

Paper CC-027

Order from Chaos: Using the Power of SAS® to Transform Audit Trail Data

Yun Mai, Susan Myers, Nanthini Ganapathi, Vorapranee Wickelgren

ABSTRACT

As survey science has turned to computer-assisted interviewing (CAI) for primary data collection, the keystroke or Audit Trail data, the automatic by-product of CAI software systems, has received increasing attention. Such data contain valuable information about the interview process, and includes responses as well as timing, audio settings, and literally every keystroke made by the respondent. How to efficiently convert these semi-structured Audit Trail ASCII files into user-friendly database files such as SAS® datasets remains challenging.

To process the voluminous Audit Trail data, RTI has developed a SAS® system to convert the ASCII files into SAS® datasets. This system expands the accessibility of text files to a wide range of users for survey analysis. The resulting data are used in many ways, such as to assess the performance of field interviewers, to assure the accuracy of the survey instrument, and to monitor the time required to complete the interview.

This paper has three objectives. First, we give an overview of our automated Audit Trail system. We highlight some of the challenges faced in developing this system and how we addressed them. Second, we illustrate several ways the resulting datasets have been analyzed. Third, we discuss potential applications and future developments of this automated information processing system.

BACKGROUND

The National Survey on Drug Use and Health (NSDUH) is a nationwide survey conducted annually since 1988 by RTI International for the Substance Abuse and Mental Health Services Administration (SAMHSA). The NSDUH is the primary source of information on the prevalence, patterns, and consequences of alcohol, tobacco, and illegal drug use and abuse in the general U.S. civilian non-institutionalized population age 12 and older. The survey collects interview data from approximately 67,500 respondents in 180,000 sampled households across the United States. There are more than 700 field interviewers working on the survey.

Large surveys generally use Computer-Assisted Interviewing (CAI) for data collection and NSDUH was a pioneer in this regard. The CAI software for the NSDUH survey, written in Blaise, is designed to collect behind-the-scene or “audit trail” data, in addition to the survey data. The audit trail data include information such as systems settings, the number of times that a question is visited, answer entered to each question, time spent on each question, time spent on each module. The complete keystroke sequence during the data collection process is saved. A separate audit trail file is generated for every interview session. Multiple audit trails are generated if a user breaks off and restarts the questionnaire. The various time reports generated from the audit trail data help to estimate respondent burden, to understand respondent difficulties, to identify interviewer falsification and to improve the overall survey design. However, processing these voluminous, detailed, semi-structured text files is time-consuming and thus may limit timely use and analysis of audit trail data.

THE CHALLENGES

The annual NSDUH survey generates about 40,000 audit trail data text files (based on 2005-2007 survey data) that need to be processed. Each text file contains information, ranging from 300 to 1000 data fields. An audit trail data file is a ‘~’ delimited text file (figure 1) that includes information on a questionnaire field, the time a field was entered, the value of the entry, the time of the field departure, the state of some system settings, and the method of navigation out of the field. The file also captures how the interview is ended. An interview can end as complete or interruption (e.g., exit due to interview breakoff, hardware failure or error - CAI crash) (figure 2). Before 2004, the massive amount of audit trail text files were processed using Perl language, then converted to SAS® dataset for analyses. The process was not done on a regular basis and took an inordinate time to run.

THE SOLUTION

To meet the increasing demand for easy access to audit trail data, we have developed an audit trail data processing system that automatically transfers audit trail data from its native text format to a structured SAS® dataset each day. The system calculates the elapsed time in seconds for each question, each questionnaire module, and

various special request sections. The datasets also capture interview interruption information, such as the specific module and field where the breakoff occurred. Additional information in the data includes frequency of a specific question visited, the corresponding values entered for the question, and the range of values entered for the question. The following section describes few key steps involved in our audit trail data processing system. Fig. 3 below shows the overall system design on the left and data flow diagram on the right.

Macro Get Files

The audit trail files are stored in directories named by the following convention:
DirectoryName\Year\Data\Month\MonthDate

We created a named pipe (figure 4) to move all files from the current day into a variable called 'infile' and then put all those filenames into a table (tblFileName). The process then loops through the table and processes each file in the table. Each of the files we are interested in have the '.adt' extension so our code parses out these filenames and prepares them for our infile statement in Figure 5.

Figure 1: Example of an audit trail file

```
"Enter Interview:"~"XXXXXXX"
"MODULE1"~"03-06-2006 09:35:49"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:50"
"MODULE2"~"03-06-2006 09:35:50"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:51"
"MODULE3.FIELD1"~"03-06-2006 09:35:51"~""~"AUDIO ON"~"NO COMMENT"~"Exit"~"03-06-2006 09:35:53"
"Leave Interview:"~"XXXXXXX"~"COMPLETE"
```

Figure 2: Example of different audit trail file ending

- **Complete**

```
"Enter Interview:"~"XXXXXXX"
"MODULE1"~"03-06-2006 09:35:49"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:50"
"MODULE2"~"03-06-2006 09:35:50"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:51"
"MODULE3.FIELD1"~"03-06-2006 09:35:51"~""~"AUDIO ON"~"NO COMMENT"~"Exit"~"03-06-2006 09:35:53"
"Leave Interview:"~"XXXXXXX"~"Complete"
```

- **Breakoff**

```
"Enter Interview:"~"XXXXXXX"
"MODULE1"~"03-06-2006 09:35:49"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:50"
"MODULE2"~"03-06-2006 09:35:50"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:51"
"MODULE3.FIELD1"~"03-06-2006 09:35:51"~""~"AUDIO ON"~"NO COMMENT"~"Exit"~"03-06-2006 09:35:53"
"Leave Interview:"~"XXXXXXX"~"Breakoff "
```

- **Cai crash**

```
"Enter Interview:"~"XXXXXXX"
"MODULE1"~"03-06-2006 09:35:49"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:50"
"MODULE2"~"03-06-2006 09:35:50"~"1"~"AUDIO ON"~"NO COMMENT"~"[ENTER]"~"03-06-2006 09:35:51"
"MODULE3.FIELD1"~"03-06-2006 09:35:51"~""~"AUDIO ON"~"NO COMMENT"~"Exit"~"03-06-2006 09:35:53"
```

Figure 3: Audit Trail Data Processing System Flow Chart

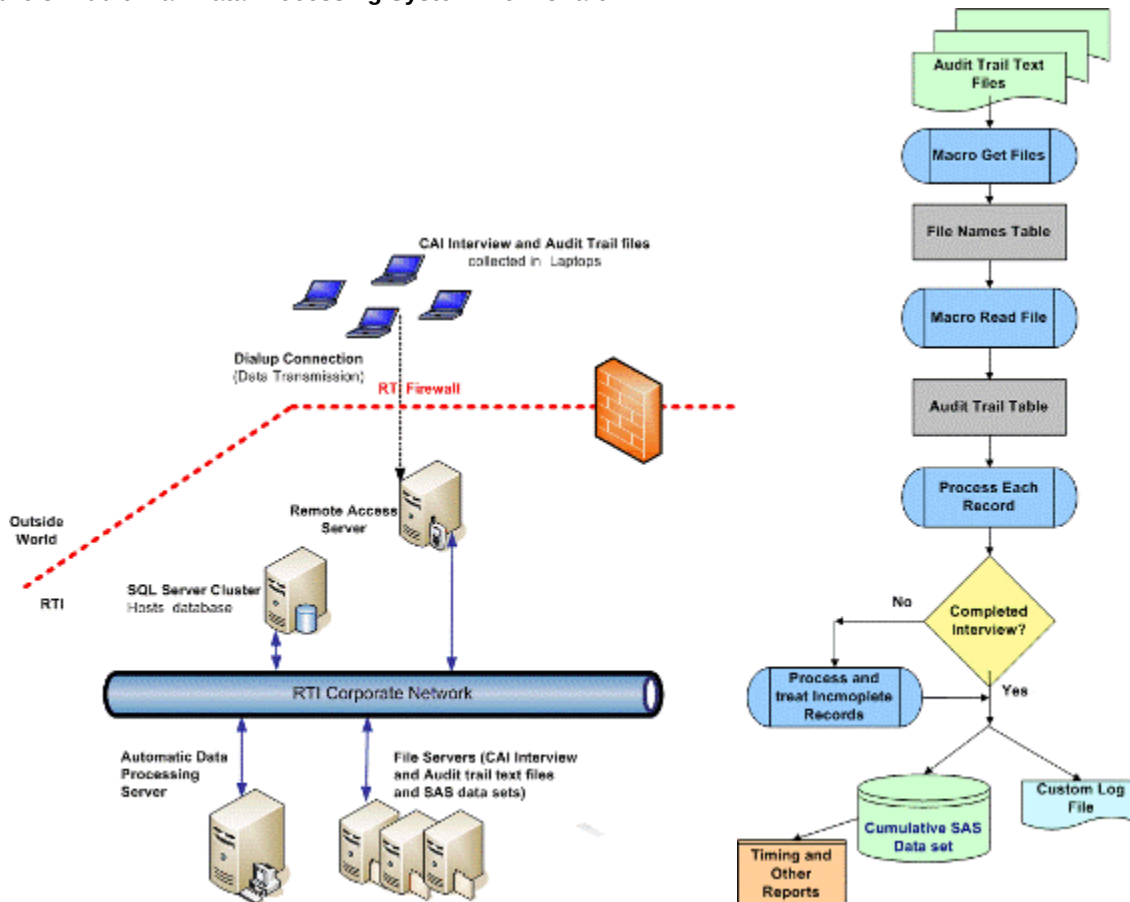


Figure 4: Get file names

```

FPath="dir DirectoryName\||trim(yr)||\data\||trim(mon)||"\||trim(mon)||d" /s";
filename in pipe "&FPath";

data tblFileName (keep = dir mem );
  infile inFile truncover ;
  input line $char132. ;
  size = scan(line, 4, ' ');
  if size ne '0';
  do;
    if index(upcase(line), "DIRECTORY") then
      do ;
        dir = substr ( line , 14 ) ;
        retain dir ;
      end ;
    list ;
    if index(upcase(line), ".ADT") ;
      mem = substr ( line , 39 ) ;
    end;
  run ;

```

Figure 5: macro to read in each file into a table

```
%Macro ReadOneFile(strFileName);  
/*input file*/  
    data data(compress=yes);  
        infile "&strFileName|" firstobs=1 dsd dlm='~' trunccover lrecl=800;  
        input Var1 :$250.  
                Var2 :$30.  
                Var3 :$30.  
                Var4 :$30.  
                Var5 :$250.  
                Var6 :$30.  
                Var7 :$30.;  
    run;  
%Mend ReadOneFile;
```

DETERMINE THE BEGINNING AND ENDING OF EACH RECORD IN A FILE

One Audit trail file can contain several different interview records transmitted by a single field interviewer. A normal record starts with the keywords 'Enter Interview:' and ends with keywords 'Leave Interview:' followed by the interview ID, then the state of the interview (COMPLETE OR BREAKOFF) unless the record abruptly ends with a hardware or power failure – CAI crash. We use the *rxparse* statement to detect the beginning and end of each normal record.

```
BeginRc=rxparse("$p 'Enter Interview:');  
EndRc= rxparse("$p 'Leave Interview:');|
```

We assign unique idNum to each interview within the file and assign counter to the entire file. The elapse time for each interview and each field within an interview are calculated by using custom Macros.

The CAI crash records (interviews) cannot be processed with the same macro, since those records are missing the end of record keyword line ("Leave Interview:"~"XXXXXXX"~" CaiCrash "). The macro will not know where to start and where to end. With less than 0.5% total records are interviews abruptly ended by hardware or power failure, we first search for where the failure occurred, and then manually add the end of record keyword line ("Leave Interview") at the appropriate location so that the same macro can be used for all files

The following code looks at the end of the file, and determines whether the record ended with a CAI crash.

```
data Lastrecd;  
set t1 end=last;  
if last;  
    rx=rxparse("$p 'Leave Interview:');  
    position2=rxmatch(rx,x);  
    if position2 = 0 then do;  
        m="crashfile";  
        call symput('LastCrash', 1);  
    end;  
run;|
```

If a CAI crash occurs in the middle of the file, a series of "proc sql" statements are used to determine the ending of first record and the beginning of the next record. Then an end of record keyword line, "Leave Interview" is inserted between the records.

```

proc sql;
  create table midcrashtbl1 as select distinct idNum, max(counter)
  as lastcounter from data group by idNum;
quit;

proc sql;
  create table midcrashtbl2 as select lastcounter from midcrashtbl1 inner
  join data on data.counter=midcrashtbl1.lastcounter and
  data.idnum=midcrashtbl1.idnum where mod(>'Leave Interview:');
quit;

```

After the CAI crash record is treated, the process calls the same macros to obtain all necessary information to a dataset.

The process also generates a custom log file (figure 6) to summarize the frequency of processed files or records each day as well as each month. This summary information may streamline the steps designed to calculate missing records during analysis. As seen in this sample we detect the process did not run on 6/29/2008. By looking at this log we can easily detect a problem and investigate the situation.

Figure 6: process log file

The SAS System				10:45 Friday, June 27, 2008	20
toDay	Total Record Num	Today Record Num	Today FileNum		
06/27/2008	1763	26	15		
The SAS System				10:45 Saturday, June 28, 2008	15
toDay	Total Record Num	Today Record Num	Today FileNum		
06/28/2008	1774	11	10		
The SAS System				10:45 Monday, June 30, 2008	10
toDay	Total Record Num	Today Record Num	Today FileNum		
06/30/2008	1781	7	5		

CURRENT USE

This section illustrates two of the most useful reports, timing Data Report and Breakoffs report generated from the structured SAS® dataset and explains their benefits.

1. TIMING DATA REPORT

In the CAI process, timestamps are captured at the beginning of each interview section or module. Since the audit trail captures the length of each individual screen, it has a clear advantage over the traditional CAI time stamp. The audit trail data may produce timing for any single question or any combination of screens as well as the entire interview. With this advantage, our current analysis focus is more on monitoring interactions between field interviewers and respondents and the amount of interview time spent on each interview (Table 1). The timing data is also used to monitor interviewer falsification. Any abnormal or outlier timings could indicate fraud. The timing data is also helpful to monitor the respondent burden. An abnormally long average time for a question could indicate the respondent's difficulty in understanding the question. Since our study is ongoing and new questions are added each year, we are able to track the timing for new or modified questions. The timing results for these areas of the instrument may get extra attention by analysts.

Audit trail data are captured each time an interviewer transmits and it is possible to get a complete interview over several transmissions and files. This scenario happens if multiple breakoffs are needed and the interview is

conducted spanning more than one day. To consolidate this type of interview into a single SAS® record for each interview, we must sum the variable times by interview id (questid) and field interviewer id (fiid).

```
proc sort data=key3;
  by questid fiid;
run;

proc means data=key3 noprint;
  by questid fiid;
  var date &tvvarlist;
  output out=keysum(drop=_type_ _freq_ date) max(date)=stddate sum=; run;
```

Table 1. 2007 NSDUH CAI Audit Trail Timing Data: Total Interview Time (Minutes) with FI Observation Section

<i>Age Category</i>	12-17			18 +		
<i>Year of Interest</i>	2005	2006	2007	2005	2006	2007
Sample Used in Analysis	22,492	22,814	22,371	45,552	44,753	45,236
Missing/Extreme Records	42	57	62	222	178	201
Summary Statistics (Minutes)¹						
Mean	63.4	64.1	63.3	61.4	63.4	62.5
Variance	266.4	269.9	272.1	334.4	357.9	352.1
Standard Deviation	16.3	16.4	16.5	18.3	18.9	18.8
<i>Quartiles</i>						
Maximum	240.0	234.5	236.1	239.0	240.0	235.2
Q3	72.2	72.8	71.8	70.2	72.5	71.3
Median	61.1	61.9	60.9	58.0	60.0	58.9
Q1	52.0	52.7	51.9	48.7	50.2	49.6
Minimum	30.0	30.0	30.0	30.0	30.0	30.0
Range	210.0	204.5	206.1	209.0	210.0	205.2
Mode	60.9	56.3	54.9	50.2	54.2	51.6
<i>Percentiles</i>						
99%	112.0	114.5	114.0	121.5	125.8	125.2
90%	83.9	84.8	83.8	84.6	87.6	86.1
10%	45.1	45.6	45.2	42.1	43.4	43.0
5%	41.5	42.0	41.4	38.8	40.1	39.7
1%	35.9	36.1	35.7	33.6	34.6	34.4
<i>Extremes</i>						
Highest	240.0	234.5	236.1	239.0	240.0	235.2
Lowest	30.4	30.3	30.3	30.0	30.0	30.1

2. BREAKOFFS

The audit trail data may be used to summarize patterns of breakoffs occurred during an interview. The number of breakoffs associated with individual field interviewers is useful to detect data errors and data falsification. The locations in the questionnaire where breakoffs occurred during an interview may suggest that respondents may have difficulties in a specific question. Tables 2 and 3 below taken from "Analyzing Audit Trails in NSDUH by Penne, M.A., J. Snodgrass, and P. Barker (2005)" –Ref 1, are examples of break off reports based on 2002 survey data that show further details about break off interviews.

Table 2. Unweighted Distribution of Breakoffs Summary

Total Audit Trail Files	68,268
Total Breakoff Instances	1,563 (2.3%)
Total Respondents	1,411
Single Instance	1,293 (91.6%)
Two Instances	92 (6.5%)
Three Instances	20 (1.4%)
Four Instances	4 (0.3%)
Five Instances	2 (0.1%)
Total Field Interviewers	523

Table 3. Distribution of Breakoffs, by Location in Interview

Location in Interview	N	Percent of Total Breakoffs
Total	1,563	100.0
Field Interviewer Administered	926	59.24
Respondent Interaction	543	34.74
No Respondent Interaction	383	24.50
ACASI	631	40.37
Tutorial	189	12.09
Main Core Drugs	102	6.53
Nonmedical		
Psychotherapeutic Drugs	52	3.33
Noncore Sections	288	15.43
Systems Crash	6	0.38

ACASI = audio computer-assisted self-interviewing.

NOTE: Respondent Interaction refers to modules administered to the respondent by the interviewer. No Respondent Interaction includes such modules as the interview debriefing.

FUTURE DIRECTION

Audit Trail data are packed with information. We have concentrated our analyses on the timing aspects thus far. Analyzing respondent and FI navigation through the instrument, counting all backups in an individual questionnaire, and studying lengths of backup paths are some of the potential new analyses that could be performed on the audit trail data. These would require additional data processing to obtain further information from the original audit trail files. Our goal is to expand the current system, to make it more flexible, and to make fuller use of information captured in the audit trail data files.

CONCLUSION

The Audit Trail data processing system is a dynamic, data driven tool. It efficiently gathers and converts data from thousands of text files into an easily analyzable SAS® dataset. We have used the traditional benefits of SAS® in analyzing large data sets to build a system for analyzing audit trail data from CAI surveys. Most of the business logic is implemented in SAS® data steps, proc SQL steps and procedures like proc MEANS from Base SAS® package. A special mention should be made about the efficiency of file management via "filepath" and execution of database queries via proc SQL. Our code modules are flexible and lend themselves easily for functional modification. We believe the improved performance of our system is coming from the high efficiency of the underlying SAS tools. We are planning to further expand our current system to meet increasing demand on audit trail data analysis.

REFERENCE

1. Penne, M.A., J. Snodgrass, and P. Barker (2005). Analyzing Audit Trails in NSDUH. In Kennet, J., and J.Gfroerer (Eds.), Evaluating and Improving Methods Used in the National Survey on Drug Use and Health. Rockville, MD: Substance Abuse and Mental Health Services Administration, Office of Applied Studies, DHHS Publication No. SMA 05-4044, Methodology Series M-5, pp. 105-120.

CONTACT INFORMATION

Yun Mai, yunmai@rti.org, 404-592-9420
Susan Myers, smyers@rti.org, 919-541-7441
Nanthini Ganapathi, nanthini@rti.org, 919-541-7143
Vorapranee Wickelgren, mvp@rti.org, 919-541-7096

Each author can be reached at this mailing address:

RTI International
P.O. Box 12194
Research Triangle Park, NC 27709-2194.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.