Blaise 5 Data In/Data Out

Andrew D. Piskorowski, University of Michigan Survey Research Center, United States

1. Abstract

The focus of this presentation is to demonstrate various methods used to move data in and out of the Blaise 5 databases. We will show you how the team at the University of Michigan is combining custom tools with Manipula to improve the process of accessing the Blaise 5 data.

1.1 Survey Preload

We have a windowed application that will preload a Blaise 5 data model. In addition to simplifying the preload process, this application allows for incremental updating and adding of sample rows to the project database.

1.2 Survey Migration

Sometimes during data collection, a new version of the survey questionnaire may need to be released. This may happen from an error in original programming, an update in question text, or a change to response options.

This data migration (from the old version to the new version) is needed can be challenging if respondents have already started and suspended on the old version. We want to ensure that we do not lose their data from the old version, and we want to make sure that they can resume the survey where they broke off. We will demonstrate our process for this complex data migration.

1.3 Survey Download

We have created various processes for getting data out of Blaise 5.

- **Manipula and SAS/Metadata** - We have developed automated processes for delivery of raw data from Blaise5. This includes export to a SAS database which integrates code frames and other metadata.
- **MQDS** - Our latest version of MQDS also allows for the downloading of the survey question text and other metadata. This is useful in creating a survey crosswalk - a spreadsheet that maps the survey data responses to the survey questionnaire. MQDS also allows for exporting selected fields into a text file.

2. Introduction

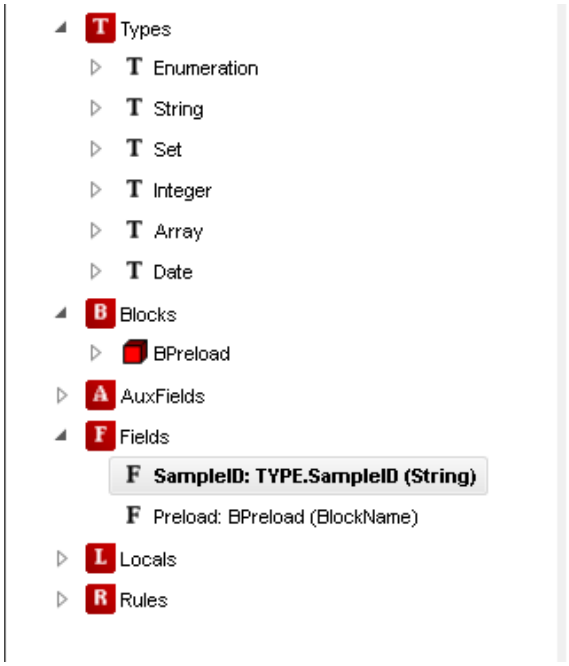
Over the evolution of Blaise 5 there have been many avenues to access the data. In this paper we will review various ways we populate the Blaise databases prior to data collection and how we export data during and after data collection – i.e., data in, data out.

3. Survey Preload

With known survey populations we know some information about the people we are gathering information from and we often look to use this during data collection. This includes respondent attributes like name, gender, number of children, etc. With Blaise 4.8, Manipula is used to preload the Blaise databases to be used by survey logic to control different actions and display functions within a survey. In the early rounds of Blaise 5 there was a learning curve to both programming surveys but also to converting many of the traditional functions and routines previously completed with Manipula.

With the new structure of Blaise 5, a unique compiled data model [.bmix] containing a subset of the main instrument blocks is required to map preload information back to the main instrument database.

Figure 1. Survey Preload



At SRO, creation of this separate preload data model structure is created by our Blaise programmers. Our data manager group has created a simple Manipula script that imports the preload information including the SampleId which is used to access the instrument. Below is an example script. This script uses the preload bmix file to load the values appropriately into the main data model instrument.

Listing 1. Manipula Script Source Code

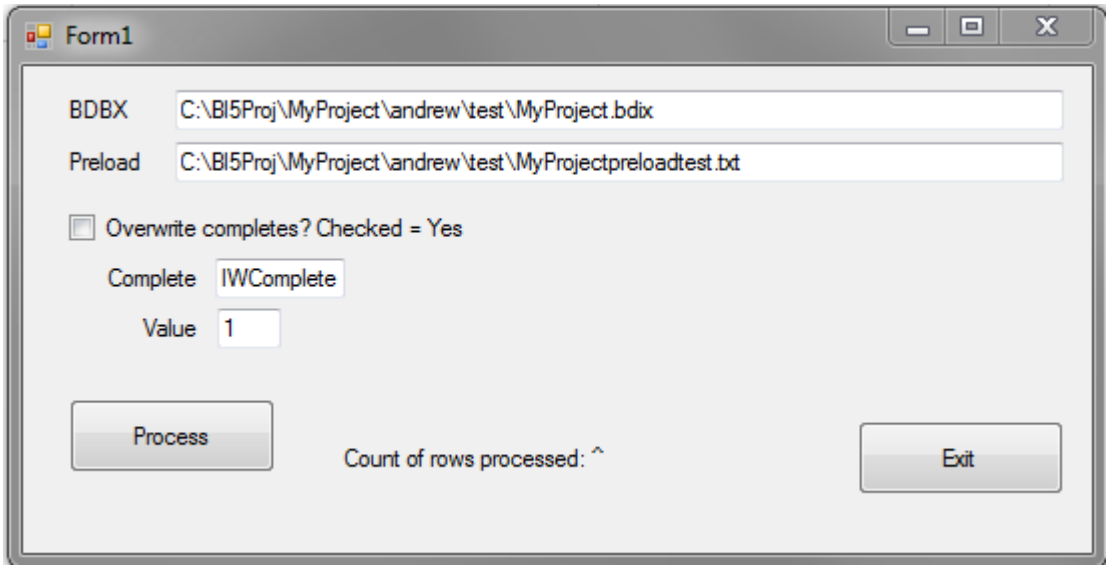
```
SETUP MySetup

settings
  DATEFORMAT = 'MMDDYY'
  DATESEPARATOR = '/'

USES
ImportMeta  'MyProjectPreload'

MyMeta  'MYPROJECTProd'

INPUTFILE
 MyInputFile: ImportMeta ('MyProjectpreloadtestwithoutheaders.txt',
ASCII)
 SETTINGS
  SEPARATOR = ','

UPDATEFILE
 MyOutputFile: MyMeta ('MYPROJECTProd', BLAISE)

MANIPULATE
 MyOutputFile.WRITE
END
```

This method allows for customization of the preload import for different data formats and a more hands-on approach to working with the data.

There are a few downsides of the Manipula approach. The script needs to be recompiled each time the data model is updated. This is required even when the information you are preloading has not changed. The file name containing the preload information is also hard-coded into the script.

As an alternative to Manipula, we created a windowed application, using the Blaise DataLink and DataRecord API , that will preload a Blaise 5 data model for use of the data in a survey administration. In addition to simplifying the preload process, the application allows for incremental updating and adding of sample rows to the project database.

Figure 2. Manipula Alternative Preload



The program takes two arguments: a Blaise Data Interface file [.bdix], and a caret delimited file to be loaded. The program overwrites entries with the same primary key but only if not complete, unless otherwise specified. A verification of the number of records (rows) updated is also provided. The program allows for one not as familiar with Manipula to load an instrument with preload.

4. Survey Migration

At times during data collection, a new version of the survey questionnaire may need to be released. This may happen because of an error in the original instrument programming or because of a needed update to question text or response options.

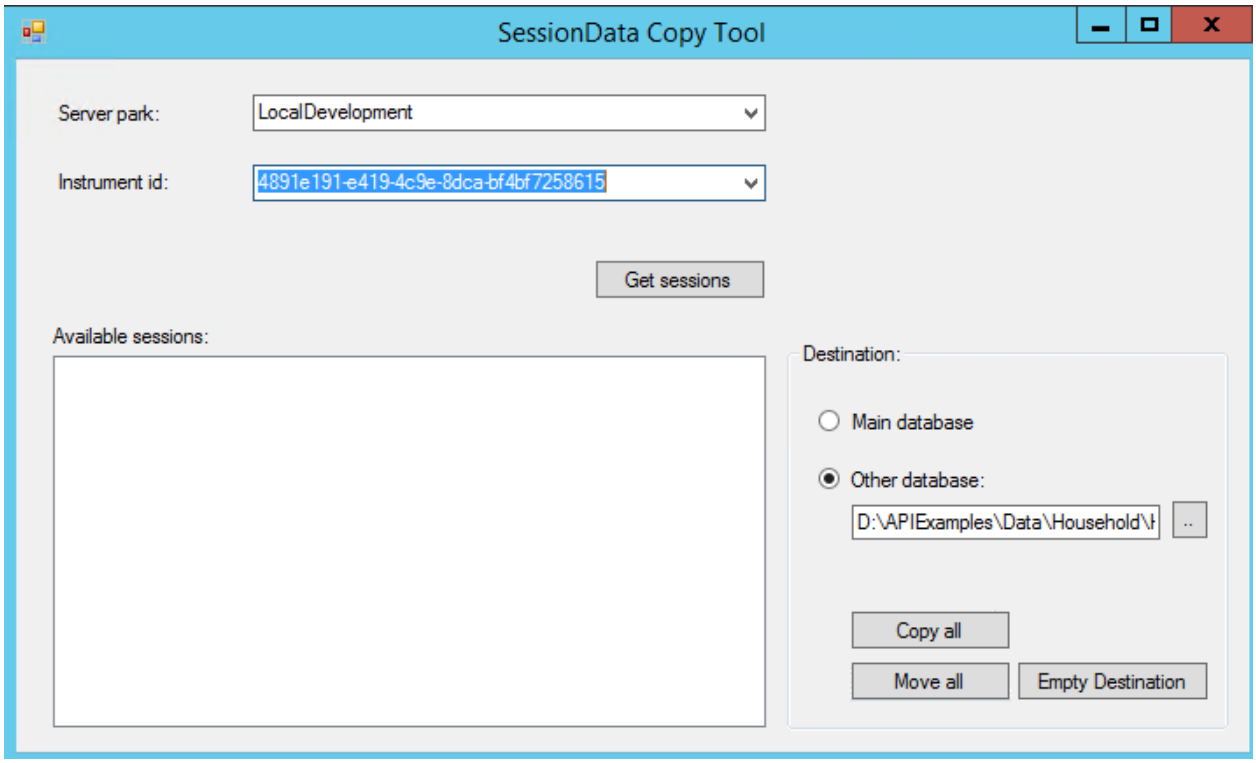
To implement this change, a data migration from the old version to the new version is needed. If a respondent has already started and suspended on the old version, we need to ensure that we do not lose their data from the old version, and we want to make sure that they can resume the survey where they broke off. This complex data migration process involves:

1. Accessing the partial, not yet completed, data stored in the Session database
2. Merging them with the completed data
3. Finally, migrating both the old version completed and partial cases to the new version of the survey questionnaire.

Programming logic was added to ensure partial respondents did not have to resume from the beginning.

To access the partial data stored as blob files in the RuntimeSessionDatabase we developed a custom application to populate an empty database of old instrument.

Figure 3. Populate Application



To use this application we created a separate server environment for data managers to utilize as this requires the installation of a copy of the survey to be migrated.

Migration Process

1. We install the current version of the survey.
2. Overwrite the empty session database with the current production session database.
3. Run the SessionData Copy Tool
    a. Enter the Instrument Id of the sessions to be downloaded.
    b. Clicking 'Get sessions' verifies there are sessions available.
    c. Clicking 'Main database' then 'Copy all' copies the available sessions to the empty database.
4. Run a Manipula [bdbx] to [bdbx] script that copies the partial cases to the new instrument database.

This allows us to preserve respondents data who have already started the instrument and eliminate their burden of having to start the survey over.

5. Survey Data Download

We have created various processes for getting data out of Blaise 5. Many of our projects need a daily delivery of raw data from Blaise 5 to use for analysis and data QC. We developed an automated process that uses a standard Manipula script for data export, batch file processes, and SAS.

Below is a standard script to export completes:

Listing 2. Export Source Code

```
SETUP MYPROJECT
SETTINGS
 DESCRIPTION = "My Manipula Setup"
```

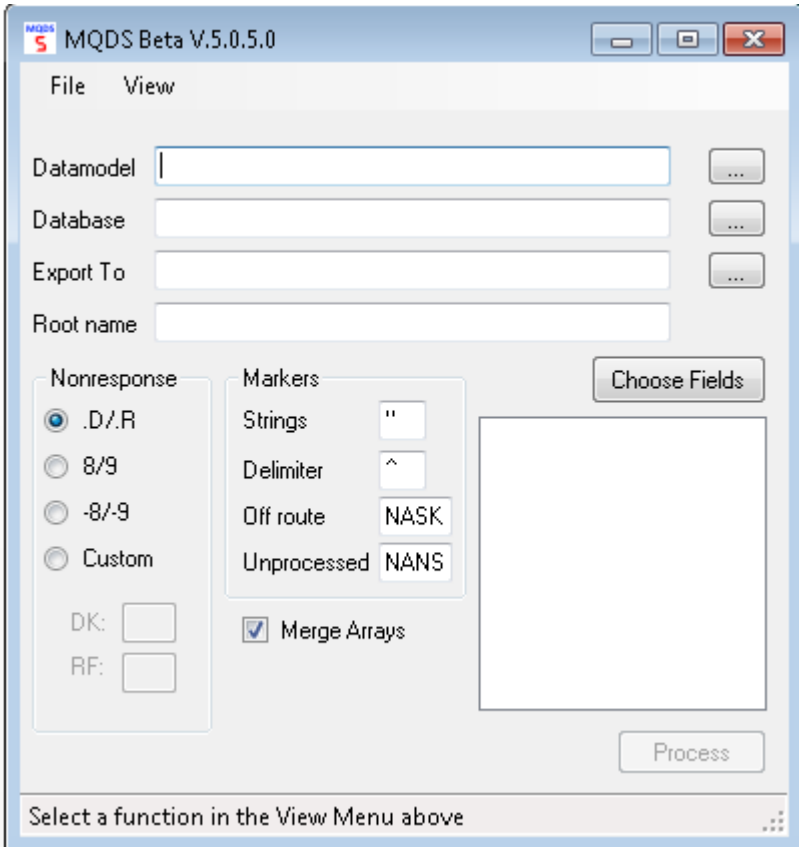
```
{ CHECKRULES = YES}
USES
 MetaID 'MYPROJECT.bmix'

INPUTFILE
  InputID: MetaID ('MYPROJECT.bdbx', BLAISE)

OUTPUTFILE
  OutputID: MetaID ('MYPROJECT_CompletesNASK.txt', ASCII)
  SETTINGS        HEADERLINE = YES
     SEPARATOR = '^'

MANIPULATE
IF InputID.IWComplete = 1 THEN
 OutputID.WRITE
ENDIF
END
```

A batch file runs the manipula and SAS import to create a new file each day. This file is a raw dataset that can be used by our analysts.

6. MQDS

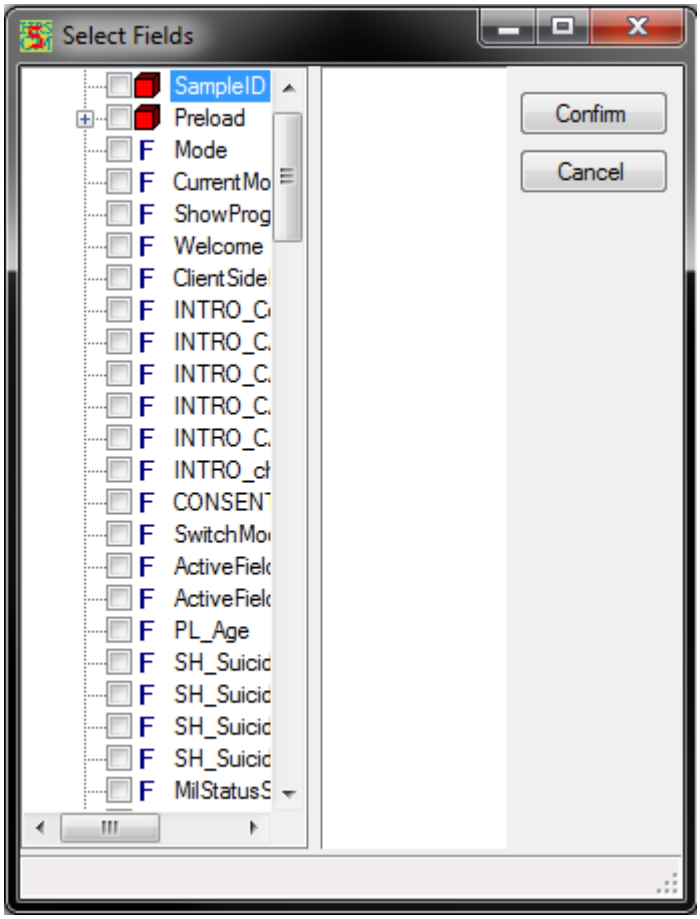
Our latest version of MQDS allows for the downloading of the survey question text and metadata for use in creating a survey crosswalk - a spreadsheet that maps the survey data responses in the dataset created above to the survey questionnaire. The program currently exports the data into separate files for each block within the instrument.

Figure 4. MQDS



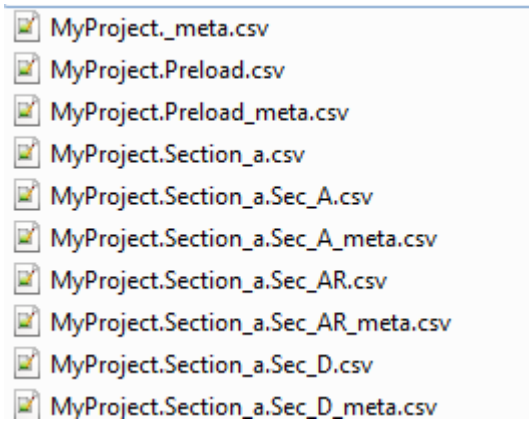
Once the Datamodel is selected, Database, Export To, and Root name are prefilled using the same directory. All can be edited, however. The utility allows for exporting non-response from a Don't Know and Refusal to a few common values or custom options. Under Markers, we are able to select a string identifier and a delimiter. The utility also allows for selecting values for off-route or unprocessed. The utility offers the ability to select only certain fields as well.

Figure 5. Select Fields



As stated earlier, the utility provides separate files for each block but also separate files for the meta data and data values.

Figure 6. Files



A data file example:

Figure 7. Data File Example

```
SampleID^xArmymilitary^A_AA_ETS^A_AA_Checkpoint_01^A_AA_ThinkReEnlist^A
"IT656436"^^1^1^1^2^^^^5^1^^^1^1^1^1^1^1^1^1^1^^1^2016^^^^^^^^^^^^9-12
"IT656566"^^5^2^2^2^^^^2^2^^^2^3^2^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
"IT657269"^^5^2^4^1^^^^3^2^^^^2^2^^^^^^^^^^2-3-4^^^^^^^^^^^^^^^^^^^^^^
```

A meta file example:

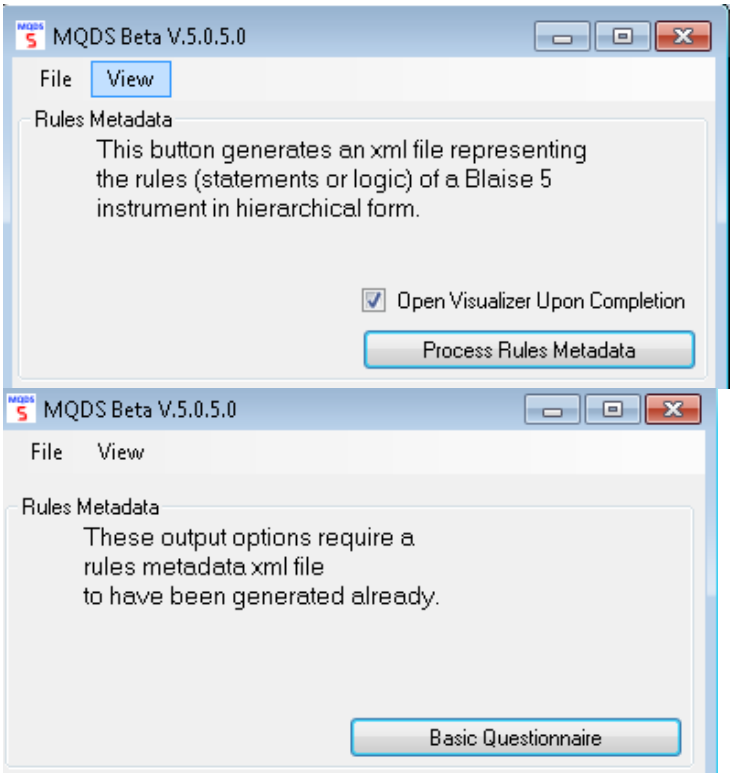
Figure 8. Meta File Example

Name	Structure	Categories	Min	Max	MaxLength
xArmymilitary	String				10
A_AA_ETS	Enumeration	1. Yes,5. No			
A_AA_Checkpoint_01	Integer		1	2	
A_AA_ThinkReEnlist	Enumeration	1. _1,2. _2,3. _3,4. _4,5. _5			

The meta file allows for creation of a SAS file to add value labels and formats to the previously exported data values. Currently, this process involves a manual creation of the SAS program but our next step is to automate this process to create a fully prepared SAS file that includes the appropriate labels and formats.

Within the MQDS utility we are able to export an XML file containing the rules to a selected data model file [bmix]. This file is used to create other more useful files and information like the ability to create a text file of the questionnaire. This file is used in conjunction with the meta data files to create a complete crosswalk for documentation of the instrument - directly from the created Blaise instrument. Below are screen captures of the options.

Figure 9. MQDS Options



Here is an example of the questionnaire that is exported.

Figure 10. Exported Questionnaire

```
***** Section_a.AA_MilStatus_b *****
Current Military status - b
************************************
Are you currently on active duty in the Regular Army, another branch of t

1. Regular Army
2. Other branch of the military
3. Army Reserve
4. Army National Guard
5. Other reserve component

Answer not required/Don't Know allowed/Refusal allowed

***** Section_a.AA_MilStatus_c *****
Current Military status - c
************************************
Are you currently in the reserves, separated from the Army, retired from

1. In reserves (not active duty)
2. Separated from the Army
3. Retired from the Army
```

Using all the files created above we create an Instrument crosswalk that is used by analysts and clients as documentation of the data. Below are the fields we include in our crosswalk.

Table 1. Crosswalk Fields

COLUMN	MEANING
#	Row number indicating a unique data point
Instrument	Instrument Name
Phase	Wave of data collection
CATI	Indicates whether this data point appears in CATI mode
WEB	Indicates whether this data point appears in WEB mode
Section Label	Section label given to each block or group of questions
Section Designation	Abbreviation for section or block
In DataModel	Indicates whether the field is included in data delivery (contains a data point)
Variable Name	Variable name from the instrument or renamed if changed during processing
Variable Label	Label created from variable name and description of question
Blaise Tag	Unique Blaise tag from Blaise
Section Letter	Larger grouping of similar blocks/sections
Question Number	Question ID
Code Frame	Code frame as programmed in Blaise instrument
Data Type	Data type
Data Range	Data range of values

Table 2. Crosswalk Fields

Instrument	Phase	CATI	WEB	Section Label	Section Designation	In DataModel	Variable Name	Variable Label	Blaise Tag	Section Letter	Question Number	Code Frame	Data Type	Data Range
MyProject	Pilot	0	1	Your Army Car	AA	1	AA_MilStatus	Current Military status	A01_Q	A	A1	MilStatus	Integer	1-14
MyProject	Pilot	0	1	Your Army Car	AA	1	AA_Checkpoint_01	Your Army career and beyond	A02_X	A	CKPT.A1.1	CKPTA11	Integer	1-2
MyProject	Pilot	0	1	Your Army Car	AA	1	AA_MilStatus2	Current Military status 2	A03_Q	A	A1.1	MilStatus	Integer	1-14
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_a	Current Military status – a	A06_Q	A	A1	Yes_No	Integer	1-5
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_b	Current Military status – b	A07_Q	A	A2	MilStatus2	Integer	1-5
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_c	Current Military status – c	A08_Q	A	A3	Milstatus3	Integer	1-3
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_d	Current Military status – d	A09_Q	A	A3a	MilStatus4	Integer	1-3
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_e	Current Military status – e	A10_Q	A	A3b	MilStatus5	Integer	1-3
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_f	Current Military status – f	A11_Q	A	A3c	MilStatus5	Integer	1-3
MyProject	Pilot	1	1	Your Army Car	AA	1	AA_OverallFeel	Overall feeling about militar	A04_Q	A	A1.2	PosNeg	Integer	1-5
MyProject	Pilot	1	1	Your Army Car	AA	1	AA_Checkpoint_02	Your Army career and beyond checkpoint (		A	CKPT.A1.2	CKPTA12	Integer	1-5
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_ETS	Army - Do you have an ETS da	A_A01_Q	A	A.AA1	Yes_No	Integer	1-5
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_Checkpoint_01	Active duty regular army chec	A_A02_X	A	CKPT.A.AA2	CKPTAAA2	Integer	1-2
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_ETSMonth	Army ETS - month	A_A03_Q	A	A.AA2	Month	Integer	1-12
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_ETSYear	Army ETS – year	A_A04_Q	A	A.AA2		Integer	2015-2021
MyProject	Pilot	0	1	Your Army Car	A_AA	1	A_AA_ThinkReEnlist	Army - Will be given the optic	A_A05_Q	A	A.AA3	Option	Integer	1-5

7. Audit Data Processing

The AuditTrailDatabase is a useful source of data providing valuable paradata that, once extracted and processed, can be used in many ways. At the University of Michigan we extract the data using SAS, perform data manipulation, and then export the data into a SQL Server to reside alongside the rest of our study data, including data from our survey management system. This allows us to use the data collected in the audit trail in conjunction with other data sources during production.

The AuditTrail is originally stored in a sql lite database. We used the following connection string to connect the database in SAS and parse the 'Content' field into the table that follows.

Listing 3. Connection String Source Code

```
libname Audit odbc complete="dsn=SQLite3 Datasource;
Driver=SQLITE3 ODBC Driver;
Database=Z:\DataManagers\LS\PilotEndGame\Audit\AuditTrailData.db";
```

Table 3. Audit Trail

VARIABLE	DESCRIPTION	HOW ITS CALCULATED
Content	Raw Content from Blaise	
LoginId	Uniqe primary key from Blaise	
SessionNumber	Computed (LoginId_SessionNum)	Each start session indicates a new session
InstrumentId	From Blaise	
SessionId	From Blaise	
TimeStamp	From Blaise	
Survey	Login or Survey Data Model	If multiple data models are used, we identify from instrument
Seq	Sequence of rows	Row count in order recorded
SessionStart	Computed using Start Session Events	Flag for StartSession events
SessionEnd	Computed using Start Session Events	Flag for last row within session
IWComplete	Computed from EndQ	Flag for EndQuestionnaire event
Width	Width of screen	Parsed from Content
Height	Height of screen	Parsed from Content
Browser	Browser used for session	Parsed from Content
AgentString	Parsed from Content	Parsed from Content
EnterField	Computed from Events (Enter Field)	Flag for EnterField event
PageUpdate	Computed from Events (Page Update)	Flag for PageUpdate event
LeaveField	Computed from Events (Leave Field)	Flag for LeaveField event
FieldName	Computed from Events (Enter Field)	Computed from Events (Enter Field)
Block	Computed from Events (Enter Field)	Computed from Events (Enter Field)
AnswerStatus	Computed from Events (Leave Field)	Computed from Events (Leave Field)
Answer	Computed from Events (Leave Field)	Computed from Events (Leave Field)
EventSeq	Sequence of Events by SampleLineId	
SessionNum	Computed Session Number	Number of current session by LoginId
SessionSeq	Sequence within Session	
SuspendField	Last field seen by respondent	Last Field in Audit Trail for that Session
FieldSeq	Sequence of Enter Fields entered	Count of all enter fields within a session
FieldTimeSec	Time spent on field	Time between EnterField to EnterField

EventTimeSec	Time spent on particular field	Time between current event to next event
PageLoadSec	Time for page to load (server)	Time between navigation click and PageUpdate event
SessionMode	From Start Session	Using parameter passed in through Langurage to indicate
AuditLayout	From Layout, to pick up/filter out Login	Parsed from PageUpdate indicating layout used for

Further, we aggregate some of this data at the respondent level along with data from our survey management system into the following table:

Table 4. Aggregated Data

COLUMN	MEANING
LoginId	Uniqe primary key from Blaise
SampleLineID	Unique key within survey management system
FirstString	First agent string provided per respondent
FirstPlatform	First platform used respondent
FirstBrowser	First browser used respondent
FirstWidth	First width of browser
FirstHeight	First height of browser
LastString	Last agent string provided per respondent
LastPlatform	Last platform used respondent
LastBrowser	Last browser used respondent
LastWidth	Last width of browser
LastHeight	Last height of browser
CATIDone	Indicates respondent completed instrument on CATI mode
WebDone	Indicates respondent completed instrument on WEB mode
SurveyComplete	Overall complete indicator
AuditCompleteDT_EST	Complete time of instrument in EST from audit trail
AuditCompleteDT_UTC	Complete time of instrument in UTC from audit trail
LastField	Last field respondent scene from audit trail
LastQAnswered	Last field respondent answered from audit trail
CompletionMode	Overall variable to indicate which mode completed on
AuditvMSMS_Check	Compares survey status of audit trail (complete or not) with status of MSMS (Michigan Survey Management System)
AuditEnd_Check	Flag to indicate if EndQuestionnaire is last event for a complete case
TotalTimeSurvey	Total time spent in survey
LoginAttempts	Number of times they attempted to login
LoginSuccess	Number of times they successfully logged in
LoginFails	Number of times they failed to login
AuditStatus	Overall indicator of survey status (Not started, started, complete)

This data allows us to follow closely, in almost real time, how a respondent is progressing through the survey. It also allows us to QC the data against our survey management system in some key areas including completion status, time to complete, and in which mode.

8. In The End

Data out of an instrument and its connected systems is only as good as the data in. It's very similar to the art of painting the inside of your house - the end results are directly correlated to the prep work

one puts in. As both Blaise and client's needs evolve we have been able to use custom applications, native tools and creative thinking to ensure we're creating a product that provides the good end data all of our users desire.