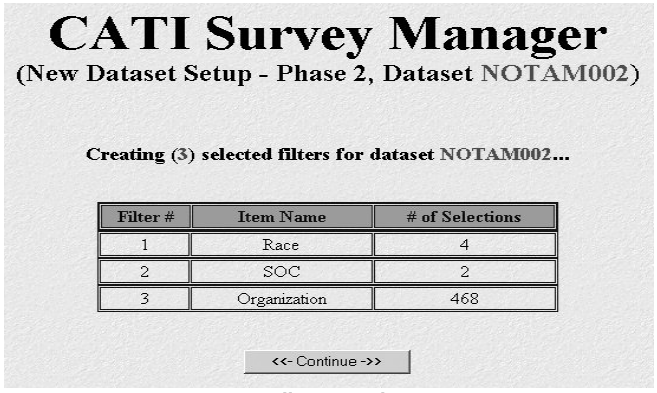
2. After entering the name of the sample data set, the analyst clicks on the 'Continue' button. This transmits the necessary information to the application via the broker. When the application server receives the data set name, the application interrogates the data set and creates a list of column-names, column labels, and the number of unique values each column contains. The following screen is displayed:



Data set Configuration Screen #2

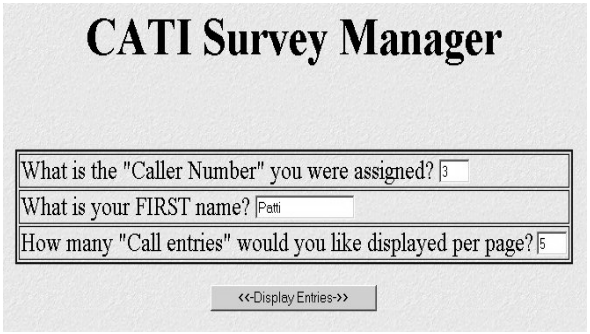
This dynamically generated HTML screen provides the analyst everything they need to know about the sample in an intuitive fashion. They are given the opportunity to specify additional items by name/label contained within the sample data set to be used for filtering of caller data. Checking the corresponding check-box next to the data-item's name/label pair includes the item in the dynamic filter list.

The following screen is generated when the analyst clicks 'Generate Filters', showing completion of the necessary configuration of the application for this CATI.



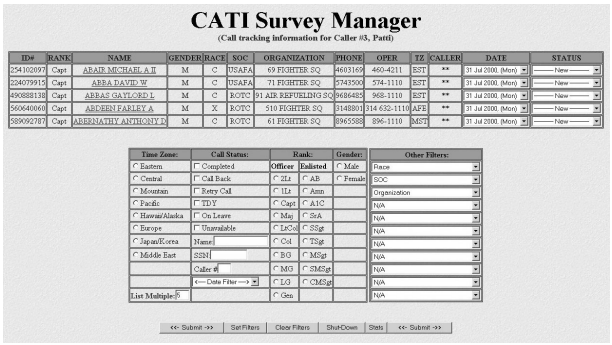
Data set Filter Creation Status

The 'CATI Survey Manager' in action: Each time the application is invoked from a 'new browser session', the caller receives the following screen:



Initial Browser Session Configuration Screen

The caller enters the appropriate identification information and clicks 'Continue'. Now that the application has all necessary information, the caller is presented with the first screen of contact information as displayed below:



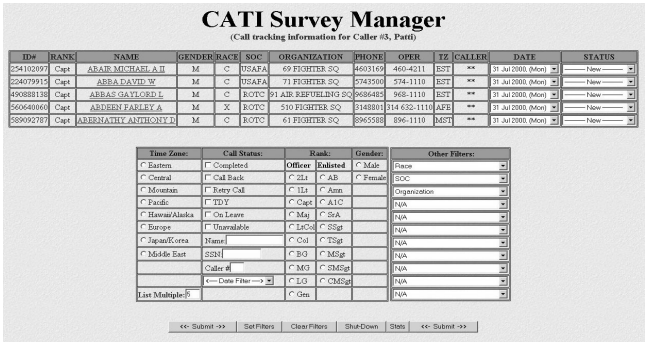
Contact Information/Filtering Screen

There are two sections in the application's output display. The top 'contact' section contains the contact information required to place the calls, and the bottom 'filter' section provides the caller with the controls necessary to very selectively filter and control the contact items displayed.

The top 'Contact' section: The data displayed in the top section are all contained within the sample data set. They contain the information necessary for callers placing the calls to gain the required understanding of the subject matter of the CATI conducted.

The data contained in the 'Name' field has a distinctly different appearance than the rest because it is a hypertext link leading to a WEB-based survey. The link also includes information necessary for the analysts to track the demographic information of each respondent and is included in the response data set.

The fields 'Caller', 'Date', and 'Status' are added onto the sample data set during the initial configuration run of the application and enable the data set to be used as a tracking mechanism. The 'Caller' field data is automatically entered for each observation where the 'Status' field has been modified and is based on the information provided by the caller on the initial run of the day. The date field keeps track of calls made to personnel that are not available but may be available within the data collection window. It allows for the selection of any weekday date during the next 13 days, and is used on subsequent days as a filter to target respondents. The 'Status' field enables the caller to indicate one of eight possible values for each call placed.



The bottom 'Filter' section: The filtering options are presented to the caller in two distinct regions. In the left region we observe the five 'default' filterable items: Time Zone, SSN, Name, Rank and Gender, corresponding to the 'required' data-items included in the sample data set. Along with these we see filters for Caller # and Date. Together these comprise the 'static filters'. During the 'configuration phase', the analyst was requested to indicate which 'additional' data items should be included for filtering of contact observations. The right region contains these 'dynamic' filter items in the form of user selectable drop-down lists. This approach was chosen because of the ability to display a data set containing data items without restriction. This capability can yield data elements/variables that contain literally hundreds of unique values and drop-down lists pose no display/selection restrictions.

All filters with the exception of 'Name' and 'SSN' are inclusive, meaning all selections are taken into consideration when filtering the data. To set, clear, or modify filtering combinations, the user simply selects the desired filtering options in combination with the appropriate function button on the display. 'Name' and 'SSN' searches are exclusive. Other filtering options are temporarily ignored, along with page display restrictions, to allow the display of all targeted individuals.

Once a caller has completed placing calls for all of the personnel listed, they click on the 'Submit' button to perform the required update to the status of the listed contacts. This action causes the application to call itself (recursion) and the process starts over.

UNDER THE HOOD

From the WEB to SAS©: Applications using SAS© Intrnet normally utilize the SAS© 'Broker'. The broker reads an HTML page, identifies the named items and their associated values, and then passes this information during its call to a SAS© application server. As part of the application server invocation, the broker passes along the information it has found in name=value pairs.

All applications using the broker must have the 'ENTRY' statement as its first line such as: 'entry optional = params 8;'. This statement allows the WEB-based application to accept the incoming data stream passed by the broker in its call to the application. The 'params' argument in the entry statement above 'names' the SCL list built by SAS© when it encounters the statement. The power of SCL lists is harnessed by this type of application. Every item of data and information gathered and used by the application is encapsulated within an SCL list, providing the developer a single source for configuration, state and operation information, as well as the actual data provided by the users.

Creating Selectable Filter Items: The code that allows the analyst to pick and choose the additional data items to be used as filters follows:

```
DS = OPEN(DATA SET,'U');
COLCOUNT = ATTRN(DS,'NVARS'); ← Get number of data items in the data set
CALL SET(DS);
COUNT=0;
VARNAME    = MAKELIST(); ← Create SCL Lists to store variable Name, Label and Length of data items
VARLABEL   = MAKELIST();
VARLENGTH  = MAKELIST();

/*** Get Data Item Count & Load Names, Labels, and Lengths Into SCL Lists ***/

DO INDEX = 1 TO COLCOUNT; ← Loop through the data set and add Name, Label, and Length information to the 3 SCL Lists
  RC=INSERTC(VARNAME,  UPCASE(VARNAME(DS, INDEX)), -1, UPCASE(VARNAME(DS, INDEX)));
  RC=INSERTC(VARLABEL, UPCASE(VARLABEL(DS, INDEX)), -1, UPCASE(VARLABEL(DS, INDEX)));
  RC=INSERTC(VARLENGTH, VARLEN(DS, INDEX), -1, VARLEN(DS, INDEX));
END;
```

Creating Selected Filter Items: The code that generates the data item filters follows:

```
DS = OPEN(DATA SET,'U');
COLCOUNT = ATTRN(DS,'NVARS');

FILTERCOUNT = 0;
DO INDEX = 1 TO COLCOUNT;
  RC=CLEARLIST(VARVALS);
  IF NAMEDITEM(PARAMS, 'FILTER'||INDEX) THEN ← If the analyst indicated to use this data item as a filter, then create the filter
    DO;
      FILTERCOUNT = FILTERCOUNT +1;
      RC=LVARLEVEL(DS, VARNAME(DS, INDEX), NLEVELS, VARVALS); ← Get unique values and place them in SCL List
      RC=SORTLIST(VARVALS); ← Sort the SCL List in ascending order for display (makes it easy for user to locate desired value)
      DO X = 1 TO LISTLEN(VARVALS); ← Loop through the values for this data item to determine the longest one (used to set up output display)
        IF LENGTH(TRIM(GETITEMC(VARVALS, X))) > MAXLENGTH THEN
          MAXLENGTH = LENGTH(TRIM(GETITEMC(VARVALS, X)));
      END;
      NEWLIST = VARNAME(DS,INDEX);
      NEWNAME=NEWLIST;
      NEWLIST=COPYLIST(VARVALS);
      FLTRVARS = INSERTL(FLTRVARS, NEWLIST, -1, NEWNAME); ← Insert the SCL List containing unique items into a master filter SCL List, by name
      VARLABS = INSERTC(VARLABS, VARLABEL(DS, INDEX), -1);
    END;
END;

RC=SORT(DS, 'AMF');
RC=CLOSE(DS);

FLTRVARS = INSERTL(FLTRVARS, VARLABS, -1, 'VARLABS');
FLTRVARS = INSERTC(FLTRVARS, MAXLENGTH, -1, 'MAXLENGTH'); ← Store the length of the longest unique item value
RC=SAVELIST('CATALOG','CATIDATA.SLIST.' || INDATA || '.SLIST', FLTRVARS); ← Save filters for subsequent calls
```

Creating Application State and Control Parameters: The recursive nature of this application requires several pieces of information to be present so that the application knows exactly what's going on at all times. The simplest way to do this is to transparently imbed the needed parameters into the actual HTML displayed to the user. This has the added benefit of providing an extremely dynamic nature to the application since the settings are localized. Here's how its done:

```
RC=FPUT(WEB,'<FORM METHOD="POST"
```

```
TARGET="_PARENT" ACTION="/PATH-TO-
BROKER/BROKER.EXE?">');
  RC=FPUT(WEB,'<INPUT  TYPE="HIDDEN" NAME="_DEBUG"
VALUE="'||DEBUG||'">');
  RC=FPUT(WEB,'<INPUT TYPE="HIDDEN" NAME="_PROGRAM"
VALUE="CATIPROG.PROGRAMS.CATI.SCL">');
  RC=FPUT(WEB,'<INPUT TYPE="HIDDEN" NAME="DATA SET"
VALUE="'||INDATA||'">');
  RC=FPUT(WEB,'<INPUT TYPE="HIDDEN" NAME="NEWCALLER"
VALUE="'||GETNITEMC(PARAMS,'NEWCALLER')||'">');
  RC=FPUT(WEB,'<INPUT TYPE="HIDDEN"
NAME="NEWCALLERNAME"
VALUE="'||GETNITEMC(PARAMS,'NEWCALLERNAME')||'">');
  RC=FPUT(WEB,'<INPUT TYPE="HIDDEN" NAME="LISTNUM1"
VALUE="'||GETNITEMC(PARAMS, 'LISTNUM1')||'">');
```

THE BENEFITS

The benefits of data-driven, WEB-based applications are astounding. They include:

- No software maintenance costs/time: As the data the application must access changes, the application changes with it. The data independence coded within the application ensures this.
- Reduced cost in data access: Using a WEB approach means that the organization no longer has to license a copy of SAS© for each workstation requiring access to SAS© data. (There goes the SUGI invite).
- Increased information availability: This compliments the previous benefit. The WEB approach only requires the user have a WEB browser, access to the Inter/Intranet, and the SAS© Application Server. This is a tremendous benefit for those who have to go on the road.
- Centralized control of data: All data is hosted and managed from one SAS© Application Server, this removes the problem of synchronizing data between systems. Everyone always reads from the 'same sheet of music'.
- Application Portability: Transfer between platforms simply requires a re-compile of the source file using the SAS© system on the new host to make the application compatible with the new platform.
- Quick, Simple Recovery: This type of application is easy to code and manage in a single file. Recovery then consists of recompiling a copy of the source file to bring the application back to an operational state.

CONCLUSION

The ever-changing information needs of organizations is rapidly requiring more and more applications capable of coping with these changing needs. The ability to create this type of dynamic application exists within the SAS© System. By simply thinking in more generic terms when dealing with data and how we interact with it and applying a few SAS© functions, we can empower our applications to keep up with this dynamic world and meet the challenge!

CONTACT INFORMATION

MSgt. Bernard R. Poisson
Headquarters Air Force Personnel Center/DPSAS
550 C. Street West, Suite #35
Randolph AFB, TX 78150
(210) 565-2448, 7-4 (CST)
E-Mail: bernard.poisson@afpc.randolph.af.mil