The Uses of Blaise Audit Trail Files at Statistics Canada

Éric Joyal, Statistics Canada

1. Abstract

The increasing use of computer-assisted data collection methods has provided a wide scope of ongoing and timely process data called "paradata". The "call transaction history" file created by the Blaise system contains a record for each call made by interviewers to contact the selected units and the "audit trail" which collects each interviewer's action during the interview process are two examples. Both files have become a tremendous source of information. Until recently, "paradata" were essentially used to monitor, evaluate and report survey progress using only a subset of available information. Over the last few years, many survey researchers have demonstrated the usefulness of paradata in the data collection context and have greatly improved the understanding of this complex and evolving process. In addition, paradata are used (and continue to be used) to identify strategic opportunities for data collection improvements for which active management and responsive design initiatives are good examples. At the same time, it is also well recognized by many survey researchers that paradata can also be of great methodological use in many other survey steps prior and after data collection, for example, questionnaire design, non-response adjustment or estimation.

The main objective of this paper is to provide an overview of the historical use of the audit trail file produced by the Blaise system at Statistics Canada. It covers activities such as monitoring, support, questionnaire design, improving and assessing the survey instrument as well as the survey process which includes both operational and methodological aspects.

2. Introduction

The Blaise audit trail data records an interviewer's interaction with the questionnaire. It shows us what fields the interview entered, what the interviewer did on the field, and how long each event took. Many special actions are also recorded, such as edits, making remarks, and changing languages. The audit trail file is produced by a dll that is included in Blaise. The raw file, called an adt file (because its file name extension is .adt), is textual, and does not directly support analysis.

Audit trail data was first activated in 2001 for CAPI social surveys applications at Statistics Canada through the insistence of the Blaise development group. There was some resistance from the operation area mainly due to the enormous amount of collected data that have to be archived and managed. The Blaise development section agreed to take the responsibility of managing this paradata source.

At the start, the main developers' motivation was to remove internal block-level timer logic and fields from subject matter content blocks and use the audit trail files to derive timing information.

3. Timing Information

There have been several attempts trying to get timing information for our Blaise instrument. Usually, the requirement was to collect time data to provide block-level timing information, but the code could be extremely difficult to write, the requirements were not detailed in the block specifications, and the code could not be verified at any level of testing. The most frustrating thing about this awkward and un-testable code was that it would produce inaccurate data even if coded perfectly, because time spent when returning to a block which had already been "finished" would be charged to another block.

The developers knew that it would be far easier to extract block level timing data from audit trail data. The timing data would be accurate, and one time-extraction process could be used on all surveys.

3.1 Audit Trail Logs Analysis System (ATLAS)

Getting timing information from the ADT files was fairly easy and extremely efficient. It also didn't take too long to figure out that there was a lot more than accurate timing data that could be derived from the audit trail files. All this initial research and prototyping gave birth to ATLAS, our first audit trail processing system.

ATLAS was developed using Maniplus for the interface and Manipula to create the output. The first goal of ATLAS was to create timing reports at the field and block level. As reports were created, it generated some new ideas in terms of other information that we could extract and also about different ways to present the output. A prototyping approach was used, and we were very accommodating to any new feature and report requests we received. This flexibility came at a certain cost as we had to implement several layers of parameterisation to configure any new analysis.

ATLAS could be described as a three steps application. The first step is to set the survey specific parameters, which are entered through the Maniplus interface. Customization such as sub-level reporting and exclusion of blocks for the analysis was performed at that stage. At the next step, ATLAS generates and prepares the "survey specific" report-generating Manipula scripts. Finally, it runs the scripts to produce the reports. The outputs were mainly timing reports with a breakdown at the fields, the block and case level. ATLAS was also tracking edits and Non-responses.

During the early stage, ATLAS was able to extract of a lot of significant and useful information in a "minimal" amount of time and effort. On average ATLAS was able to process 20 interviewing minutes per second, which we assume was satisfying. But, as we expand our usage from pilot tests to surveys with larger sample, we realised that processing time increased considerably. Request from our internal clients also increased exponentially, which resulted in several new iterations of the software and also a constant degradation in performance. We were producing more information for sure, but we started noticing that the content of certain reports were maybe more trivial then useful.

3.2 AtCetera

We took a step back and asked ourselves what really needed to be produced by an Audit trail files analysis. Basically, we needed to cut out extra and useless output, and create a solid application which could be use by everyone without any assistance. We came up with AtCetera, which is a Visual Basic interface with standard Manipula scripts used to produce the output.

The principal goals of AtCetera were to simply and quickly produce useful descriptive information from audit trail data, for any questionnaire of any survey. We simplified our approach by getting rid of the survey specific aspect that was predominant in ATLAS. The core of AtCetera is three Manipula scripts that organize the raw audit trail data, prepare that data for aggregation, and then perform the aggregation. The scripts do not need to be recompiled; they work for all surveys, for all questionnaires. In terms of performance, AtCetera processes audit trail data at the speed of 50 interviewing minutes per second, which was significantly faster than its predecessor.

For the output we kept field and block timing information as it is globally requested. All clients want this information, especially from field tests, as the timing data shows how long their questionnaire really takes. This is quantitative data. There are some special events that are tracked by AtCetera. We keep track of when edits are triggered, where answers are changed, where the language is changed, and where remarks are written. This data is more qualitative than quantitative, as it looks into areas that could indicate problems with the design of the questionnaire.

In AtCetera, there are four ways of visiting (entering and exiting) a field.

- **Entry:** A field was EMPTY when entered, and it had a value or a status when it was exited. An empty field has a normal status and no value.

- **No Change:** A field was empty when entered, and remained empty when it was exited. This either means that the field is allowed to remain empty, or that the interviewer backed out of a field without entering any data.

- **Update:** A field has a value or status when entered, and had a different value or status when exited. Normally, a previously entered answer is being changed.

- **Review:** A field has a value or status when entered, and has the same value or status when exited. Normally, the interviewer is going through this question to get somewhere else.

There are several special events that can happen during an interview which are tracked by AtCetera. This list of special events is not exhaustive; it's just the events that are tracked by AtCetera.

- **Question Level Interviewer Remarks:** By pressing F4, interviewers can record remarks for the current question.

- **Changing Language of Interview:** When an interviewer changes the language of interviewing, it is recorded.

- **Edits:** When an interviewer encounters an edit (SIGNAL or CHECK), it is recorded.

The standard output of AtCetera is a single Excel file which has three sheets. One sheet has data at the unique field level, a second sheet has data at the (data model) block level, and the third sheet explains the column headers used in the other two sheets. Using Excel lets output recipients look at the data in many different ways, using standard software.

The following example of the field level output shows all of the fields in a block called AD.

Table 1. AtCetera field level report

CapTime	DropT	Hits	LngH	RpdE	LngE	RpdN	LngN	RpdU	LngU	RpdR	LngR	Edit	Rmrk	F2	Block-ID
3079	0	310	0	5	305	8	3	4	4	68	9	0	0	0	AD.AD_Q1
705	0	80	0	5	72	1	9	0	1	20	0	0	0	0	AD.AD_Q2
3545	0	308	0	13	295	7	2	0	0	58	3	0	0	0	AD.AD_Q3
3000	22	308	1	22	286	3	1	2	0	54	3	0	0	0	AD.AD_Q4
2627	0	308	0	36	272	4	2	0	1	50	3	0	0	0	AD.AD_Q5
2404	0	308	0	39	269	1	0	1	0	50	6	0	0	0	AD.AD_Q6
2658	0	308	0	38	270	3	2	7	5	57	11	0	0	0	AD.AD_Q7
670	0	68	0	4	68	6	9	1	0	6	0	0	0	0	AD.AD_Q8
3063	0	308	0	40	268	5	8	2	1	56	4	0	0	2	AD.AD_Q9
3340	0	310	0	2	0	134	265	0	0	4	0	0	0	0	AD.AD_QINT

The next example is of the block level sheet. The AD line shows what happened in all of its fields.

Table 2. AtCetera block level report

CapTime	DropT	Hits	LngH	RpdE	LngE	RpdN	LngN	RpdU	LngU	RpdR	LngR	Edit	Rmrk	F2	Block-ID
24091	22	2616	1	204	2100	172	301	17	12	418	39	0	0	2	AD
159292	305	21900	6	4969	14430	1805	1697	234	202	3627	246	43	7	4	AL
154455	6393	13265	5	441	12822	202	195	124	133	2002	199	2	1	17	ALS

4. Questionnaire evaluation and design

The qualitative aspect of the audit trail data quickly gained some interest in the question design, operation and methodology fields. Some studies were performed to learn on how questionnaires were used by the interviewers, while others were focusing on how to improve and/or correct the design of a questionnaire.

4.1 Adapting questionnaire based on Interviewer Interaction

You can learn a lot about your questionnaire based on how the interviewers use it. By analysing the audit trail files and looking at numerous actions, you can derive information that could help you improve your design.

4.1.1 Remarks

In the course of an interview, the interviewer has access to a Remark window, in which he can capture "parallel" information provided by the respondent that is not part of the questionnaire itself, but that can help survey analysts in understanding the content of the data. The capture of remarks by the interviewer can be a good source of information about the questionnaire design, the data quality, or characteristics of the respondent. The fields that often necessitate the addition of a remark can indicate that a question was not clear or that the choices of response categories provided to the respondent did not cover some potential responses. Also, having numerous remarks for a given respondent might provide information on the respondent. For example, a woman that has just recently delivered could feel the need to justify her responses to a set of questions about her level of physical energy by her "temporary situation".

4.1.2 Help

Fields where the Help key is frequently used could indicate that a question is not clear. Most studies are showing that very few problematic fields can be identified using this concept. In fact, we observed that the Help key is very rarely used. This may indicate that the interviewers have a good knowledge of the survey content and procedures, and hence rarely need assistance, but it could also mean that the interviewers do not make use of the information available in those screens for other reasons. Perhaps they are not aware that Help screens are available, or perhaps they don't found them to be useful.

4.1.3 Language change

Changing the language in which the interview is conducted during the course of an interview between French and English can indicate that the respondent or the interviewer was seeking a clarification on the question wording, perhaps not fully understanding what was asked. Fields that often necessitated a change of language can indicate a problem with question wording.

4.1.4 Flow

Analyzing the flow of the questionnaire can shed some light on potential problems with the question wording or questionnaire design. A "perfect" interview that goes from beginning to end without any backward navigation in the questionnaire should consist only of visits of the "Enter" type. That is, each field would be empty when visited for the first time, filled on the first visit, and never visited again. By exploring fields that do not seem to follow such a pattern, or blocks with many such fields, one could highlight problematic areas of the questionnaire. However, it can be difficult to define what should be considered problematic, and to target the source of the problem. Fields or blocks that seem to have "outlying patterns" could be investigated. These outputs could also be used to assess specific issues that might have been raised up by interviewer during collection.

4.1.5 Multiple visits

A high proportion of visits that are not the first visit to the field indicates interviewers often had to revisit the field, either for updating it, or to move to previous or following fields.

4.1.6 Proportion of time spent on Enters

Fields with a low proportion of interview time spent on visits of the type "Enter" can give an idea of the magnitude of "wasted interview time". Note that the proportion of visits that are not of the type "Enter" should be highly correlated with the proportion of multiple visits, since in theory, each respondent to a field should not have more than one visit of the type "Enter".

4.1.7 Updates

Fields that often necessitated to be updated could indicate a problem with the question wording, the definition of a concept, or an edit rule that would be too strict.

4.1.8 Rare visits

Fields visited by very few respondents could indicate a problem with the skip patterns defined in the question, if the question was not designed to be addressed to a rare population. Note that fields that are never visited cannot be identified solely with the Audit Trail data. To identify these, a list of all existing survey fields would have to be available.

4.1.9 Long visits

Visits that last more than 180 seconds should be rare. A field with a high proportion of Long Visits should be investigated.

5. Monitoring

As we spent time looking at how interviewer were using the Blaise questionnaire in order to improve our design, we realised that some level of monitoring could be done on the interviewing task.

5.1 POInt System

Interviewers are trained to maintain a moderate pace while interviewing, to read all questions at the same pace, and to wait for the respondent to answer. The POInt (Pace Of Interviewing) system uses audit trail data to determine if interviews are, in fact, being conducted in that manner.

Blaise audit trail data contains a wealth of information on the interviewing process. For each field that is traversed, the time and state of the field is recorded, as it is entered and as it is exited. That data can be used to determine what happened to the field, and how long it was active. By evaluating the field level data for the subject matter portion of a data collection application, and comparing it against criteria for that application, it is possible to identify calls that are irregular (where the fields were traversed too rapidly).

CATI (Computer Assisted Telephone Interviewing) interviewers at Statistics Canada have been subject to unobtrusive monitoring for many years. CATI monitoring is performed on a fraction of calls, and only a fraction of a call is monitored. CATI monitoring evaluates an interviewer subjectively, in ways that cannot be done by computers. The monitor determines if the interviewer is being polite, is speaking clearly, is asking the question as worded, and is entering the answer as given.

POInt evaluates all calls, but its evaluations are objective, rather than subjective. POInt is not a replacement for CATI monitoring, as it cannot do the subjective evaluations. It is a complement to CATI monitoring.

5.1.1 POInt Genesis and development

In the summer of 2006, we looked into a small number of interviews which seemed to have been done at a rapid pace. A manual review of the audit trails of several interviews showed that some of them had been completed quickly. A more thorough investigation was clearly needed, as there might have been other irregular interviews. It was not feasible to review the 8,000 audit trail files for this survey by hand, so a computer program was developed to automate the process. This was the infancy of the POInt system.

This original program determined how many subject matter fields had been changed, how much time had been spent in those fields, and how many of the fields were non-response. Pace and item non-response rate were determined for each call. It was noticed that this base code could be used on any of our social surveys, with only minor changes (thanks to our Blaise coding standards). We soon had generic code that could be used to evaluate calls from any social survey application.

We were very fortunate in some of our existing practices. Audit trail data was already being returned from regional offices to head office on a nightly basis. There was an expandable report interface that was used by regional office staff (Data Integration and Production Planning system - DIPP). We had the ability to process data daily, and an existing portal through which POInt reports could be delivered. Having determined that full scale production was viable, development for a generic POInt evaluation and reporting system began in late spring 2007. POInt started being used in production in December 2007.

At first, a single "speed limit" for all collection was thought to be appropriate. The study of historic audit trail data quickly disproved that notion: surveys are collected at surprisingly different paces. Clearly, each survey needed its own speed limit. Our early thoughts on item non-response also needed to be updated. Initially, the boundaries for item non-response were too low. Non-response is almost always a reflection of the respondent, rather than the interviewer. Only extremely high item non-response should cause a call to be flagged.

At the start of development, evaluation criteria were hard-coded. Individual survey level scripts were considered, but it was quickly noticed that the only lines of code that would change would be those that defined the criteria. The most obvious mechanism to record the evaluation criteria for a survey was initialization files; the use of survey-specific initialization files removed the need for survey-specific process control scripts.

Each survey (application) in POINT has an initialization file, which tells the standard programs how to identify content fields, which field is used to determine whether the case is a full-complete, and what the evaluation criteria are. An example follows.

Listing 1. POINT Example Source Code

```
[PointMode]
Parms=Known
CallsToSet=600

[Dates]
StartDate=20160101
EndDate=20201231

[ContentDef]
ContentField=Content.
```

```
LastField=Content.SO_N01

[Criteria]
AvePace=6.7
MaxPace=11.7
MaxINR=25.0
MinFCH=20

[ProcessControl]
MakeUFICReport=Yes
CopyReportsToArchive=Yes
CopyMATToArchive=Yes
```

In the *PointMode* section, we can see that the parameters have been established (*Parms=Known*).
The *Dates* section can be used to restrict processing to files from certain dates. This feature has only been used during testing, and is not a universal restrictor: some programs do not have any *StartDate* or *EndDate* logic.

In the *ContentDef* section, the content superblock is Content (*ContentField=Content*.), and the field used for auto-coding full-complete outcomes is *Content.SO_N01*.

In the *Criteria* section, we can see that the early average pace of content collection was 6.7 field changes per minute (fcpm), which is another way of saying "questions per minute". The maximum pace has been established at 11.7 fcpm, and the maximum item non-response rate (INR) is 25.0% (this is a default value). We can also see that at least 20 content fields have to be changed for a call to be evaluated (MinFCH=20).

In the *ProcessControl* section, we can control whether UFIC (*unusual fields for irregular calls*) reports are generated, and whether reports and audit trail files are copied to the archive. During POINT testing, setting these controls to "No" will allow for faster and simpler repetitions.

Before being able to generate its reports, POInt needs to create a processable data file from the Blaise audit trail file. The first script will produce a field by field (FxF) file which contains one record for each visit to each field. The following is an example of the FxF data file.

20080123,15400702171,1,intrvwr,YITS.U.U_Q01[1],178,2,,N,1,1,2,1,N,7,1300,21:40,
20080123,15400702171,1,intrvwr,YITS.U.U_Q03[1],179,2,,N,1,1,2,1,N,5,1305,21:45,
20080123,15400702171,1,intrvwr,YITS.U.U_Q04,181,2,,N,1,1,2,4,N,8,1320,22:00,
20080123,15400702171,1,intrvwr,YITS.U.U_Q04a,182,2,,N,1,1,2,2,N,2,1322,22:02,
20080123,15400702171,1,intrvwr,YITS.U.U_Q07,183,2,,N,1,1,2,2,N,7,1329,22:09,
20080123,15400702171,1,intrvwr,YITS.U.U_Q09,184,2,,N,1,1,2,1,N,8,1337,22:17,
20080123,15400702171,1,intrvwr,YITS.U.U_Q37B,185,2,,N,1,1,2,2,N,9,1346,22:26,
20080123,15400702171,1,intrvwr,YITS.U.U_Q59,186,2,,N,1,1,2,2,N,7,1353,22:33,
20080123,15400702171,1,intrvwr,YITS.U.U_Q60[1],187,2,,N,1,1,2,6-4-2,N,21,1374,22:54,
20080123,15400702171,1,intrvwr,YITS.U.U_R01,177,2,,N,1,2,2,,N,2,1293,21:33,
20080123,15400702171,1,intrvwr,YITS.U.U_R04,180,2,,N,1,2,2,,N,7,1312,21:52,

Next step, using the FxF as input, is to create a call by call (CxC) file. This is an example of the content of a CxC data file. This represents one call made by one interviewer. It is for the same interview that was used to provide the previous example.

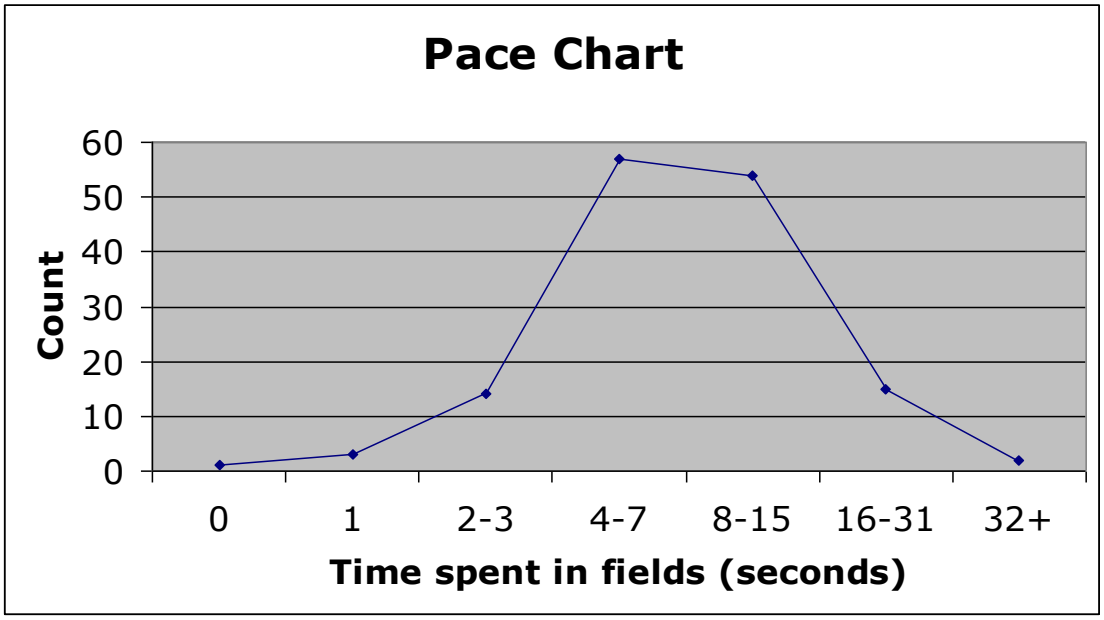
20080123,15400702171,1,intrvwr,159,6,146,0,0,0.0,1273,1210.4,6.9,7.2,0,0,0.0,1,3,14,57,54,15,2,1

To the experienced eye, this is a very readable set of data. It starts with the *Date*, *CaseID*, *CallID* and *InterviewerID*. 159 fields were visited. 6 field-visits were considered unusual, and 146 fields were changed. There was no item non-response: 0 DK, 0 RF, 0.0% item-non-response rate (0,0,0.0).

A total of 1,273 seconds were spent in content fields that changed. The amended time for these fields is 1,210.4 seconds. The actual pace of collection was 6.9 field changes per minute (fcpm), while the amended pace was 7.2 fcpm. A call is evaluated on amended pace.

There were no extra-long-visit fields. A field-visit must last at least 180 seconds to qualify as an "extra-long-visit field". The next few numbers (1,3,14,57,54,15,2) are frequencies of how many fields were active for certain time periods. In the following chart, you can see that most of the questions were active for between 4 and 15 seconds.

Figure 1. Pace Chart



The last data item shows that the last question was answered. This was a full-complete interview.

The last step is to run the scripts that create the POInt reports.

5.1.2 POInt Reports

POInt produces report data in the form of comma delimited text files. These files are sent to the DIPP system, where they are converted into a series of inter-related Microsoft® Excel spreadsheets. This allows report users to manipulate report data using familiar software. They are able to drill into more detailed reports through hyperlinks in the spreadsheets.

DIPP reports are role sensitive. Regional office collection managers can see details about individual interviewers, while head office staffs get no interviewer-level detail.

Summary of Calls by Survey is the highest level report. It shows the number of evaluated calls by survey and collection period, with counts of regular and irregular calls.

Survey Calls by Interviewer summarizes the performance of all interviewers who worked on a specific survey. It shows counts of regular and irregular calls, amended pace, and field-level non-response rate. A somewhat related report is *Interviewer Calls*, which has the same report content for all the surveys of one interviewer.

The *Irregular Calls* report has information for each irregular call in a given survey and collection period. Interviewer-id, pace and field-level non-response rate are the major items of interest.

Finally, the *Unusual Fields of Irregular Calls* report shows, as its title implies, all of the "unusual fields" that were encountered during an irregular call. This report allows managers to better understand why a call was flagged as irregular, and provides a solid set of information that can be used during a discussion with the interviewer. An "unusual field" is one that: was changed in less than two seconds; or was changed to Don't Know (DK)/Refusal (RF); or had an interviewer comment; or was active for at least 60 seconds.

Collection managers should use POInt reports to coach interviewers towards better performance, and to provide positive feedback for good performance. A fact that leaps out from looking at these reports is that most interviewers do an excellent job, every day.

5.1.3 POInt Status

POInt has been in use since December 2007. Almost all CATI social surveys now use POInt. To date, about 1% of calls have been evaluated as irregular. There have been instances where offices had more than 1,000 calls for a survey, with all calls evaluated as regular. In 2009, POInt was expanded to evaluate very short interviews. Rather than evaluating individual units of work (calls or questionnaires), a body of work (an interviewer's shift) is evaluated.

6. Centralisation and standardization

With audit trail data being used extensively throughout Statistics Canada, a need to centralise and standardise the audit trail process was identified.

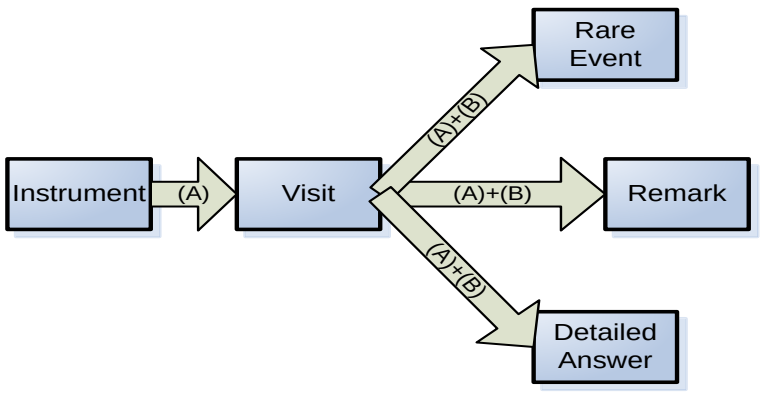
6.1 Standardized Audit Trail data

In practice, each visit to a survey question generates about 180 characters on the audit trail file. This format does not allow for direct and friendly use and analysis of the data. In order to fully benefit from this enormous and detailed amount of data, the raw audit trail files were transformed into a more structured and coherent format using a standardised process that can be applied to all surveys.

The standard storage structure for processed audit trail data for all CATI and CAPI surveys was influenced by concerns for ease of use, concerns about storage burdens and data base normalization theories. The standard storage structure is an inter-related set of five (.txt flat file) outputs.

In the early discussions that led to this proposal, it was agreed that the structure should hold only that data which would be required by more than one group. Unique needs would be handled by giving the group which had the unique requirements access to the raw audit trail files.

Figure 2. Standardized output



The Output Tables can be linked together using (A) "Uniqueinst" and (B) "VisitSeq" variable.

The ***Instrument Table*** contains summary information about the use of the collection instrument. Each combination of *Enter Form* and *Leave Form* corresponds to one record on the Instrument Table. This table records what instrument was used, who used it, when it was used, which sample record was involved and the file name from which the data are coming from. The table is sorted by SampleID and in chronological order. The table can be linked to the Visit Table using the variable *Uniqueinst*.

The ***Visit Table*** contains field-visit level data. There is one record for each visit to each field. A visit to a field represents the sequence of actions that are performed between the moment a field is entered (keyword *Enter Field*) and the moment the following field is entered (through another *Enter Field*). The Visit Table contains a visit sequence identifier, the field name, the value when entering the field and the value when leaving the field, the type of visit and the duration of the visit. Very long field visits in terms of duration are capped to 18000 seconds (5 hours), to be consistent with BTH outlier treatment.

The Visit Table can be linked to the Instrument Table using the variable *UniqueInst*, and to the Rare Event Table, the Remark Table and the Detailed Answer Table using the combination of the variables *UniqueInst* (A) and *VisitSeq* (B).

Each visit can be categorized into one of 5 "types of visit" (variable *Visit_Type*), based on the value when entering the field (either blank or not) and the value when leaving the field (either value modified or unchanged during the visit).

The main aim of this variable is to help to quickly have an idea of how the interviewer is navigating in the application, that is:

Table 3. Type of visit

Type of visit	Value when entering field	Value when leaving field	Interpretation
1: Entry	Empty	Value	Initial capture of a value for the field. An interview that would go from start to finish without any back and forth in the questionnaire should consist of visits that are all of this type (except if the survey has pre-filled fields).
2: Update	Value	Different value	Modification to previously reported value
3: StillBlank	Empty	Empty	Leaving a field without having entered a value. Usually because the interviewer goes back into earlier fields.
4: Review	Value	Same Value	Moving upward or downward in the application, through fields that were already responded to
5: Truncated record	In some rare instances, problems with the survey application (e.g. "frozen application") can result in incomplete Audit Trail data, which makes it impossible to determine the value at entry and at exit with certainty.		

To minimize the size of the Visit Table, the two variables containing the answers to the survey questions (*EntryValue* and *ExitValue*) are only 8-digits long. When a value is either a numeric value of more than 8 bytes or a character value, only the length of the value is stored in the Visit Table (if needed, the initial complete values can be retrieved from the Detailed Answer Table).

The values in the Visit Table are enclosed in one of three types of brackets, depending on the type of data it contains. That is:

- **Square brackets [ ]**: Value reported by the respondent. This can be either a numerical value with length <= 8, or *Don't Know* and *Refusal* (stored as [??] and [!!]), or blank values (stored as [])

- **Curly brackets { }**: Length of numerical values when length is greater than 8

- **Parenthesis ( ):** Length of character value

Below is an example of how different values found in the raw data would be stored in the Visit Table. It also shows what the value of *Visit_Type* would be for the different values at entry and at exit. Note that Visit_Type is derived from the raw values, not from the values found in the Visit Table (for example, a change from 1-4-9-13-17 to 1-5-9-14-18 results in Visit_Type = 2: Update, even though both values appear as {11} in the Visit Table.

Table 4. Types of brackets used to store different types of values in the Visit Table

Values in the raw ADT files		Values in the Visit Table		
Entry	**Exit**	**EntryValue**	**ExitValue**	**Visit_type**
	1	[]	[1]	1: Entry
Don't Know	Refusal	[??]	[!!]	2: Update
		[]	[]	3: StillBlank
Ottawa Region	Ottawa Region	(13)	(13)	4: Review
1-2-3	1-2-3-6-9-13	[1-2-3]	{12}	2: Update
	K1A 0T6		(7)	2: Update
2-8-9	2-6-8	[2-8-9]	[2-6-8]	2: Update
1-4-9-13-17	1-5-9-14-18	{11}	{11}	2: Update
1-2-3-6-9-13	Don't Know	{12}	[??]	2: Update
9 St-Denis	209 Saint-Denis	(10)	(16)	2: Update
1-2	One or Two	[1-2]	(10)	2: Update

The ***Rare Event Table*** contains information about rare events that happen during collection. It is understood that storing these rare data in a separate table should significantly reduce the size of this data base. Rare events are things like use of the Help key, changing the language of the CAI application, entering field-specific remarks and interviewer's action after an edit failure.

The interviewers' reactions to edit failures are also recorded on the file. The interviewer can react in three different ways. The interviewer can press the Escape key or 'Close' button, which will make the application go back to the field that was just left, allowing the interviewer and respondent to make the necessary modifications to previously reported fields in order to correct the inconsistency (*Action: Error Escape*). The interviewer can also press the Suppress button that appears on the pop-up window, which will make the application move on to the next field (*Action: Error Suppress*). The Suppress button is only available for soft edits, since hard edits detect impossible values and are not allowed to be ignored. Finally, the interviewer can press the 'Goto' button (*Action : Error Jump*). For *Error Jump*, the application will return to the field highlighted in the edit pop up window.

The ***Remark Text Table*** contains the remark text entered by the interviewer about specific field. There should be one record for each record of the Rare Event Table that has *RmrkChng* = 1.

The ***Detailed Answer table*** contains the complete values reported by the respondent when only the length of the value was kept in the Visit Table. This table aims at preserving as much as possible the information found in the raw Audit Trail files.

The Detailed Answer Table contains the complete reported values for such cases, with one record for each visit of the Visit Table that has the length of the value instead of the value itself (at entry and/or at exit). All values in the Detailed Answer Table are stored in square brackets. This is consistent with the convention used for the Visit Table, since all values in the Detailed Answer Table are values reported by the respondent.

Below is an example of how different values found in the raw data would be stored in the Detailed Answer Table. The records of the Visit File that have both *EntryValue* and *ExitValue* enclosed in square brackets (i.e., both are values reported by the respondent) do not have a corresponding record on the Detailed Answer Table.

Table 5. Detailed Answer Table

Values in the raw ADT files		Values in the Visit Table		Values in the Detailed Answer Table	
Entry	**Exit**	**EntryValue**	**ExitValue**	**Detailed EntryValue**	**Detailed ExitValue**
	1	[]	[1]		
Don't Know	Refusal	[??]	[!!]		
		[]	[]		
Ottawa Region	Ottawa Region	(13)	(13)	[Ottawa Region]	[Ottawa Region]
1-2-3	1-2-3-6-9-13	[1-2-3]	{12}	[1-2-3]	[1-2-3-6-9-13]
	K1A 0T6		(7)		[K1A 0T6]
2-8-9	2-6-8	[2-8-9]	[2-6-8]		
1-4-9-13-17	1-5-9-14-18	{11}	{11}	[1-4-9-13-17]	[1-5-9-14-18]
1-2-3-6-9-13	Don't Know	{12}	[??]	[1-2-3-6-9-13]	[??]
9 St-Denis	209 Saint-Denis	(10)	(16)	[9 St-Denis]	[209 Saint-Denis]
1-2	One or Two	[1-2]	(10)	[1-2]	[One or Two]

7. Support

As a developer, the primary use of the audit trail files is predominantly for support purposes. It helps in the testing and the debugging of the Blaise code. Audit trail data is usually the first source of information we will look at when issues are reported during testing and collection. It will generally provide more information than what will be sent to us in the issue description. The adt files allow us to troubleshoot the issues more efficiently and effectively. And this is just the tip of the iceberg, this last section will briefly highlight other technical uses of the audit trail data throughout the last decade.

7.1 Automated Testing

Testing can be a repetitive and tedious task. Finding ways to improve and simplifying it will always be at the centre of development practices discussions. Automated testing is an area where we feel we can make some gain in our quest to streamline the testing process. Different initiatives were put in place, and one of them used audit trail files.

Some recurrent surveys utilise the same pre-defined testing scenarios rounds after rounds to perform some level of regression testing. This set-up allowed us to develop a custom made “replayer” which would receive audit trail files from a previous round as input and then “replay” the scenario to perform regression testing. This was saving a lot of keying time for the tester and we were making sure that the same testing situations were applied between rounds.

However, this strategy had its limitations; for instance, a change in flows, randomization in the questionnaire, or the addition or the deletion of a question resulted in unmanageable situations for our replayer program. More development time would have been required to make this process more robust, but it has been put on hold for resourcing issues.

7.2 Data recovery

As documented, the audit trail file also offers a means for data recovery in case of technical glitches, power shutdowns/failures or improper Blaise coding. We did make use of this feature occasionally. It has been a good emergency plan when for a short period of time we experienced some databases corruption issues throughout several applications. During that time, several cases were dropped during our overnight process from the databases after the execution of the Hospital recovery program. While we were troubleshooting the cause of this issue, we used audit trail files to repopulate the dropped cases in the databases applying an adapted version of the strategy supplied with the Blaise software (AuditSummary.man and Repopulate.man). The plan in place allowed us to automate the data recovery process and concentrate our effort on identifying the source of the problem (which was an issue with the Hospital program).

Audit trail files saved several interviewing hours for another of our CAPI project. The Longitudinal and International Study of Adults (LISA) consists of a very complex datamodel. For the wave conducted in 2014, we realised once in production that in certain circumstances part of the data wasn't kept when the interview required multiple visits. A KEEP method was missing in our source code and a patch was required and sent out in the field. Unfortunately, it took a while before realising this problem, and the damage was already done. Over 1,000 cases were affected by this programming error and our internal client initial thought that this data was gone and irrecoverable. Using the adt files generated by Blaise, we were able to recuperate the missing values by developing a process to recreate the output required by our internal client.

7.3 Performance metrics

It's not always easy to assess the performance of a system. To be able to determine the impact of a change on the infrastructure and/or inside the Blaise application we often need to trust the eye test. In the last years, we went through several infrastructure changes (Vista, Windows 7, VDI, Blaise services, etc.) and some resulted in perceived performance degradation based on interviewers' feedback. We always felt that we needed a way to baseline the performance and have metrics to proper evaluate the effect of a change on the system.

The audit trail files can, to a certain extent, provide valuable metrics to assess performance. For example, we are using the question to question delays recorded on the audit trail files (calculate to the thousandth of a second as we are using the PreciseTiming option in the AIF file) to calculate system responsiveness. The assumption is that the faster the system responds, the better the performance. Using years of audit trail data from our Labour Force Survey, we've been able to come up with good indicators and determine more accurately the performance impact of a change in the collection system and/or on the infrastructure.

8. Summary

Even before implementing the audit trail files functionality into our projects we knew how useful it would be to technically support our applications. Being able to derive precise timing information using the ATLAS and AtCetera system was already a big plus of having this feature enabled, but it turns out to be only the beginning of an interesting journey. We improved our data quality by enhancing our questionnaires design practices and by monitoring objectively our interviewers using the POInt system. We now have a centralized and standardized approach set up for audit trail file analysis, which should translate in new ways to utilise this vast and rich source of information.

9. References

Ali, Jennifer, "Data Quality Monitoring Using the Blaise Audit Trail", 2003, SSC Annual Meeting, June 2003, Proceedings of the Survey Methods Section.

Egan, Michael D.A., "POINT – A Method for Evaluating the Quality of Interviewing", 2010, Statistics Canada internal documentation.

Hamel, Sylvain, "Audit Trail Files (ADT) – Structured Output", 2014, Statistics Canada internal documentation.

Lapierre, Bruno and Meyer, Scott, "Using the Audit Trail data to evaluate the quality of collection of the Canadian National Longitudinal Survey of Children and Youth", 2006, American Statistical Association 2006 Proceedings of the Section on Survey Research Methods.