Transforming Survey Paradata

Lisa Wood, Andrew Piskorowski, and Jennie Williams, University of Michigan Survey Research Center, United States

1. Abstract

Paradata that are captured during the survey process are a valuable source of information in helping us understand and improve the data collection process.

One of the advantages of using Blaise for survey data collection is the rich capture of paradata that is native to the application. Paradata which are linked directly to the administration of a survey instrument are collected automatically through the Blaise software (i.e., audit trail). The ADT file from Blaise 4 has been very valuable in understanding interviewer behavior. With the advent of Blaise IS and Blaise 5, we now are able to increase the richness of our paradata collection for understanding respondent behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer and web-SAQ combined).

The main focus of this paper will be to share a process to make these sources of Blaise 5 paradata more usable for analysis and reporting. The native Blaise 5 paradata, the user agent string, and the University of Michigan's client-side paradata (CSP) are unstructured data and very cumbersome to "untangle". To make these data more useful for analysis, various data management techniques are applied with the following goals.

- Parse this unstructured data (e.g., SAS, Python, SQL)
- Calculate new measures (e.g., time on/time between page, last question seen/answered)
- Aggregate data at various levels (e.g., page-level, session-level, respondent-level)

In addition, reporting tools such as OLAP cubes and SSRS (SQL Server Reporting Services) are used to distribute the data to various user groups (e.g., PIs, production managers, statisticians, etc.). The resulting output is also available in a SQL database and can be accessed using other reporting and analysis tools. The transformation techniques and standard paradata reports can be implemented by any user of Blaise 5 paradata to enhance the use of this data.

2. Introduction

Paradata that are captured during the survey process are a valuable source of information in helping us understand, monitor and improve the data collection process (Couper, 1998; Couper and Peterson, 2016; Cheung, Piskorowski, Wood and Peng, 2016). With the advent of BlaiseIS and Blaise 5, we now are able to increase the richness of our paradata collection for understanding respondent behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer and web-SAQ combined).

There are many possible sources of paradata surrounding the survey data collection process. In this paper we focus on the two key types of survey paradata:

1. *Native Blaise 5 paradata (Server-side)*
    a. This is the paradata which are linked directly to the administration of a survey instrument, and that are collected automatically through the Blaise software (i.e., audit trail).
    b. One of the advantages of using Blaise for survey data collection is the rich capture of paradata that is native to the application.
    c. We have seen that the ADT files from Blaise 4 have been very valuable in understanding interviewer behavior (Devonshire, Liu and Cheung, 2013).
2. *Supplemental Client-Side Paradata (CSP)*

a. The team at the University of Michigan has implemented a process to capture paradata from the client-side using JavaScript (Peng and Ostergren, 2016).
b. This client-side paradata (CSP) can capture movement within a page on the respondent's or interviewer's device, like scrolling and mouse-clicks.
c. As the use of mobile devices has increased, the CSP can now capture events specific to the mobile device like tapping, orientation change, pinch in/out, etc.

The usefulness of the paradata is dependent on the ease with which it can be accessed, manipulated, and analyzed by end users. This stream of data can quickly grow exponentially, and in its raw data state, it is nearly impossible to use for analysis.

The main focus of this paper is to share how we are working to make these sources of Blaise 5 paradata more usable for analysis and reporting. These sources of survey paradata are unstructured data and very cumbersome to untangle. To make these data more useful for analysis, various data management techniques are applied with the following goals:

- Parse this unstructured data (e.g., SAS, Python, SQL)
- Calculate new attributes and measures (e.g., time on/time between page, last question seen)
- Aggregate data at various levels (e.g., page-level, session-level, respondent-level)

In addition, reporting tools such as OLAP (online analytical processing) cubes and SSRS (SQL Server Reporting Services) can be used to distribute the data to various stakeholders (e.g., Principal Investigators, production managers, statisticians, etc.). The resulting output is available in a SQL database and can be accessed using other reporting and analysis tools. The transformation techniques can be implemented by any user of Blaise 5 paradata to enhance the use of this data.

3. Where the Paradata Starts (Native Blaise 5 Audit)

From the Blaise 5 Help documentation, *"Audit trail or audit log is a chronological sequence of audit records, each of which contains evidence directly pertaining to and resulting from the execution of a Data Entry session.* ***Audit Trail is a feature of data entry which comes disabled by default in Blaise****. With this feature enabled you can show detailed information on visited pages, keys pressed, mouse clicks etc."* (2015 Statistics Netherlands)

When the audit trail is enabled, there are various levels of information that can be captured (see the Blaise 5 help documentation for more details). The team at the University of Michigan uses the most detailed level of logging called "Keyboard".

We begin by looking at the native Blaise 5 paradata in its initial (raw) form. This data lives in the AuditTrailData.db (note: this data may be stored in a SQLlite or a SQL db, but the structure/contents are the same).

The Blaise 5 native paradata has one row for each "event" and the attributes include:

1. KeyValue - the unique key for the specific case (e.g., SampleId, LoginId)
2. InstrumentId - this identifies the specific Blaise 5 instrument
3. SessionId - this identifies all events within a Blaise session
4. TimeStamp - the datetime that the server records the event
5. Content - all the data about the event; the structure/contents of the paradata in this attribute depends on the type of event.

These are the types of events that we have uncovered (note: this list may not be exhaustive):

1. StartSession - this should be the first event at the start of a survey session; it includes information about device used to access the instrument including browser and the user agent string

2. Action - e.g., NextPage, PreviousPage
3. UpdatePage - capturing the movement between pages
4. EnterField - shows when the page is actually rendered. However, multiple enter fields events can happen on a page, and shows when a field is in focus (at beginning of a page load and navigation within a page)
5. Category and Keyboard - shows the response selection on radial buttons (Category) and the recording of keystrokes (Keyboard) on a page
6. LeaveField - capturing the movement from the page, and like the Enter field events, multiple leave field events can happen on a page
7. EndQuestionnaire - shows the completion of the Blaise instrument (note: there is no end session event if a survey is started and abandoned before completion)
8. InterruptSession - this is when the interviewer clicks the "quit" (interviewer administered)

Table 1. An Example of the Data in Its Raw Form

End of Session - CATI (Suspended)				
KeyValue	**InstrumentId**	**SessionId**	**TimeStamp**	**Content**
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:24:08	<LeaveFieldEvent FieldName="Section_a.Sec_S.S_SC_AcadRank_b" Value="5" AnswerStatus="Response" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:24:08	<ActionEvent Action="NextField()" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:24:08	<UpdatePageEvent LayoutSetName="Cati" PageIndex="109" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:24:08	<EnterFieldEvent FieldName="Section_a.Sec_S.S_SC_CertsDeg" AnswerStatus="Empty" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:27:08	<KeyboardEvent KeyStrokes="5" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:27:08	<LeaveFieldEvent FieldName="Section_a.Sec_S.S_SC_CertsDeg" Value="5" AnswerStatus="Response" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:27:08	<ActionEvent Action="NextField()" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:28:08	<UpdatePageEvent LayoutSetName="Cati" PageIndex="149" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:28:08	<EnterFieldEvent FieldName="Section_b.HE_GenHlth" AnswerStatus="Empty" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:31:08	<LeaveFieldEvent FieldName="Section_b.HE_GenHlth" AnswerStatus="Empty" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:31:08	<ActionEvent Action="Save()" />
L19178	{9b255695-0a9c-40	{2d6e47c6-d489	8/29/2016 11:00:31:08	<InterruptSessionEvent />
Start of Session - Mobile				
KeyValue	**InstrumentId**	**SessionId**	**TimeStamp**	**Content**
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:53:37:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="2" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:53:37:08	<EnterFieldEvent FieldName="Welcome" AnswerStatus="Empty" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:53:37:08	<StartSessionEvent Width="980" Height="1461" Device="Browser" Browser="Safari" Language="Mobile" KeyValue="L19330" Platform="HTML" OS="Mozilla/5.0 (iPhone; CPU iPhone OS 9_3_4 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13G35 Safari/601.1" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:53:46:08	<ActionEvent Action="NextPage()" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:53:46:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="5" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:53:46:08	<EnterFieldEvent FieldName="INTRO_ConsentScreen_01" AnswerStatus="Empty" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:54:56:08	<ActionEvent Action="NextPage()" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:54:56:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="9" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:54:56:08	<EnterFieldEvent FieldName="xxxxdummy4" AnswerStatus="Empty" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:55:06:08	<LeaveFieldEvent FieldName="xxxxdummy4" AnswerStatus="Empty"/>
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:55:07:08	<ActionEvent Action="NextPage()" />
L19330	{9b255695-0a9c-40	{bba30c98-470f	8/24/2016 13:55:07:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="13" />

4. Processing the Native Blaise 5 Paradata

The data management team at the University of Michigan has developed SAS routines to parse the native Blaise 5 audit data. These routines interrogate the attribute called "Content". They have been developed to be transferable across any Blaise 5 project with just the definition of some key parameters. A sample of this code is provided in Appendix A.

When the native Blaise 5 paradata is processed for analysis, the resulting output is a data set that has the same number of records as the input file. However, new attributes or measures have been added. A data dictionary for the output data set is provided in Appendix B.

Here is an overview of the types of attributes/measures that are added to the Blaise 5 native paradata.

- **Event Type Flags**: Flags were added that indicate the event type (0/1 values). These flags are called SessionStart, SessionEnd, IWComplete, EnterField, PageUpdate, LeaveField.
- **Sequence/Counters:** The order in which events occur are very important. There are sequence or counters to indicate (a) the number of unique sessions for the case (SessionNum), (b) the sequence of events within the session (SessionSeq), (c) the sequence of events for a case across all sessions (EventSeq), and (d) the sequence of fields for a case (FieldSeq)
- **User Agent String:** Various elements from the user agent string (found in the StartSession event) were parsed out. These include Width, Height, Browser, and Platform. Additional fields can be parsed or inferred (e.g., type and name of device – e.g., desktop, tablet, phone).
- **Field Information:** Each row or event in the audit data can be associated with a field. This is provided in the new attributes called FieldName and Block. As the associated field/block are not always provided in the Content (native Blaise 5) for all events, some decisions are made on how to attribute an event/row to a Blaise field. In addition, if the response to a Blaise field is included in the Content, it is stored in the new attribute called Answer.
- **Timings:** Now we can start to determine timing between each event (EventTimeSec), time in each field (FieldTimeSec) and the time it took for a page to update (PageLoadSec).

Table 2a: An Example of the Processed Data – Start Sessions

KeyValue	Content	Width	Height	Browser	Platform	AgentString	Language
L19178	<StartSessionEvent Width="1936" Height="1056" Device="WindowsDesktop" Language="_CATI" KeyValue="L19178" Platform="Windows" />	1936	1056		Windows		_CATI
L23925	<StartSessionEvent Width="1920" Height="997" Device="Browser" Browser="Chrome" Language="WEB" KeyValue="L23925" Platform="HTML" OS="Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36" />	1920	997	Chrome	HTML	Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36	WEB
L23925	<StartSessionEvent Width="980" Height="1546" Device="Browser" Browser="Chrome" Language="Mobile" KeyValue="L23925" Platform="HTML" OS="Mozilla/5.0 (Linux; Android 6.0.1; SAMSUNG-SM-G935A Build/MMB29M; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/52.0.2743.98 Mobile Safari/537.36" />	980	1546	Chrome	HTML	Mozilla/5.0 (Linux; Android 6.0.1; SAMSUNG-SM-G935A Build/MMB29M; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/52.0.2743.98 Mobile Safari/537.36	Mobile

Table 2b: An Example of the Processed Data

End of Session - CATI (Suspended)										
KeyValue	**TimeStamp**	**Session Start**	**Session End**	**SessionType**	**FieldName**	**SessionSeq**	**FieldSeq**	**FieldTime Sec**	**EventTime Sec**	**PageLoad Sec**
L19178	8/29/2016 11:00:24:08	0	0	<LeaveFieldEvent	Section_a.Sec_S.S_S(	235			0.00	
L19178	8/29/2016 11:00:24:08	0	0	<ActionEvent	Section_a.Sec_S.S_S(	236			0.28	
L19178	8/29/2016 11:00:24:08	0	0	<UpdatePageEvent	Section_a.Sec_S.S_S(	237			0.00	0.28
L19178	8/29/2016 11:00:24:08	0	0	<EnterFieldEvent	Section_a.Sec_S.S_S(	238	62	3.51	2.99	
L19178	8/29/2016 11:00:27:08	0	0	<KeyboardEvent	Section_a.Sec_S.S_S(	239			0.20	
L19178	8/29/2016 11:00:27:08	0	0	<LeaveFieldEvent	Section_a.Sec_S.S_S(	240			0.00	
L19178	8/29/2016 11:00:27:08	0	0	<ActionEvent	Section_a.Sec_S.S_S(	241			0.32	
L19178	8/29/2016 11:00:28:08	0	0	<UpdatePageEvent	Section_b.HE_GenHlth	242			0.00	0.32
L19178	8/29/2016 11:00:28:08	0	0	<EnterFieldEvent	Section_b.HE_GenHlth	243	63	3.65	3.57	
L19178	8/29/2016 11:00:31:08	0	0	<LeaveFieldEvent	Section_b.HE_GenHlth	244			0.00	
L19178	8/29/2016 11:00:31:08	0	0	<ActionEvent	Section_b.HE_GenHlth	245			0.08	
L19178	8/29/2016 11:00:31:08	0	1	<InterruptSessionEvent	InterruptSession	246	64			
Start of Session - Mobile										
KeyValue	**TimeStamp**									
L19330	8/24/2016 13:53:37:08	1	0	<StartSessionEvent	StartSession	1	1	8.90	8.84	
L19330	8/24/2016 13:53:46:08	0	0	<ActionEvent	Welcome	2			0.06	
L19330	8/24/2016 13:53:46:08	0	0	<UpdatePageEvent	INTRO_ConsentScree	3			0.00	0.06
L19330	8/24/2016 13:53:46:08	0	0	<EnterFieldEvent	INTRO_ConsentScree	4	2	70.07	70.01	
L19330	8/24/2016 13:54:56:08	0	0	<ActionEvent	INTRO_ConsentScree	5			0.06	
L19330	8/24/2016 13:54:56:08	0	0	<UpdatePageEvent	xxxxdummy4	6			0.00	0.06
L19330	8/24/2016 13:54:56:08	0	0	<EnterFieldEvent	xxxxdummy4	7	3	10.79	10.27	
L19330	8/24/2016 13:55:06:08	0	0	<LeaveFieldEvent	xxxxdummy4	8			0.43	
L19330	8/24/2016 13:55:07:08	0	0	<ActionEvent	xxxxdummy4	9			0.08	
L19330	8/24/2016 13:55:07:08	0	0	<UpdatePageEvent	Section_a.AA_MilStatu	10			0.00	0.08

5. Client Side Paradata Processing

In addition to the events captured from the server side, the actions/events that occur on the client are also very useful paradata for understanding the user behavior as they interact with the Blaise 5 instrument. In particular, the analysis of the behaviors on a mobile device can provide insight into questionnaire design and user experience (Cheung et. al, 2016). This is important to understand as we move from interviewer-administered, to self-administered mode of data collection (Couper and Peterson, 2016).

Of course, the Blaise 5 native paradata captures some client-side paradata. The University of Michigan team is working on routines to supplement, capture and parse this complex data related to actions on the client-side (Peng and Ostergren, 2016). The data transformation processes for enhanced client-side paradata continue to be developed and are beyond the scope of this discussion. However, as development continues, it is our wish that the client-side paradata capture is integrated into a native solution within the Blaise software (e.g., providing a paradata API for us to write custom code against, or capturing the supplemental CSP data). The value of both the server and client-side paradata will be enhanced if these data can be linked, and analysis can more easily integrate both of these rich sources of survey paradata.

6. Aggregation and Reporting

Many different types of aggregation and reports can now be generated from this processed Blaise 5 audit data and client-side paradata (Piskorowski, 2016). To facilitate the access to this processed pardata, we have developed routines to push the data into SQL database(s).By increasing access to the paradata for analysis, there is almost no limit to the questions that we answer Some examples include:

- Timings: Total Interview Length, Interview Length by Respondent, by Mode, by Field, by Block, by Interviewer
- Surveys Abandoned: Average Number of Sessions, Session Length, Number of Sessions (by Mode) for each Respondent, Last Question Seen (Answered), Last Mode Accessed.

More details and examples can be found in Piskorowski, 2016.

7. Some Challenges, Lessons Learned and Recommendations

We have been working with the native Blaise 5 audit data for over one year now and for many beta releases. Some of the kinks are being worked out (with dedicated response and collaboration from the Statistics Netherlands Blaise 5 team at CBS). We want to share some of the challenges and lessons learned, and highlight a few of the more important enhancements that we would like to recommend.

1. The consistent sequence of events in the audit data is important!
    - When milliseconds were added to the date/time stamps, we got very valuable information for sequencing events.
    - **Challenge:** Although not occurring very often, we are still finding that some events are not sequenced properly. For example, the StartSession event is sometimes not always the first event in a new session.
2. Some other data messiness has been detected and audit data processors should be on the look-out.
    - We continue to see multiple rows with the same enter field event (with the same timestamp). Our processes have been modified to ignore these duplicate events.
    - We have also seen events without the date/timestamp information.
3. Are we getting leave fields? We really need them!
    - Although more stable now, we have found that the evolving versions of the Blaise 5 software produce different results in terms of the inclusion of leave field events when expected.
    - In order to capture leave fields, the Blaise programmer needs to set the audit trail level of capture to "Field" (or more extensive level of logging).
    - It's tricky to figure which fields should have a leave field event. Some fields that involve display of information (or click yes to continue) do not have leave field events by design.
    - At the University of Michigan, the data manager reviews the audit data with the Blaise programmer to determine which fields should not have a leave field event (**Hint:** generally the "should not have" is a much shorter list than the "should have" list).
    - **Lesson Learned:** We have established quality checks to make sure expected fields have a leave field event and to make sure that leave field events are sequenced correctly.
4. What about page-level analysis?
    - **Challenge:** The presentation of multiple fields on one page can definitely be a challenge when trying to analyze the audit data at a page-level. Mostly this presents us with an interpretation challenge (i.e., how do we identify the visit to the same page across respondents - or even for multiple visits to the same page by the same respondent?!?!). Of course, page-level analysis is much easier when there is only one field on each page.
    - Our approach has been to identify a page using the first field presented, but this does not always work.
        - The Blaise programmer controls where the cursor goes on the initial page load. So the programmer is very powerful - and should be thoughtful when setting up!
        - However, grids are one object on a page, and the question order within the grid may be randomized (so now the "first field" solution won't work).
    - The Blaise 5 audit data captures a PageIndex. However, we are still exploring how best to use. Of course, the same page number could be displayed differently to different Respondents (based on logic and/or randomization of questions). Pages may be very different across modes too!

- **Lesson learned:** We try to get everything down to the field level. Recommendations or collaborations for other approaches to page-level analyses are welcomed!

5. Combining the server-side (native Blaise 5) and the supplemental client-side paradata
   - **Challenge**: We are still working on processes for combining these two sources of paradata for analysis
   - **Recommendation:** As indicated in section 4, it is our wish that the supplemental client-side paradata capture is integrated into a native solution within the Blaise software!
6. Determining the end of a Blaise session using the paradata
   - There are various ways that a Blaise session might end including completed, aborted, expired or interrupted (and removed?)
   - In the audit data, we do get EndQuestionnaire (completed) and InterruptSession events, but we can only make inferences about expired sessions.
   - The StatNeth.Blaise.API.ServerEvents Namespace provides components for listening to server-side events (Statistics Netherlands, 2015 Help documentation), and includes expired session events.
   - **Recommendation**: It is our wish to also have other session-level events (like Expired) included in the Blaise native audit data.

8. References

Cheung, Gina, Piskorowski, Andrew, Wood, Lisa and Peng, Hueichun**. (**2016). **'**Paradata Uses'. *Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam*

Couper, Mick P. (1998). Measuring survey quality in a CASIC environment. *In Proceedings of the Section on Survey Research Methods of the American Statistical Association.*

Couper, Mick P. and Peterson, Gregg J. (2016). 'Why Do Web Surveys Take Longer on Smartphones?' *Social Science Computer Review*, ssc.sagepub.com, p 1-21.

Peng, Hueichun and Ostergren, Jason**. (**2016). **'**Paradata Capture'. *Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam*

Piskorowski, A (2016). 'Data In and Data Out'. *Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam*

Statistics Netherlands. 2015. *Blaise 5 Help Documentation. Paradata - Audit Trail.* http://help.blaise.com/

9. Appendix A – SAS Processing Code

Listing 1. SAS Processing Source Code

```
/*********Developed by the Survey Research Center at the*****************/
/********* University of Michigan for use with SAS v9.4*****************/
/***************Last updated: September 2, 2016 ***********************/
/***************Questions contact tsdataops@umich.edu ******************/

/* SAS options to set - replace [] where indicated */
options symbolgen;
options noxwait; /*External command prompts will auto-close*/
options xsync; /*SAS will wait for applications to finish before resuming
processing*/
options nodate nonumber compress=yes;

/* Date parameters */
proc format;
  picture Processdt other='%0d%0b%0y' (datatype=date);
```

```
run;

%let datetimenow=%sysfunc(date(), Processdt);
%put &datetimenow;

/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
/*!!!!!!!!!! Project-specific parameters to set !!!!!!!!!!!!!!!!!!!!!!!!*/
%let servertimeout=600; /*This represents the Blaise 5 server timeout value
(in seconds - e.g., 10 minutes=600);
This value is used as a cutoff for session breakoff timings calculations */

%let Instrumentid={[INSTRUMENTID]}; /* This is for Audit data - replace
[INSTRUMENTID] and make sure to keep the {} */
%let InstrumentName=[INSTRUMENTID]; /* This is for Session data - replace
[INSTRUMENTID], but no {} needed */

%let ServerPath=[SERVERPATH] /* This is the location of the native Blaise5
Audit data - replace [SERVERPATH] */
%let YourPath=[YOURPATH] /* This is the location of the output  - replace
[YOURPATH] */

%let keep=[YOURCRITERIA1];  /* Used for filtering out records - used for
Blaise audit data - replace [YOURCRITERIA1]*/
%let keep3=[YOURCRITERIA2]; /* Used for filtering out records - used for
Blaise session data - replace [YOURCRITERIA1]  */
/* NOTE the keep criteria may be different based on the unique identifier -
e.g., audit uses SampleId and session uses PrimaryKeyValue */

/*!!!!!!!!!! End setting project-specific parameters !!!!!!!!!!!!!!!!!!*/
/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/

/* USE IF YOU ARE STORING AUDIT/SESSION data in a SQLlite db*/
/* Make a copy of the audit and session data (SQLlite) before doing
anything else */
%sysExec copy "&ServerPath.\AuditTrailData.db"
"&YourPath.\AuditTrailData_&datetimenow..db" ;
%sysExec copy "&ServerPath.\RuntimeSessionData2.db"
"&YourPath.\RuntimeSessionData2_&datetimenow..db" ;

/*************************************************************************
* ACCESS SQLITE DATABASE FROM SAS
* REQUIREMENT:   HAVE SQLITE ODBC DRIVER INSTALLED
*************************************************************************/

/* Connection to audit db (B5 native), Session DB (B5 Native) - ODBC
connections to the SQLlite DBs need to be set-up on the user's computer */
libname Audit odbc complete="dsn=SQLite3 Datasource;
                                        Driver=SQLITE3 ODBC Driver;

Database=&YourPath.\AuditTrailData.db";

libname session odbc complete="dsn=SQLite3 Datasource;
                                        Driver=SQLITE3 ODBC Driver;

Database=&YourPath.\RuntimeSessionData2.db";

/* USE IF YOU ARE STORING AUDIT/SESSION data in a SQL db */
/*Blaise Audit Data*/
%let db=[YOURDATABASE];     /* !! ENTER DATABASE HERE */

LIBNAME b5AUDIT OLEDB
```

```
INIT_STRING = "PROVIDER=SQLNCLI11.1;
INTEGRATED SECURITY=SSPI;
PERSIST SECURITY INFO=TRUE;
INITIAL CATALOG=&db;
DATA SOURCE=&ServerPath;"
SCHEMA=&schema4 PROMPT=NO
PRESERVE_TAB_NAMES=YES PRESERVE_COL_NAMES=YES AUTOCOMMIT=NO DBMAX_TEXT =
32767;

/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
/*    NOW LET'S START THE PROCESSING !!!!                                    */
/* STEP 1: First step for RAW ADT is to merge Session and Event data   */
/* Let's also rename KeyValue as SampleId */
proc sql;
create table RawADT1 as
      select a.KeyValue as SampleId,
             b.instrumentId,
             b.SessionID,
             b.TimeStamp,
             b.Content
      from audit.AuditSessionData a join audit.EventData b
      on a.SessionID = b.SessionID and a.InstrumentID = b.InstrumentID
            where b.instrumentid ="&instrumentID"
      order by a.KeyValue,b.timestamp,b.sessionid;
quit;

/* Use if you want to filter out any cases */
data RawADT1a;
  set RawADT1a;
  if &keep.;
run;

/*Get list of only Start Session events*/
data RawADT2;
      set RawADT1;
      where Content like ('%StartSessionEvent%');
run;

/*Get list of EnterField events*/
data RawADT3;
      set RawADT1a;
      where Content like ('%EnterFieldEvent%');
run;

/* STEP2: Let's start with some parsing
NOTE: THIS STEP NEEDS SOME UDPATES FOR ADDITION OF CATI */
data work.RawADT4 ;
      set RawADT1a;
      Seq = _N_ ; /* Create a "overall" sequence for this Instrument(s)
data - as found in the DB */
/*Data is already sorted by SampleId then timestamp - see STEP 1*/
/*This is crucial to the paradata tables - not adding explicit sort in the
assumption */

/* Flag the Session Start/End Events and parse out session level data */
      format SessionStart SessionEnd IWComplete 8.;
      format EventType $50.;
      format Width Height $8.;
      format Browser Language LoginId Platform $20.;
      format AgentString $200.;
      SessionStart = index(content, "<StartSessionEvent");
```

```
/* Flag the Start Session Event */
      SessionEnd = index(content, "<InterruptSessionEvent");

/* Flag the Interrupt Session Event */
      IWComplete = index(content, "<EndQuestionnaire");
/* Flag the End Questionnaire Event*/
      EventType = strip(scan(content,1,' '));
      if IWComplete ne 0 then SessionEnd=1;
/* Also flag the end of the Q as the session end */
      if SessionStart ne 0 then Width = strip(scan(content,2, '"'));
/* Width is the first double quote value in start session */
      if SessionStart ne 0 then Height = strip(scan(content,4, '"'));
/* Height is the second double quote value in start session*/
      if SessionStart ne 0 then Browser = strip(scan(content,8, '"'));
/* Browser is the fourth double quote value in start session
(ignore third, which says device, but it isn't) */
      if SessionStart ne 0 then Language = strip(scan(content,10, '"'));
/* Language is the fifth double quote value in start session */
      if SessionStart ne 0 then LoginId = strip(scan(content,12, '"'));
/* LoginId is the sixth double quote value in start session*/
      if SessionStart ne 0 then Platform = strip(scan(content,14, '"'));
/* Platform is the seventh double quote value in start session*/
      if SessionStart ne 0 then AgentString = strip(scan(content,16, '"'));
/* AgentString is the eighth double quote value in start session*/

/* Flag the Enter Field Events and parse out field level data - note:
Start/End session and IwComplete are considered field events */
      format EnterField PageUpdate LeaveField  8.;
      format FieldName Block $100.;
      format AnswerStatus $10.;
      format Answer $100.;

      /* Add more flags for each type of event */
      EnterField = index(Content,"<EnterField" );
      PageUpdate = index(Content,"<UpdatePageEvent" );
      LeaveField = index(Content,"<LeaveFieldEvent" );
      PageAction = index(Content,"<ActionEvent" );

if SessionStart ne 0 or IWComplete ne 0 or SessionEnd ne 0 or LoginPassed
ne 0 then EnterField=1; /* Also flag these as Enter Field events */

      /* Now let's get the field name */
      if EnterField ne 0 then FieldName = strip(scan(content,2, '"'));
      if LeaveField ne 0 then FieldName = strip(scan(content,2, '"'));
      if SessionStart ne 0 then FieldName = "StartSession";
      if IWComplete ne 0 then FieldName = "EndQuestionnaire";
      if IWComplete =0 and SessionEnd ne 0 then FieldName =
InterruptSession";
      *if PageUpdate ne 0 then FieldName="PageUpdate";
      Block = strip(scan(FieldName,1, '.'));

      /* Let's capture if response was given at leave field */
      if LeaveField ne 0 then AnswerStatus = strip(scan(content,4, '"'));
      if LeaveField ne 0 and AnswerStatus ="Response" then
Answer=strip(scan(content,6, '"'));
      WHERE timestamp ^= .; /* This removes any records without the
timestamp*/
run;
```

```
/* This is to remove the "extra" enter field events - i.e., one that is
right after another */
/* If it is a EnterField event and the row preceding is also enter field
event, then keep the 1st occurrence of the Enter */
/* We have to index for EnterField again because the code above also
assigns some events (e.g., StartSession) as EnterField */
/* Sort to correct for the fact that StartSession events don't always go to
the top of each session because of duplicate timestamps, missing
milliseconds*/
proc sort data=RawADT4;
      by SampleId TimeStamp  descending SessionStart ;
run;

/* Now taking care of some duplicate events */
data rawADT4a;
 set rawADT4;
 EnterOnly = index(Content,"<EnterField" );
 EnterDupe=0;
 if EnterOnly=1 and lag(EnterOnly)=1 and lag(seq)=Seq-1 then EnterDupe=1;
run;

data RawADT4_noEnterDup (drop=EnterDupe EnterOnly);
  set rawADT4a;
  if EnterDupe=0;
run;

/* Data with and without Category events */
data RawADT4_noCategoryEvent;
      set RawADT4_noEnterDup;
      if EventType ne '<CategoryEvent';
run;

data RawADT4_CategoryEvent;
      set RawADT4_noEnterDup;
      if EventType='<CategoryEvent';
run;

/* Now we are populating FieldName and Block for all records - use the row
above if missing */
data RawADT4b;
  set RawADT4_noCategoryEvent;
  retain _FieldName;
  if not missing(FieldName) then _FieldName=FieldName;
  else FieldName=_FieldName;
  drop _FieldName;
  Block = strip(scan(FieldName,1, '.'));
run;

proc freq data=rawADT4b;
  table fieldname /out = adt_fieldnames noprint;
run;

/*Here, we are sorting descending, because the previous code assigns update
page(first row of page) to last page, so Im now doing it in reverse
to assign the update page to the row after - a reverse lag of sorts -
Piskorowski*/
proc sort data= rawADT4b;
      by sampleid descending Seq ;
run;

data rawadt4c;
```

```
set rawadt4b;
by sampleid;
FieldN = lag1(FieldName);
if index(content, 'UpdatePageEvent') > 0 then FieldName = FieldN;
Block = strip(scan(FieldName,1, '.'));
drop FieldN;
*SID = lag(sampleid);
*seqn = lag(Seq);
*end;
run;

/*Sort it back to normal*/
proc sort data=rawadt4c;
      by SampleID Seq;
run;


/*Now let's add Event/Session Sequence and Session Counters */
data RawADT5;
      set RawADT4c;
      by SampleId;
            EventSeq+1;
            If First.SampleId = 1 then do;
                  SessionNum = 1;
                  EventSeq = 1;
                  End;
            If First.SampleId ne 1 and SessionStart=1 then SessionNum+1;
            if SessionNum < 10 then
NewCount=Compress(SampleId||"00"||SessionNum); /* We are adding these
leading zeros so that Rs with 10+ 100+ sessions are ordered correctly*/
            if 10 <= SessionNum < 100 then
NewCount=Compress(SampleId||"0"||SessionNum);
            if SessionNum >= 100 then
NewCount=Compress(SampleId||SessionNum);
run;

/* REVIEW WITH AP/JENNIE this code that adds mode and AttemptNumber */
data RawADT5b (drop=NewCount);
  set RawADT5;
  by NewCount;
            If First.NewCount = 1 then do;
                  SessionSeq = 1;
            End;
  If First.NewCount ne 1 then SessionSeq+1;
  if last.NewCount then SessionEnd=1;  /* We also need to flag the
SessionEnd - i.e., last Record for the SID/Session is the SessionEnd */
  if last.NewCount then SuspendField=FieldName;
  AttemptNumber = strip(SampleId) || '_' || put(SessionNum,z2.);
  /*EDIT these per Survey - Programmers use different laynames for various
versions

/*********************************************************************/
Format Mode $10.;
if index(content, 'LayoutSetName="Interviewing1"') > 0 then Mode = 'Web';
if index(content, 'LayoutSetName="Web"') > 0 then Mode = 'Web';
if index(content, 'LayoutSetName="Cati"') > 0 then Mode = 'CATI';
if FieldName = 'StartSession' then Mode = 'Web';
run;

/*Getting a better mode variable here, putting all rows as Login, CATI or
Web*/
```

```
data RawADT5c;
  set RawADT5b;
  retain _Mode;
  if not missing(Mode) then _Mode=Mode;
  else Mode=_Mode;
  drop _Mode;
  /*EDIT these per Survey - Programmers use different laynames for various
versions

/*************************************************************************/
  if index(content, 'UpdatePageEvent') > 0 then Page = strip(scan(content,
4, '"'));
  if index(content, 'StartSession') > 0 then Page = 'StartSession';
  if index(content, 'SwitchSession') > 0 then Page = 'PassedintoSurvey';

  retain _Page;
  if not missing(Page) then _Page=Page;
  else Page=_Page;
  drop _Page;

  run;
proc sql;
create table pagefields as
select distinct page, fieldname from
rawadt5c
where page is not null; quit;

/*Here we're going to use the events 'UpdatePageEvent' Action Event
NextPage and Previous page for the timings on a page.
We'll also identify each page using the page index. We'll link this later
to a crosswalk*/
data rawadt6a;
 set RawADT5c;
if index(content, 'UpdatePageEvent') > 0 or index(content, 'ActionEvent') >
0 then output rawadt6a;
run;

/* Now we are going to separate the EnterField Events from all other events
so that we can add Field Counters and Time between fields */
data RawADT6a RawADT6b ;
  set RawADT5c;
  if EnterField=1 then output RawADT6a;
  else output RawADT6b;
run;


proc sort data= RawADT6a;
  by SampleId SessionNum descending SessionStart;
run;

data RawADT6a2;
      set RawADT6a;
            By SampleId SessionNum;
            FieldSeq+1;
            If First.SampleId = 1 then do;
                  FieldSeq = 1;
            End;
run;

proc sort data=RawADT6a2;
      by SampleId descending FieldSeq;
```

```
run;

data RawADT6a3 ;
      set RawADT6a2;
      format FieldTimeSec 20.4;
      FieldTimeSec = Lag(Timestamp) - TimeStamp;
      if FieldTimeSec > &servertimeout then FieldTimeSec=0;
      if SessionEnd=1 then FieldTimeSec=.;
run;

proc sort data=RawADT6a3;
      by SampleId FieldSeq;
run;

/* Combine together again and sort so that we can add duration calculation
(EventTime) */
data RawADT6;
      set RawADT6a3 RawADT6b;
run;

proc sort data=RawADT6;
  by SampleId descending EventSeq;
run;

data RawADT7;
      set RawADT6;
      format EventTimeSec 20.4 ;
      EventTimeSec = Lag(Timestamp) - TimeStamp;
      if EventTimeSec > &servertimeout then EventTimeSec=0;
      if SessionEnd=1 then EventTimeSec=.;
run;

data RawADT7b;
  set RawADT7;
  PreviousField = lag(FieldName);
run;

proc sort data=RawADT7b;
      by SampleId EventSeq;
run;

data RawADT_final;
  set RawADT7b;
  format PageLoadSec 20.4 ;
  PageLoadSec = Lag(EventTimeSec) ;
  if PageUpdate=0 then PageLoadSec=.;
run;
```

10. Appendix B: Data Dictionary of Processed Data

Figure 1. Data Dictionary of Processed Data

FieldName	Type	Length	VarNum	Field Description	Source	Notes
SampleId	VarChar	255	1	KeyValue	Blaise Native	Rename of KeyValue
InstrumentId	VarChar	40	2	InstrumentId	Blaise Native	
SessionId	VarChar	40	3	SessionId	Blaise Native	Each SampleId has unique SessionId, but all sessions for a SampleId use the same SessionId
TimeStamp	Numeric	8	4	TimeStamp	Blaise Native	We are getting all the milleseconds with more recent build!
Content	VarChar	1024	5	Content	Blaise Native	
Seq	Numeric	8	7	Overall Sequence	Computed	Basically the same as the record # for the data set (to preserve the sequence as written by the Blaise native)
SessionStart	Numeric	8	8	Flag to show that this record indicates the START of a session	Computed	0/1
SessionEnd	Numeric	8	9	Flag to show that this record indicates the END of a session	Computed	0/1
IWComplete	Numeric	8	10	Flag to show that this record indicates the IW was Completed	Computed	0/1
EventType	VarChar	50	11	Text describing the type of Event	Computed	Rename of SessionType (in SRC original code)
Width	VarChar	8	12	Width (at Session Start=1)	Computed	This information is populated only on Session Start events
Height	VarChar	8	13	Height (at Session Start=1)	Computed	This information is populated only on Session Start events
Browser	VarChar	20	14	Browser (at Session Start=1)	Computed	This information is populated only on Session Start events
LoginId	VarChar	20	15	LoginId (at Session Start=1)	Computed	This information is populated only on Session Start events
Platform	VarChar	20	16	Platform (at Session Start=1)	Computed	This information is populated only on Session Start events
AgentString	VarChar	120	17	AgentString (at Session Start=1)	Computed	This information is populated only on Session Start events
Language	VarChar	20	18	Language (at Session Start=1)	Computed	Looks like a proxy for mode
EnterField	Numeric	8	19	Flag to show that this record is considered an EnterField event	Computed	0/1 (note: sometimes we get 2 enter field events in a row)
PageUpdate	Numeric	8	20	Flag to show that this record is considered an PageUpdate event	Computed	0/1 (note: the FieldName on this row indicates the field that is being exited)
LeaveField	Numeric	8	21	Flag to show that this record is considered an LeaveField event	Computed	0/1 (note: we don't always get leave field for each Q, but we can start to examine)
FieldName	VarChar	100	22	Field Name - populated for all records	Computed	If this is not provided in Contents, populated using the FieldName from the Row above
Block	VarChar	100	23	Block	Computed	Based off of Field Name
AnswerStatus	VarChar	10	24	Answer Status (if LeaveField=1)	Computed	Values of Empty/Response
Answer	VarChar	100	25	Answer (if LeaveField=1 and AnswerStatus=Response)	Computed	
PageAction	Numeric	8	26	Flag to show that this record is considered an Action event	Computed	0/1
EventSeq	Numeric	8	27	Counter of events for the SampleId (across sessions)	Computed	
SessionNum	Numeric	8	28	Number of Sessions for the SampleId	Computed	
SessionSeq	Numeric	8	29	Counter of events within a Session for each SampleId	Computed	
SuspendField	VarChar	100	30	Last Question accessed in that Session (if EndSession=1)	Computed	
Mode	VarChar	10	31	Using "LayoutSetName" to determine mode	Computed	Most useful if a survey is using multiple modes. Some custom SAS editing may be required.
Page	VarChar	20	32	PageIndex (as provided by Blaise native)	Computed	
FieldSeq	Numeric	8	33	Counter of Fields accessed for the SampleId (across sessions)	Computed	
FieldTimeSec	Numeric	8	34	Total Time from Field to Field (i.e., between each "EnterField")	Computed	
EventTimeSec	Numeric	8	35	Time between each event	Computed	If there is a session break-off, there is no EventTimeSec for the last event in the session
PreviousField	VarChar	100	36	Simply a "LAG" of the field - which returns the field from the row prior	Computed	See SAS LAG function for more details
PageLoadSec	Numeric	8	37	Time took for page to update to the next page	Computed	FieldName indicated is the last field before the Page Update (i.e., the field being EXITED not ENTERED)