

**Proceedings of the 18<sup>th</sup>  
International Blaise Users Conference**

**IBUC 2018**

**Baltimore, Maryland USA  
October 23 – 25, 2018**



## Preface

This document contains the papers presented at the 18th International Blaise Users Conference (IBUC) held on October 23 to 25, 2018, in Baltimore, Maryland USA. The conference included three days of presentations on the capabilities of the Blaise Survey Processing System, new uses of Blaise to meet the high demands of computer assisted interviewing (CAI) survey environments, and future plans for system development. A pre-conference training on Blaise 5 capabilities took place on Monday, October 22, 2018.

The conference program was organized and planned by the Scientific Committee, chaired by Gina-Qian Cheung, University of Michigan, SRC. Other members of the committee included:

Rob Wallace - US Census Bureau, USA

Mark M. Pierzchala - MMP Survey Services, LLC, USA

Mike Hart - Office for National Statistics, UK

Éric Joyal - Statistics Canada

Leif Bochis Madsen - Statistics Denmark

Tim Carati - Statistics Netherlands

Jane Shepherd - Westat, USA

Westat, as the host organization for the conference, has collated and printed the proceedings for the benefit of conference participants and others.



# Table of Contents

## **Presentation Session 1 – Current Blaise 4 Uses and Moving to Blaise 5**

Migrating from Blaise 4 to Blaise 5 - The AHS experience .....	1
Using Blaise 5 for CAPI.....	21
Rolling Out Blaise 5 – An Interesting Journey .....	31

## **Presentation Session 2 – Accessibility and Testing**

Developing Accessible Blaise Surveys.....	43
Enhancing Blaise 4 Surveys for JAWS Screen Reader Compatibility.....	55
A Process for Blaise Data Emulation of Complex Instruments .....	69
An Approach to Unit Testing .....	79

## **Presentation Session 3 – Multimode and Survey Management**

Experiences with Blaise 5 CATI and Multimode at Statistics Norway .....	83
Implementing a Blaise 5 Mixed-Mode Solution.....	93
Blaise 5 with RTI’s Integrated Field Management System on Field Interviewer Laptops .....	101
Conducting Offline Blaise 5 Surveys in a Distributed Environment.....	111

## **Presentation Session 4**

Blaise 5 Multimode Management – A Report by the BCLUB Multimode Management Group.....	119
--	-----

## **Presentation Session 5 – Screen Design Part 1**

Some Uses of Roles in Blaise 5 .....	125
Web Screen Presentation Using Blaise 5.....	135
Developing Organization Level Screen Layout and Design Guidelines for Blaise 5.....	149
Using Blaise 5 in CAWI.....	163
Implementation of Blaise 5 Web Questionnaires into an Existing Business Survey Management System.....	171

## **Presentation Session 6**

Transitioning an Established Longitudinal Study to Blaise 5 and to a Mixed Mode Design..... 181

## **Presentation Session 7 – Screen Design Part 2 – Mobile**

Using Blaise 5 to Solve Multimode and Multi-Device Challenges .....243

Mobile Usability on Household Survey on Blaise 5.....253

## **Presentation Session 8 – Surveys, Data and Training**

Surveys with Sensors: The Future of Data Collection?.....259

Transforming Survey Paradata .....263

A Different Approach to Blaise Remarks .....289

ScriptWriter – The Compilation of Script Generation .....299

## **Presentation Session 9**

New Features in Blaise Colectica Questionnaires .....309

# Migrating from Blaise 4 to Blaise 5 – The AHS experience

*Roberto Picha, Richard J Squires Jr. - U.S. Census Bureau.*

## 1. Introduction

This paper documents the process of converting a complex U.S. Census Bureau Production Instrument developed in Blaise 4.8.4 Build 1861 (B4) to Blaise 5.3.0 Build 1517 (B5). The American Housing Survey (AHS) instrument was selected for this research. The instrument is currently in production and has a data collection mode of CAPI (Computer Assisted Personal Interviewing). For the conversion of this instrument into Blaise 5, the research team aimed to launch the instrument as a web survey. At the time we began this research, we did not have a Blaise server setup nor could our team see the results in a web browser due to the security restrictions imposed on our PC's. Therefore, the initial research was focused as CAPI mode with the intention to generate a web browser version as soon as the Blaise server was accessible.

## 2. Overview of Blaise 4 Instrument – AHS

The primary goal of the AHS survey is to provide a current and continuous series of data on selected housing and demographic characteristics. Policy analysts, program managers, budget analysts, and Congressional staff use AHS data to monitor supply and demand, as well as changes in housing conditions and costs, in order to assess housing needs. AHS interviews are conducted with respondents at single attached and detached units, multi-unit buildings, and manufactured/mobile homes. The 2015 AHS had a new sample drawn consisting of 115,398 cases. The sample contains national cases as well as cases in 15 metropolitan areas. There are 6,000 cases from the previous longitudinal sample, which was originally drawn in 1973, that were carried forward to 2015. These cases are part of a “bridge” sample, which will act as a control to measure the change in the data and survey redesign.

## 3. Conversion of AHS to Blaise 5

### 3.1 Goals of this research and expectations for the AHS in Blaise 5

The main goal of this research was to be able to launch an instrument in a browser mode. We chose to focus on Internet Explorer and Mozilla Firefox. Since we had already converted other instruments for research purposes, we took the same steps previously taken and applied them to this effort:

- After converting the instrument from B4, we compiled the instrument and identify errors now produced by B5 and correct them.
- Next, preview the instrument “as is” and identify the areas that needed attention, such as the kind of templates needed, correcting code that no longer works in B5, and find an alternative solution.

Another requirement given was minimizing the changes in the instrument; if possible, retain the structure of the instrument and the functionality for which it was built. This particular requirement was difficult to retain since B5 is a different system and the presentation in B4 may no longer be possible in B5; therefore, we had to be careful and opt for some new solutions or alternatives.

### 3.2 Blaise 4 to Blaise 5 Conversion process

The areas of conversion covered using the conversion tool in this process can be identified as follows

- **Datamodel** - The instrument itself along with all include files necessary to build the survey. In doing this, we preserved the folder structure currently used in our B4 development, this may change as we learn B5 best practices. This process created the final solution and the project.
- **Manipula Scripts** - Each script was converted, and each one created a solution and project. For AHS the following scripts converted were:
  - Setup script that allows us to load data from an input ASCII file into a BDBX file.
  - Standard scripts with our Case Management (CM).
    - CAPI trans, main script that dictates what to do with the case.
    - CAPI in, script that reads information from an ASCII file and updates the instrument.
    - CAPI out, script that writes information from the instrument to an ASCII File.
  - AHS did not have any customized Manipula/Maniplus, so these steps were excluded.
- **External files** that are used as Lookup table were converted separately as well, each one with a respective Solution and Project.

After this, the main solution (Inst) included all Manipula projects. In addition, the externals were added to the main solution, this was done after rebuilding the bdbx's (lookup table) inside the AHS instrument.

### 3.3 Using the Conversion Tool provided in Blaise 5

The Blaise tool converter was pretty straight forward, 99% of the code translated without much difficulty. The conversion Tool was used for the instrument/ External files (Datamodel), and Manipula scripts. However, this step did not provide a clean build after conversion, and can be summarized as follows:

- In the Data model:
  - **Warnings** - We experienced more than 300 warnings during the conversion process.
    - Not all field references of the Resource Database are mapped to Fields of the Datamodel.
    - The meaning of this expression differs from Blaise 4,  
The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value 0.  
The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value 0.  
The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value 0.
    - Self-Reference warnings.  
Self Reference: Field MMORT of Block Type BMortgages is put on route with generated parameter 'mmort.Mort\_intro.NUMMORTG' that contains an instance of the same type  
Self Reference: Field MMORT of Block Type BMortgages is put on route with generated parameter 'mmort.Mort\_intro.NUMMORTG' that contains an instance of the same type  
Block BBACK refers to an instance back of itself in 'back.SameApptFill' (Consider removing the prefix back.)
    - KEEP statements were generated
    - String constant exceeding assigned variables  
String Constant is too long for assigned variable : C:\AHS\_B5\includes\DateprocsAHS.prc.incx

- **Lots of layout removal** (B4 legacy layout). In earlier version of Blaise 5, a warning message was provided that pointed out that LAYOUT statements would eventually have to be removed. In later versions, it transitioned from a warning to an error, forcing us to go through dozens of modules to remove layout code before a clean compile could be achieved.
- **Adjustments to the external files used in B4**, that is rebuild them again.
- In the Manipula scripts:
  - While all the scripts were converted, the Setup script was the most important script to convert so that the input data could be loaded and generate the case databases. Some adjustments were made to this script in order to load the cases as in B4.
  - Early in this migration we received the error “*Element 'leftHandeSize' was not found. Line 1, position 1*” and the number 385917. It was temporarily commented out. These errors had to do with SET OF questions. Later versions of B5 corrected this issue and the code was reinstated.

### 3.4 Code Modifications in Blaise 5 (The AHS instrument)

Once migration was completed, it was obvious that we had to touch the code to correct some bugs and previous design issues used during B4 development.

#### 3.4.1 Problematic Blaise 4 Code

During the conversion process we discovered that even though we have coding standards in our group, these were not always followed by the original authors working on this project. Due to this B4 code, modifications were required with B5 code in order for the instrument to run like it did in B4. This is not specifically an issue with B5, but we do point this out since this had to be addressed in order to continue our research and to get the instrument to work under B5 as it did under B4. Main problematic areas in the B4 code were:

- **Redundancy in the code**, repetitive code through the instrument. The use of procedures to simplify the code is always encouraged, however not always applied.
- **Different styles in the code**, best programming practices had not been enforced. This resulted in convoluted code which was very hard to maintain and convert. Contributing to this is that the instrument used for our research is 14 years old and at least 6 programmers worked on this over the years, who each brought in a different level of programming best practices and Blaise expertise.
- **Deviate from Standards** - Instrument did not follow some of the standard collection practices used by many other surveys, disregarding a good solid structure of the instrument. For example, format of phone number or method to collect address information was done differently.
- **Data Processing for output within Instrument** - Many Sponsors have the tendency to request that the instrument massage data to help simplify their output process. Not only does the instrument collect data, it also calculates variables for output. This “data processing” part of the B4 instrument added complexity to the conversion process to create a B5 instrument that would perform exactly like the B4 instrument.
- **Parameterization** - Another item found during this process was the fact that there was no parameterization across the blocks in the AHS instrument; consequently, it was hard to dissect all

pieces for analysis during module testing, when issues were identified, such as fills not showing up or skipping variables.

- **Full Qualified References** – Too many full qualified references in the code or at times referencing variables from within block at different levels. There were times where certain information needed in a block was reaching a variable located at a deeper level, this caused some confusion and as much as possible were corrected for the proper functionality of the survey.

### 3.4.2 Coding Modification Suggestions when migrating to B5

As we modified the AHS instrument in B5 we used some best programming practices that align with the recommendations given by Blaise community. These will be passed along to our team when we move other project to B5:

Checks & Signals can be built and display differently in B5; still, B5 can compile the B4 syntax code. Furthermore, the message or text of the check & signals will render in the page in B5 contrary to the pop window they are rendered in B4. For example, our edit messages in B4 display in a pop-up window in Blue Text:

```
♦'No Prior Survey Interviewing Experience' cannot be selected with any other answer category. Please go back and correct your answer.
```

In B5, this edit message looks more like our B4 interviewer instruction since those are displayed in blue text near the question text. Therefore, we must make a small adjustment to the color to minimize confusion. For example, we may use red text to visually aid the interviewer and the reason for erring out.

```
[err 01] 'No Prior Survey Interviewing Experience' cannot be selected with any other answer category. Please go back and correct your answer.
```

Also, we believe that for simplicity, the Checks & Signal should have an area where they can be placed within the code, and therefore, remove from the rules section.

From this in the rules
<pre>IF (No_prior_survey_interv_experience IN PRIOR_SIPP) THEN   ERROR INVOLVING(PRIOR_SIPP)"&lt;newline&gt;&lt;Z&gt;s&lt;/Z&gt;&lt;L&gt;"No Prior Survey   Interviewing Experience' cannot be selected with any other answer category.   Please go back and correct your answer. &lt;/L&gt;" ENDIF</pre>
Add a Check Section and...
<pre>CHECKS CK_PRIOR_SIPP "&lt;/R&gt;[Err 01] 'No Prior Survey Interviewing Experience' cannot be selected with any other answer category. Please go back and correct your answer. &lt;/R&gt;" = NOT(No_prior_survey_interv_experience IN PRIOR_SIPP) INVOLVING (PRIOR_SIPP)</pre>
In the Rules :
<pre>IF PRIOR_SIPP.CARDINAL &gt; 1 THEN   CK_PRIOR_SIPP ENDIF</pre>

- The need of parametrization across blocks moving into B5, is important to isolate issues during modular testing. In addition, it adds simplicity for maintainability. Furthermore, we noticed that when “reaching” for variables across the instrument, the information needed is not always available right away.
- The use of functions and procedures are strongly encouraged, as their use simplifies the code for readability and maintainability.
- Decline any data output manipulation in the instruments within B5; especially if these create performance issue. However, these may be ok for small things, such as:
  - Setting a value based on a condition

```

IF blkNoint.BBEnd.BYOBS = Observation THEN
  BYOBS := 1
ELSEIF blkNoint.BBEnd.BYOBS = Information THEN
  BYOBS := 2
ELSE
  BYOBS := EMPTY
ENDIF

```

- However, when dealing with arrays or massive amount of variables this should be avoided.

### 3.4.3 Utilizing Blaise 5 Functions

The use of function in B5 is one of the nicest enhancements in the software and it is valued. While we praise the procedures in B4, B5 functions are easy, simple, and effective for the small things they are used for. Procedures are still useful, however, the functions return a simple single value. In the AHS, due to the redundancies of the code - specifically with formatting texts in different areas - these functions helped to reduce the clutter in various sections. Functions were also used for assignments and for formatting codes among other things (see Figure 3.4.1).

Figure 3.4.1 – Example of function created in Blaise 5

```

FUNCTION fGendertoStringFormat : STRING
  PARAMETERS IMPORT iGender : INTEGER
  RULES
  IF iGender = 1 THEN RESULT := 'Male'
  ELSEIF iGender = 2 THEN RESULT := 'Female'
  ELSE
    RESULT := "
  ENDIF
ENDFUNCTION

```

How it is used in the code:      aGenderFill := fGendertoStringFormat (Gender.ord)

### 3.5 LAYOUT Modification Required in Blaise 5

One of the areas requiring modifications during the conversion from B4 to B5 had to do with layout and moving away from Layout Statements within the B4 code, and moving it to the Blaise Resource Editor as the way to assign proper templates or modify certain controls for better view.

Our research effort identified the following major areas that we encountered along the way:

#### 3.5.1 Master template

We took the approach to use as much of the default Master Template that is provided with B5. There were a couple of areas that required adjustments in order to present the AHS screen under B5 similar to the way it is presented under Blaise 4. These areas include:

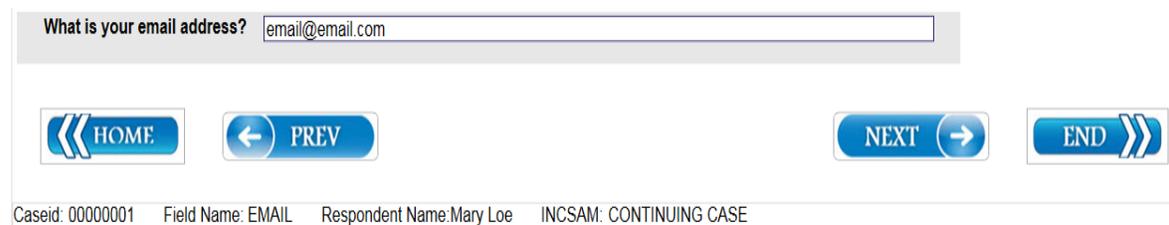
- **The Header Template** – We kept the header template simple, displaying three items: Our logo and the name of the survey, a dropdown list to switch languages, and the parallel tabs to jump at any given time from the normal interview (see Figure 3.5.1.1).

Figure 3.5.1.1 – Example of Header created for AHS Blaise 5



- **Buttons** - We experimented with different styles of buttons. We anticipate this is one area that will be addressed when we establish our Standard Instrument Master Template, in order to have a consistent look and feel for the instruments between projects (see Figure 3.5.1.2).
- **Bottom Status Bar** – Our Blaise 4 standards displays a status bar at the bottom of the screen. At a minimum, the Case ID (primary Key) and Field Name are contained in this area, along with project specific fields. We added this to the Master Template for consistency (see Figure 3.5.1.2).

Figure 3.5.1.2 – Example of Bottom Status Bar created for AHS Blaise 5



- **Master Template** - The final result of this Master Template is displayed below. It was decided to maintain questions using the entire real estate given in the screen. Any menu or navigation may be added in the future; however, the goal was to have a simple template where the focus was the question to be asked. This template allows the user to scroll vertically and it is not limited to a small number of questions as we originally intended (see Figure 3.5.1.3).

Figure 3.5.1.3 – Master Template example for AHS Blaise 5

American Housing Survey -- Version 7.38

United States Census Bureau American Housing Survey ENGLISH

AHS FAQ Back HHROS CR

(Last time) we recorded that your house used electricity most for cooking.

Is this information still correct?

Yes  
 No

? [F1]

(Last time) we recorded that your house had a refrigerator that is in working order.

Is this information still correct?

◆ Exclude ice boxes

Yes  
 No

◀ HOME    ← PREV    NEXT →    END ▶▶

Caseid: 00000001    Field Name: REFRV    Respondent Name: Mary Loe    INCSAM: CONTINUING CASE

### 3.5.2 Phone Template

Currently, in B5, phone numbers can be collected using a single string field.

- By adding a text role “EDITMASK” in the source code, it provides a hint to the user as to what the input element is used for, or the type of input that is required (see Figure 3.5.2.1).

Figure 3.5.2.1 – Example – Phone Masking for AHS Blaise 5

```

FIELDS
  TelNumber
  "What is your phonenumber?"
  <i>This field shows a regular string-field with a text role in the source code."
  EDITMASK "(999) 999-9999"
  : TPhoneNumber

RULES
  TelNumber
  
```

**What is your phonenumber?**  
*This field shows a regular string-field with a text role in the source code.*  
 ( ) -

However, in AHS and in B4, phone numbers are collected using separate fields or into a single field. Depending in the questions, extension or phone type were necessary, but not always: (Area+Prefix+Suffix+Extension+Phonetype).

Ideally, we wanted to incorporate something in B5 that would handle any scenario. To do this, we created field pane templates for each field mentioned above. Then, each template was individually linked to a mapped type reference in the instrument. For example, the area code template references a list type called “TPhoneArea”. When this Type is used in the instrument, the area code template is automatically assigned to that field.

In order to group all templates together on one screen, place the source code under the same GROUP name (example “grpPhoneMask”). This creates a table template that will contain a collection of all of the field pane templates to be grouped together. If a field in the group does not exist in the source code, the template will simply be ignored and all other templates will truncate.

Below is an example of the screen if all fields (Area+Prefix+Suffix+Extension+Phonetype) exist under the same group name.

- By adding a text role “WATERMARK” in the source code, it provides a hint to the user as to what the input element is used for, or the type of input that is required (see Figure 3.5.2.2).

Figure 3.5.2.2 – Example – Watermark example for Phone Masking

Source Code:

```

GROUP
grpPhoneMask "<newline><Z>s</Z><L> Please enter your phonenumber
              <newline> Current phone number: ^{Dial_Number}"

FIELDS
Phone_Area      "" WATERMARK "999"      : TPhoneArea
Phone_Prefix    "" WATERMARK "999"      : TPhonePrefix
Phone_Suffix    "" WATERMARK "9999"     : TPhoneSuffix
NOPHONEXT       "" WATERMARK "EXT"      : TPhoneExt
NOPHONETYP      "" WATERMARK "Phone Type" : TPhoneType

RULES
Phone_Area
Phone_Prefix
Phone_Suffix
NOPHONEXT
NOPHONETYP
ENDGROUP

```

Resulting



- The **WATERMARK** role provides a hint to the user as to what input elements are used for.

### 3.5.3 Mailing & Physical Address Templates

The AHS instrument collects Physical and Mailing address. In B4 they are collected in two blocks. The team created a template that collects Physical & Mailing address separately as in B4, both addresses are encapsulated into a group and both groups inside a main block.

Address did not have to be replicated in each address question but instead was displayed in the header of the group. This made presentation simple and easy to enter or correct data. Below a comparison between B5 and B4 (see Figure 3.5.3.1).

Figure 3.5.3.1 – Example – Blaise 5 vs. Blaise 4 address templates

Blaise 5 Address layout

♦ Correct the Address, if no correction needed simple press ENTER

Old address:  
105 BEACH RD  
ANY TOWN, CA 99997-9997

House Number.	<input type="text" value="105"/>
House Number Suffix.	<input type="text"/>
Street Name.	<input type="text" value="BEACH RD"/>
Street Name.	<input type="text"/>
Non-City Style Address.	<input type="text"/>
GQ Unit Description.	<input type="text"/>
Physical Description.	<input type="text"/>
City.	<input type="text" value="ANY TOWN"/>
State.	<input type="text" value="CA"/>
Zip Code.	<input type="text" value="99997"/>
Zip Code Ext.	<input type="text" value="9997"/>
Building Name.	<input type="text"/>

Blaise 4 Current Address collection

? [F1]

♦ Enter the correct House Number or press ENTER for Same.

Survey intro.	1	House number	184	Physical description	
Verify address	2	House no. suffix		Locality	ANY TOWN
		Street name	OCEAN VIEW	State	ND
		Unit designation		Zip code	99997-9997
		GQ unit info.		Building name	
		Non-city address			

The team created another template for the same purpose but this time the intention is to display both addresses on one screen and allow editing of the address that is not correct. We still needed to see the address that was correct as a show and gray background. The advantage of this template was to have both information side by side (see Figure 3.5.3.2).

Figure 3.5.3.2 – Another example of Blaise 5 address template

Mailing address House Number		
Physical Address	Labels	Mailing Address
12345	House Number	12345
Building 2	House No. Suffix	Building 2
my road	Street Name.	my road
2B	Unit Designation	2B
turn right out the elevator	GQ Unit Descrip.	turn right out the elevator
my_wonderful_area	City	my_wonderful_area
53	State	53
99999	Zip Code	99999
9999	Zip Code Ext.	9999

### 3.5.4 Tables and Groups Templates

Tables and Groups layouts go hand in hand. In B4, tables are used to take advantage of their two dimensional view, especially but not limited to demographic sections. B5 presents a different format and there were times that the code and new controls (dropdown) had to be adjusted to align with B5's new way to represent the two dimensional view. In particular, some adjustment was necessary in the code as seen below and naturally we used the default Table/Group layout from B5. Some changes were made for the final presentation.

The table below shows the implementation of a group from the original table in B4 (see Figure 3.5.4.1).

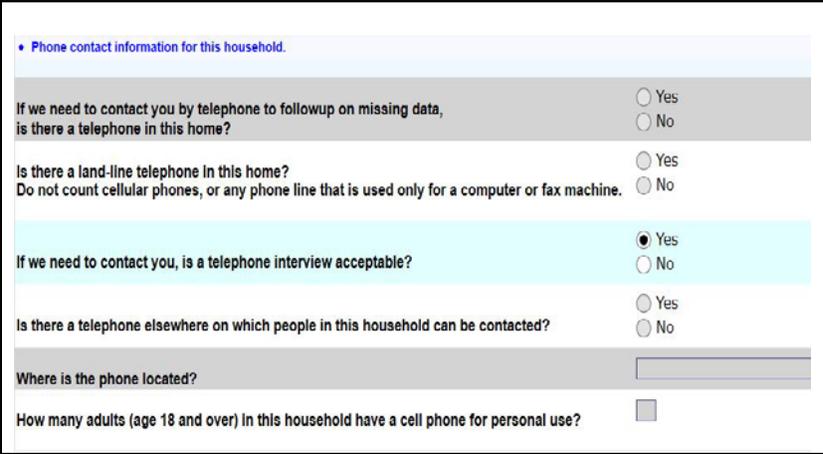
Figure 3.5.4.1 – Example of Blaise 4 template to Blaise 5 group

One way in B4	Another way in B4	New approach in B5
<b>Block</b> bRoster ... <b>Endblock</b>  <b>Table</b> tblRoster Roster: array [] of BRoster <b>EndTable</b>	<b>Table</b> tblRoster <b>Block</b> bRoster ... <b>Endblock</b>  Roster: array [] of BRoster <b>EndTable</b>	<b>Block</b> bRoster ... <b>Endblock</b>  <b>Group</b> grpRoster Roster: array [] of BRoster <b>Endgroup</b>

The team created a few templates that were enhanced from the default templates that B5 provides. These templates were used in Groups. For the AHS, we grouped questions whenever answers choices were similar amongst them. This simplified the presentation and the flow of the questions; however, there was an issue encountered that it is worth mentioning. The rules inside our tables/groups template do not always work as expected, i.e., all question were on route when they should not have been. The use of “critical items” and client rules to circumvent the issue failed and neither option was valid. As a result, a workaround was necessary using the rules and code as shown below (see Figure 3.5.4.2).

Figure 3.5.4.2 – Example of templates created under Blaise 5

B5 Code Without Changes	Result is All questions on route (Only one question should be)
<pre> IF IPHONENUM = EMPTY THEN   TELHH.ASK ENDIF IF TELHH = Yes THEN   TELAND.ASK ENDIF IF TELHH = Yes OR IPHONENUM &lt;&gt; EMPTY THEN   TELIN.ASK ENDIF IF TELHH = NO OR TELHH = NONRESPONSE THEN   TELAV.ASK ENDIF ... more code           </pre>	<p>Phone contact information for this household.</p> <p>If we need to contact you by telephone to followup on missing data, is there a telephone in this home? <input type="radio"/> Yes <input type="radio"/> No</p> <p>Is there a land-line telephone in this home? <input checked="" type="radio"/> Yes <input type="radio"/> No</p> <p>Do not count cellular phones, or any phone line that is used only for a computer or fax machine.</p> <p>If we need to contact you, is a telephone interview acceptable? <input checked="" type="radio"/> Yes <input type="radio"/> No</p> <p>Is there a telephone elsewhere on which people in this household can be contacted? <input checked="" type="radio"/> Yes <input type="radio"/> No</p> <p>Where is the phone located? <input type="text" value="basement"/></p> <p>How many adults (age 18 and over) in this household have a cell phone for personal use? <input type="text" value="1"/></p>

B5 Code With Changes	Results only One question on route Note: TELIN is the only question that should be asked in the group of questions.
<pre> IF IPHONENUM = EMPTY THEN   TELHH.ASK ELSE   TELHH.SHOW ENDIF IF TELHH = Yes THEN   TELAND.ASK ELSE   TELAND.SHOW ENDIF IF TELHH = Yes OR IPHONENUM &lt;&gt; EMPTY THEN   TELIN.ASK ELSE   TELIN.SHOW ENDIF IF TELHH = NO OR TELHH = NONRESPONSE THEN   TELAV.ASK ELSE   TELAV.SHOW ENDIF ... more code </pre>	 <p>• Phone contact information for this household.</p> <p>If we need to contact you by telephone to followup on missing data, is there a telephone in this home? <input type="radio"/> Yes <input type="radio"/> No</p> <p>Is there a land-line telephone in this home? <input type="radio"/> Yes Do not count cellular phones, or any phone line that is used only for a computer or fax machine. <input type="radio"/> No</p> <p>If we need to contact you, is a telephone interview acceptable? <input checked="" type="radio"/> Yes <input type="radio"/> No</p> <p>Is there a telephone elsewhere on which people in this household can be contacted? <input type="radio"/> Yes <input type="radio"/> No</p> <p>Where is the phone located? <input type="text"/></p> <p>How many adults (age 18 and over) in this household have a cell phone for personal use? <input type="text"/></p>

### 3.5.5 Buttons vs Radio buttons

The use of these controls as answer lists are very simple, for PC's, the regular radio button does fine. For a web browser the buttons seem to be an ideal option. When implementing the buttons, we noticed that the rendering of the buttons inside the group table does not display as expected as they are shown vertically instead of horizontally. (see Figure 3.5.5.1).

Figure 3.5.5.1 –Example of buttons inside group table

This is how the radio buttons look in a group table layout for PC's.
<p>• Phone contact information for this household.</p> <p>If we need to contact you by telephone to followup on missing data, is there a telephone in this home? <input type="radio"/> Yes <input type="radio"/> No</p>
This is how the buttons look in a group table layout in a browser.
<p>• Phone contact information for this household.</p> <p>If we need to contact you by telephone to followup on missing data, is there a telephone in this home? <input type="button" value="Yes"/> <input type="button" value="No"/></p>

### 3.5.6 Dropdown list

The dropdown list control was selected for use inside the demographic tables to show answer lists containing more than two options. By presenting answer choices this way, the two dimensional presentation remains in the new AHS B5 instrument. This is demonstrated in the picture below (see Figure 3.5.6.1).

Figure 3.5.6.1 – Dropdown table

The screenshot shows the 'American Housing Survey -- Version 7.38' interface. At the top, there are navigation links: AHS, FAQ, Back, HHROS, and CR. Below this is a table with columns: Ino, Name, SEX, Relationship, Ver Age, BirthM, BirthD, and BirthY. The first row shows 'John Doe' as Female with a 'Select a value' dropdown for Relationship. The second row shows 'Jane Doe' as Male with a 'Select a value' dropdown for Relationship. A dropdown menu is open for the Relationship field, listing options: Same-sex husband/wife/spouse, Same-sex unmarried partner, Biological son or daughter, Adopted son or daughter, Stepson or stepdaughter, Grandchild, Father or mother, Brother or sister, Parent-in-law, Son-in-law or daughter-in-law, Other relative, Foster child, and Housemate/roommate. Navigation buttons include HOME, PREV, NEXT, and END. At the bottom, it shows Caseid: 00000001, Field Name: RRP, and Respondent Name: John Doe.

### 3.5.7 Dropdown list for SET OF

Unfortunately, the dropdown list does not present a real solution when being used in a two dimensional presentation that requires using a “SET OF” (see Figure 3.5.7.1). Currently B5 does not provide a template that can handle multiple answers SET OF in a dropdown. In this particular case, we have only 6 multiple options, but there are other questions with more than 10 choices. For purpose of continuing with the research we opted to leave “as is” and revisit at a later time for an alternative.

Figure 3.5.7.1 – Example Dropdown for SET OF

The screenshot shows the 'American Housing Survey -- Version 7.38' interface. At the top, there are navigation links: AHS, FAQ, Back, HHROS, and CR. Below this is a table with columns: Ino, Name, Sex, REL, AGE, Origin, and Race. The first row shows 'John Doe' as Female with a 'Select a value' dropdown for REL. The second row shows 'Jane Doe' as Male with 'Son or daughter' for REL and a 'Select a value' dropdown for AGE. A dropdown menu is open for the Race field, listing options: White, Black or African American, American Indian or Alaska Native, Asian, Native Hawaiian or other Pacific Islander, and Other; Specify. Navigation buttons include HOME, PREV, NEXT, and END. At the bottom, it shows Caseid: 00000001, Field Name: RRP, and Respondent Name: John Doe.

## 4. Challenges Encountered

### 4.1 Manipula Issues

Early versions of Manipula B5 created some issues during the migration; however, these were corrected in the version 5.3.5 Build 1472.

- **Did not Allow for a SET OF values** - This was needed to load variables with multiple answers from an input file. As an alternative, we attempted to use GETVALUE and PUTVALUE; however, these function did not work in full as well.

We also did not work with our main scripts. They seemed to work without the EDIT function, which was not yet available. The other minor issue was mentioned earlier. In essence, each build or version of B5 is coming along and translatable in full (version 5.3.0.1517).

### 4.2 Blaise 5 Bugs

Throughout the conversion process, multiple B5 updates were received. Many issues were resolved by the updates, and on occasion, additional bugs/issues were created. This added complexity to the research effort, since we needed to evaluate if the issue was created due to programming, or was an issue with B5 itself. A number of times we submitted questions and examples to the B5 development team, in which they were helpful in either confirming a B5 bug, or providing programming guidance.

### 4.3 Blaise 5 and In-House App

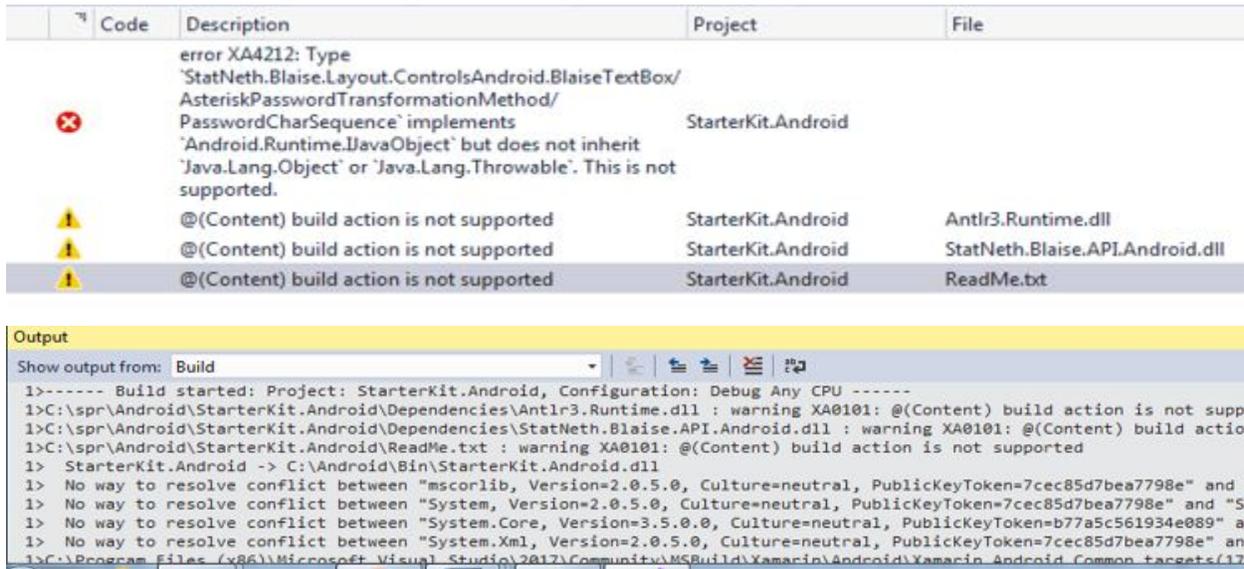
The research team spent some time looking into the Blaise 5 Sample App. The purpose was to examine the Android “Starterkit.Android” project and explore its functionalities. We intended to use lessons learned to develop apps to potentially access our future Blaise 5 Surveys-Blaise 5 data entry.

At the time of the research, we had Microsoft Visual Studio 2015 and Blaise 5 version 5.3.0.1472 installed in desktop; Microsoft Visual Studio 2017 community and Blaise 5 version 5.3.0.1472 on a Windows 7 laptop. We opened project Visual Studio 2015, copied the dependencies into the built folder, downloaded the specified packages, and conducted a static analysis of the Android project. We looked through each folder and explored the codes associated with each file in the folder: Activities, Adapters, Resources, and so on. We searched for references of Blaise Data entry API and identified where and how they are being used in the project code. When it comes to running or building the Blaise 5 sample project, we encountered multiple issues.

- In the Visual Studio 2015, we were unable to build the project. The first major challenge we encountered involved the Android SDK. We found that the Xamarin Android SDK Manager is not compatible with Visual Studio 2015 and when attempting to install the SDK Manager tools provided by Google, we had a licensing and other restrictions on installing third party software on the current development desktop.

Independently, we tried using the Visual studio 2017 community, we were able to build the solution, however we still encountered some issues. The picture below shows the error messages while building the project (see Figure 4.3.1).

Figure 4.3.1 – Error message while building project



The errors above seem to be related to an incompatible Blaise API “statNet.Blaise.API.Adroid.dll” and the “schemas.android.com” link referenced in some of “.axml” files in the resources/layout folder. An attempt to debug the errors, we revealed that:

- The reference of “schemas.android.com” link in the “xx.axml” file in layout folder in Resources folder is no longer available.
- Antlr3.Runtime.dll is not supported.
- StatNeth.Blaise.API.Android.dll in the dependencies folder is no longer supported. A much deeper search revealed that the dll did exist in the Blaise 5 bin folder under c:\program files (x86)\StatNet folder. However, we noticed a new API library "statNet.Blaise.API.AdroidLib.dll" in the bin folder; we do not know if it is replacement for the "statNet.Blaise.API.Adroid.dll" referenced startkit.Android project.

We hope to continue experimenting with the Blaise 5 app.

#### 4.4 Exporting the Audit Trail Database to XML

Another item researched during this conversion was how to retrieve and use the audit trail.

Our recommended SQL statement:

```

SELECT a.InstrumentId,a.SessionId,a.KeyValue as CaseId, b.TimeStamp, b.Content
FROM AuditSessionData a
LEFT JOIN eventdata b
ON a.instrumentId=b.instrumentId and a.sessionId=b.sessionId
WHERE [criteria list]
  
```

In the Audit Trail Setting, select "Answer" level to provide the level of granularity that includes the values entered by the user.

When one looks at the "Content" field where data is entered in XML format, there are UNBALANCED QUOTES, e.g.;

```
<UpdatePageEvent LayoutSetName=FRLaptop" PageIndex="5" />
```

Unbalanced quotes will create problems for XML parsers because this is NOT well-formed XML format. And the value for the attribute in this example:

The proper format for the attribute LayoutSetName should be:

- **LayoutSetName="FRLaptop"** and not
- **LayoutSetName=FRLaptop"** (a quote should be before the "F" in FRLaptop)

Note: All the entries appear to have this unbalanced quotes problem!

Since the data is stored in a SQL Lite database, how is the data secured? The connection strings for SQL Lite (per "https://www.connectionstrings.com/sqlite/") shows only:

```
Data Source=c:\mydb.db;Version=3;Password=myPassword;
```

Note the UserId is not part of the connection string, which is the usual format in other DB connection strings. Do we have any other security options available that can be implemented?

Would it be ideal and useable to have the Blaise utility to export the SQL Lite data according to the SQL statement recommended above? The team had to copy-paste the results from SQL query into a spreadsheet.

#### 4.5 Dealing with Alien Routers and Services

Time was spent to observe and test the following two Alien Router LockSection and LockSection2:

- Checking the HPServices using a .NET solution.  
[//In the "HPServices" .NET solution, this works fine.](#)  
PROCEDURE LockSection  
PARAMETERS  
IMPORT value: INTEGER  
EXPORT lockIt: INTEGER  
ALIEN("http://localhost:8733/Design\_Time\_Addresses/WCF/LockSectionService", 'LockSection', 'LockSection')  
ENDPROCEDURE

Enabling this procedure code in the bla and calling it in the RULES will not cause an exception, but neither will it produce output as expected.

- **Function Overload** - The service has a function overload "decimal LockSection(decimal importValue)" and "decimal LockSection(decimal importValOne, decimal importValueTwo)". Note that both have different attribute names, the first overload is "[OperationContract(Name="LockSection")] while the second one is "[OperationContract(Name = "LockSection\_Two")]".

Blaise either doesn't call it (because it's a function overload), or it can't process the service's return value. This is not a problem in .NET consumers though.

```
PROCEDURE LockSection_Two
PARAMETERS
  IMPORT value1: INTEGER
  IMPORT value2: INTEGER
  EXPORT lockIt: INTEGER
  ALIEN('http://localhost:8733/Design_Time_Addresses/WCF/LockSectionService', 'ILockSection', 'LockSection_Two')
ENDPROCEDURE
```

- Preliminary info: the C# code has 2 functions
  - "public decimal LockSection(decimal importValue)" and
  - "LockSection(decimal importValOne, decimal importValueTwo)".

These are overloaded functions, differing only in the parameter signatures. In the C# code for the Service, each of the overloaded functions have the proper attribute declarations in the Interface ("ILockSection"):

```
public interface ILockSection
{
    [OperationContract(Name="LockSection")]
    decimal LockSection(decimal importValue);

    [OperationContract(Name = "LockSection_Two")]
    decimal LockSection(decimal importValOne, decimal importValueTwo);
}
```

Alien Router #1 calls the overloaded function that has been decorated with the attribute value "[OperationContract(Name="LockSection")]". The attribute of the native C# function matches that of AlienRouter #1's 3rd argument.

The same is true for Alien Router #2 with its corresponding call to the overloaded function decorated with the attribute value "[OperationContract(Name = "LockSection\_Two")]".

- Alien Router #1 works fine, but Alien Router #2 does not. Does Blaise 5 have a problem with overloaded functions? A C# test project for the Service functions as expected and can handle the overloaded functions, so this doesn't make sense.
- NOTE: The Blaise 5 IDE also shows a blue squiggly line underneath the "LockSection\_Two" in the line "PROCEDURE LockSection\_Two". We are unsure what does this blue line mean.

```
PROCEDURE LockSection_Two
PARAMETERS
  IMPORT value1: INTEGER
  IMPORT value2: INTEGER
  EXPORT lockIt: INTEGER
  ALIEN('http://localhost:8733/Design_Time_Addresses/WCF/LockSectionService', 'ILockSection', 'LockSection_Two')
ENDPROCEDURE
```

## 5. Next Steps / Future Plans

- Full conversion of the instrument and be able to use as web survey.
- Upgrade to the full version of Microsoft Visual Studio 2017 and use the sample Android app project to develop our own app to access the survey.
- Initiate the use of apps to hit the server and launch the instrument
- Begin to create a process for data output including the audit trail
- Add more smart templates
- Work with sponsors and key stakeholders in developing master templates and identify similar components that can be shared between surveys.

## 6. Conclusion

### 6.1 Achievements

As a result of this conversion effort, we achieved the following:

- Increased our technical knowledge and capability of developing/converting instruments in B5.
- Gained a better understanding of the level of effort required to convert an existing B4 instrument over to B5.
- Enhanced our familiarity with the data resources, especially creating layouts that can provide a better presentation for our users.
- Expanded our understanding of requirements needed for setting up a project in a server environment and classifying roles within surveys.

### 6.2 Blaise 5 Learning Curve

There was a big learning curve for most authors in the research team, especially in understanding the concept of layouts using the Blaise 5 Resource Editor. This was partially due to the fact that the research team could not solely focus on Blaise 5 as they had other production priorities in Blaise 4. However, the team was able to make progress in defining a corporate layout by studying the examples provided by the software and reading documentation of previous research attempts by other members from our team. In summary, the lack of a strong understanding of how to create templates, prevented us from developing a rapid solution that could be easily used or applied to the AHS instrument.

### 6.3 Convert or Rebuild from Scratch?

During this process we spent a lot of time fixing/modifying/enhancing the Blaise 4 code that was converted to Blaise 5. Whether this was the consequence of moving to Blaise 5, non-ideal coding of the Blaise 4 instrument, or just new and different ways we wish to present our questions in Blaise 5, there is a lot of work required when a survey moves from Blaise 4 to Blaise 5. One must determine if it is better, in the long run, to rebuild the instrument from scratch in Blaise 5 or convert the Blaise 4 instrument code and modify as necessary. The answer will not always be clear, especially with older instruments that have been maintained by many different programmers.

## **7. Acknowledgements**

The authors of this paper would like to acknowledge the work and knowledge of Erin Slyne, Jose Vela, Michael A Johnson, Chris Borillo and Mecene Desormice for their assistance with this research project. Their contributions were important in this endeavor, and much appreciated.

*The views expressed in this paper are those of the authors and not necessarily those of the U.S. Census Bureau.*



# Using Blaise 5 for CAPI

*Rick Dulaney and G J Boris Allan, Westat*

Blaise 5 has been engineered for data collection on the web (CAWI), reflecting the general trend in survey research. However, many organizations have deep, ongoing investments in CAPI fieldwork using Blaise for data collection. These organizations have made a human investment by training hundreds or thousands of field data collectors to use Blaise applications, and they may additionally have several Blaise data collection efforts underway on specific projects.

For example, in one study, there were two distinct requirements:

- A smaller sample was to be questioned by interviewers supplied by the client. These very experienced interviewers were trained in the use of the keyboard. Keyboard CAPI was a client requirement for this sample.
- A much larger sample consisted of respondents who answered the same set of questions by themselves via the web, using web browsers or (in some circumstances) mobile devices.

The CAPI sample used Blaise 4.8 and the CAWI sample used a web interviewing tool. Blaise 4.8 and the web tool used different types of database.

## 1. CHALLENGE: USE OF KEYBOARD IN CAPI

One key challenge facing long-term projects currently using Blaise 4 for CAPI data collection is how to address the use of the keyboard. The default Blaise 5 screen presentation relies on a pointing device: generally interviewers or respondents will use a mouse or similar device, or will touch the screen with finger or stylus in order to interact with the Blaise 5 application. Organizations may spend significant time developing the look and placement of the Next and Back buttons so interviewers and respondents can locate and use them more naturally. However, the field data collectors on CAPI projects have generally been trained to use the keyboard to record answers and move to the next screen. This is done for several reasons:

- Field data collectors on CAPI projects have generally been trained to use the keyboard to record answers and move to the next screen.
- Many field data collectors have extensive experience with keyboard-based systems
- Pointing tools may change from one laptop or tablet to another, but keyboard look and layout remain fairly standard
- Field data collectors are generally hired for their interpersonal skills and ability to collect data of the highest possible quality, rather for than their computer skills
- There is a strong sense that the higher efficiency of keyboard data collections leads to greater effectiveness, by allowing field data collectors to focus on a quality interaction with the respondent.
- There is a strong sense that keyboard data collection is more efficient for field data collectors. For example: pressing “1” and “Enter” three times on successive screens is simpler than locating and pressing the Next button on three successive screens.
- There is a strong sense that the higher efficiency of keyboard data collections leads to greater effectiveness, by allowing field data collectors to focus on a quality interaction with the respondent.

Organizations or projects deciding which version of Blaise to use for large field projects have a choice. If they use Blaise 5 and accept the default Blaise 5 screen look and feel, they will need to retrain their staff to use the new, web-oriented screens and move away from established practice. If they choose to stay with Blaise 4.8, they can preserve their investment in keyboard data collection, but they cannot move forward with new features and techniques in Blaise.

However, this does not mean that the choice is between CAPI (Blaise 4.8) and CAWI (Blaise 5) because Blaise 5 is flexible enough to accommodate both modes on the same survey. The flexibility comes courtesy of the Blaise 5 resource database (the blrd file) and other layout tools (such as the layout editor in the control centre).

It is possible to configure Blaise 5 to approximate the CAPI look and feel of Blaise 4.8 (which is what interviewers often know best). The Blaise 5 resource database has example templates that can be modified to replicate the CAPI look-and-feel until a time that experienced interviewers can embrace new approaches.

Hopefully, interviewers will soon be able to progress with the CAWI interface.

## **2. ONE SOLUTION: CONFIGURE BLAISE 5 TO RESEMBLE BLAISE 4 CAPI**

However, it is possible to configure Blaise 5 to approximate the CAPI look and feel of Blaise 4.

- For a categorical question, the key elements not present in Blaise 5 by default are:
  - Each category needs an explicit number, usually starting with 1, such as “1. YES”.
  - The screen needs an input field to record the response, rather than pressing a radio button
- The behavior of the input field and radio buttons should mimic Blaise 4:
  - when the user enters a “1” in the input field, the system should show option 1 as selected;
  - when the user selects a category using the radio button, the value “1” should replace whatever is currently in the input field.

To demonstrate how various question types might look in a Blaise 5 CAPI application that retains the ability to use the keyboard, we developed an example application – CapiCawi – for a variety of question types. DK/RF responses are triggered by the F5/F6 keys which are defined in the resource database.

- CapiCawi uses several questions from the Commercial Building Energy Consumption Study (CBECS), chosen to represent different question types.
- CBECS collects energy consumption and expenditures data from US commercial buildings.

To allow us to move between CAPI and web modes, for the same question, CAPI and CAWI buttons are on every screen in CapiCawi. When using CapiCawi, even in CAPI mode, it is easier to select the CAWI button using a pointer than trying to navigate using the keyboard.

### 3. AN EXAMPLE SCREEN

We will give some CAWI and CAPI screens to compare functionality, remembering that in CapiCawi the Blaise 5 codebase is the same. We start with a Yes/No question.

#### Yes/No question in CAWI



The screenshot shows a web-based survey interface. At the top, there is a purple header with the Westat logo and the text "Comparing CAPI and CAWI interfaces - CapiCawi". Below the header, the word "CAWI" is displayed in the top left corner, and "A20 - Escalators" is in the top right. The main content area contains the question "Are there any escalators in this building?" followed by two radio button options: "Yes" (unselected) and "No" (selected). At the bottom of the screen, there is a navigation bar with five buttons: "Next" (with a right-pointing arrow), "Back" (with a left-pointing arrow), "CAPI", "CAWI", and "Save and Continue Later".

The CAWI version has little information on the screen: just the question, responses and navigation buttons.

## Yes/No question in CAPI

The screenshot shows the CAPI interface for the question "Are there any escalators in this building?". At the top, there is a purple header with the Westat logo and the text "Comparing CAPI and CAWI interfaces - CapiCawi". Below the header, the question is displayed. There are two radio buttons: "1 Yes" (unselected) and "2 No" (selected). Below the question, there is a "Select a value" section with four input fields: "Floor-to-ceiling height" (with a placeholder "Enter a number" and a "DontKnow" indicator), "Elevators" (with the value "1"), "Number of elevators" (with the value "1"), and "Escalators" (with the value "2"). At the bottom, there are navigation buttons: "Next" (blue arrow), "CAWI" (black button), "CAWI" (black button), "Back" (blue arrow), and "Save and Continue Later" (blue button).

In the CAPI mode there is a radio button and a number next to each response, an instruction at the bottom of the “Info” pane, and an input field with label in the “Answer” pane.

The interviewer can see answers to other questions.

IN CAPI, the Info and Answer panes were simulated using settings in the resource database to resemble these screen areas in Blaise 4.

If the user presses a radio button, the corresponding response category number appears in the input field at the bottom.

Conversely, when the user enters a value in the input field, the appropriate radio button becomes selected.

Using the F5/F6 function keys toggles a DontKnow/Refusal indicator to the right of the question answer in CAPI mode. The treatment of special answers (DK/RF, etc) in CAWI is usually by means of buttons or check boxes.

## 4. FURTHER SCREENS

The following screens show the same question in both modes, where the differences are controlled by the resource database and layout tools in the control centre.

## Categorical question in CAWI

Westat Comparing CAPI and CAWI interfaces - CapiCawi A7 - Square feet category

CAWI

We understand that it may be difficult to give an exact figure for square footage. Please select which category best describes the total gross square footage in this building. There are examples provided to help you estimate.

- 1,000 square feet or less  
*(1,000 square feet is approximately 2 times the size of a two car garage)*
- 1,001 to 5,000 square feet  
*(Example: fast food restaurant)*
- 5,001 to 10,000 square feet  
*(Example: sit-down style chain restaurant)*
- 10,001 to 25,000 square feet  
*(Example: one or two screen movie theater)*
- 25,001 to 50,000 square feet  
*(Example: supermarket)*
- 50,001 to 100,000 square feet  
*(Example: large discount or home improvement store)*
- 100,001 to 200,000 square feet  
*(Example: 3-level department store)*
- 200,001 to 500,000 square feet  
*(Example: professional basketball arena)*
- 500,001 to 1 million square feet  
*(Example: convention center)*
- Over 1 million square feet  
*(Example: skyscraper)*

Next CAPI CAWI

Back Save and Continue Later

This is a familiar style of Blaise 5 screen.

## Categorical question in CAPI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

We understand that it may be difficult to give an exact figure for square footage. Please select which category best describes the total gross square footage in this building. There are examples provided to help you estimate.

<input type="radio"/> 1 1,000 square feet or less <i>(1,000 square feet is approximately 2 times the size of a two car garage)</i>	<input type="radio"/> 6 50,001 to 100,000 square feet <i>(Example: large discount or home improvement store)</i>
<input type="radio"/> 2 1,001 to 5,000 square feet <i>(Example: fast food restaurant)</i>	<input type="radio"/> 7 100,001 to 200,000 square feet <i>(Example: 3-level department store)</i>
<input checked="" type="radio"/> 3 5,001 to 10,000 square feet <i>(Example: sit-down style chain restaurant)</i>	<input type="radio"/> 8 200,001 to 500,000 square feet <i>(Example: professional basketball arena)</i>
<input type="radio"/> 4 10,001 to 25,000 square feet <i>(Example: one or two screen movie theater)</i>	<input type="radio"/> 9 500,001 to 1 million square feet <i>(Example: convention center)</i>
<input type="radio"/> 5 25,001 to 50,000 square feet <i>(Example: supermarket)</i>	<input type="radio"/> 10 Over 1 million square feet <i>(Example: skyscraper)</i>

Select a value

Square feet category

Next CAPI CAWI

Back Save and Continue Later

There are many categories, arranged in 2 columns.

If the interviewer types in a number outside the code limits, an automatic message is generated:

Select a value

**Square feet category**

Enter a valid code

### Categorical question with images in CAWI

The screenshot shows a CAWI interface titled "Comparing CAPI and CAWI interfaces - CapiCawi" with a sub-header "A13 - Building shape". The question asks the user to select the shape that most resembles a building's footprint. The interface includes a "Next" button, a "Back" button, and a "Save and Continue Later" button. The question is presented in a CAWI format with visual icons for each shape option.

CAWI

Please select which of these shapes most resembles the floorplan of this building at ground level. This is sometimes called the "footprint" of the building.

Square

Wide rectangle

Narrow rectangle

Rectangle or square with an interior courtyard

"H" shaped

"U" shaped

"E" shaped

"T" shaped

"L" shaped

"+" or cross shaped

Other shape

Next

Back

CAPI

CAWI

Save and Continue Later

In face-to-face interviews we can use "show cards" but in web interviews we have to show images.

## Categorical question with images in CAPI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

Please select which of these shapes most resembles the floorplan of this building at ground level. This is sometimes called the "footprint" of the building.

1 Square

2 Wide rectangle

3 Narrow rectangle

4 Rectangle or square with an interior courtyard

5 "H" shaped

6 "U" shaped

7 "E" shaped

8 "T" shaped

9 "L" shaped

10 "+" or cross shaped

11 Other shape

Select a value

Roof tilt

Building shape

Percent exterior glass

Next CAPI CAWI

Back Save and Continue Later

If we use show cards, we can still have visual reminders of what is on a card, so that it is easier for interviewers to interpret content.<sup>1</sup>

The three questions listed in the input area are all image questions.

## Select-all question in CAWI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

CAWI

A27 - What renovations

Please select which types of renovations have been done since 2000.

Select all that apply.

Cosmetic improvements (interior or exterior)  
*(Examples: new paint, siding, furniture, wallpaper, carpeting)*

Addition or annex

Reduction of enclosed floorspace  
*(Example: demolition of unused wing of building)*

Interior wall re-configuration  
*(Example: individual offices turned into area for cubicles)*

Roof replacement

Window replacement

HVAC equipment upgrade  
*(includes hot water heaters)*

Lighting upgrade

Plumbing system upgrade

Electrical upgrade

Insulation upgrade

Fire, safety, or security upgrade

Structural upgrade  
*(Examples: foundation upgrade, seismic upgrade)*

Other. Please describe the other type of renovation.

Next CAPI CAWI

Back Save and Continue Later

Select-all (code-all-that-apply, or SET) are standard questions in CAWI, as shown here.

<sup>1</sup> See "Some uses of Roles in Blaise 5" (G J Boris Allan, IBUC 2018) on changing question texts to reflect differences between interviewer-administered and self-administered questionnaires.

## Select-all question in CAPI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

Please select which types of renovations have been done since 2000.

Select all that apply.

<input checked="" type="checkbox"/> 1 Cosmetic improvements (interior or exterior) <small>(Examples: new paint, siding, furniture, wallpaper, carpeting)</small>	<input type="checkbox"/> 8 Lighting upgrade
<input type="checkbox"/> 2 Addition or annex	<input checked="" type="checkbox"/> 9 Plumbing system upgrade
<input type="checkbox"/> 3 Reduction of enclosed floorspace <small>(Example: demolition of unused wing of building)</small>	<input type="checkbox"/> 10 Electrical upgrade
<input type="checkbox"/> 4 Interior wall re-configuration <small>(Example: individual offices turned into area for cubicles)</small>	<input type="checkbox"/> 11 Insulation upgrade
<input type="checkbox"/> 5 Roof replacement	<input checked="" type="checkbox"/> 12 Fire, safety, or security upgrade
<input type="checkbox"/> 6 Window replacement	<input type="checkbox"/> 13 Structural upgrade <small>(Examples: foundation upgrade, seismic upgrade)</small>
<input type="checkbox"/> 7 HVAC equipment upgrade <small>(Includes hot water heaters)</small>	<input type="checkbox"/> 14 Other. Please describe the other type of renovation.

Select all values that apply

What renovations

Next CAPI CAWI

Back Save and Continue Later

Notice the standard Blaise 4.8 notation for multiple selections from a select-all list of categories “1-9-12”.

## Continuous number question with explicit DK in CAWI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

CAWI A6 - Square feet

What is the gross or total square footage of all the space in this building both finished and unfinished, including basements, hallways, lobbies, stairways, elevator shafts, and indoor parking levels?

square feet

I don't know

Next CAPI CAWI

Back Save and Continue Later

This shows a standard number text box, with a “units” specifier, and an explicit DK. Normally all non-response values are hidden when a question appears for the first time, or it has a response value. The requirement is that, for this question, the DK is always shown.

## Continuous number question with DK in CAPI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

CAPI

What is the gross or total square footage of all the space in this building both finished and unfinished, including basements, hallways, lobbies, stairways, elevator shafts, and indoor parking levels?

Enter a numeric value between 1 and 999999999.

Square feet  square feet DontKnow

Next CAPI CAWI

Back Save and Continue Later

With CAPI we have an interviewer who can probe, and so there is no need to show the explicit DK. If the answer is truly DK, then we need some way to show this clearly. Using the F5 function key (project-dependent) shows a clear indication of “DontKnow”.

## A large enumeration question as a drop-down list in CAWI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

AS : Zip code

CAWI

State:

- FL - FLORIDA
- FL - FLORIDA
- GA - GEORGIA
- HI - HAWAII
- ID - IDAHO
- IL - ILLINOIS
- IN - INDIANA
- IA - IOWA
- KS - KANSAS
- KY - KENTUCKY
- LA - LOUISIANA
- ME - MAINE
- MD - MARYLAND

Next CAPI CAWI

Back Save and Continue Later

Collecting state information can be performed in many ways. It was decided that, for the envisaged web audience, selecting from a list of states was best accomplished using the DropDownList template for an enumerated type.

## A large enumeration question as a category list in CAPI

Westat Comparing CAPI and CAWI interfaces - CapiCawi

State:

<input type="radio"/> 1 AL - ALABAMA	<input type="radio"/> 14 IL - ILLINOIS	<input type="radio"/> 27 MT - MONTANA	<input type="radio"/> 40 RI - RHODE ISLAND
<input type="radio"/> 2 AK - ALASKA	<input type="radio"/> 15 IN - INDIANA	<input type="radio"/> 28 NE - NEBRASKA	<input type="radio"/> 41 SC - SOUTH CAROLINA
<input type="radio"/> 3 AZ - ARIZONA	<input type="radio"/> 16 IA - IOWA	<input type="radio"/> 29 NH - NEW HAMPSHIRE	<input type="radio"/> 42 SD - SOUTH DAKOTA
<input type="radio"/> 4 AR - ARKANSAS	<input type="radio"/> 17 KS - KANSAS	<input type="radio"/> 30 NJ - NEW JERSEY	<input type="radio"/> 43 TN - TENNESSEE
<input type="radio"/> 5 CA - CALIFORNIA	<input type="radio"/> 18 KY - KENTUCKY	<input type="radio"/> 31 NM - NEW MEXICO	<input type="radio"/> 44 TX - TEXAS
<input type="radio"/> 6 CO - COLORADO	<input type="radio"/> 19 LA - LOUISIANA	<input type="radio"/> 32 NY - NEW YORK	<input type="radio"/> 45 UT - UTAH
<input type="radio"/> 7 CT - CONNECTICUT	<input type="radio"/> 20 ME - MAINE	<input type="radio"/> 33 NV - NEVADA	<input type="radio"/> 46 VT - VERMONT
<input type="radio"/> 8 DE - DELAWARE	<input type="radio"/> 21 MD - MARYLAND	<input type="radio"/> 34 NC - NORTH CAROLINA	<input type="radio"/> 47 VA - VIRGINIA
<input type="radio"/> 9 DC - DISTRICT OF COLUMBIA	<input type="radio"/> 22 MA - MASSACHUSETTS	<input type="radio"/> 35 ND - NORTH DAKOTA	<input type="radio"/> 48 WA - WASHINGTON
<input checked="" type="radio"/> 10 FL - FLORIDA	<input type="radio"/> 23 MI - MICHIGAN	<input type="radio"/> 36 OH - OHIO	<input type="radio"/> 49 WI - WISCONSIN
<input type="radio"/> 11 GA - GEORGIA	<input type="radio"/> 24 MN - MINNESOTA	<input type="radio"/> 37 OK - OKLAHOMA	<input type="radio"/> 50 WV - WEST VIRGINIA
<input type="radio"/> 12 HI - HAWAII	<input type="radio"/> 25 MS - MISSISSIPPI	<input type="radio"/> 38 OR - OREGON	<input type="radio"/> 51 WY - WYOMING
<input type="radio"/> 13 ID - IDAHO	<input type="radio"/> 26 MO - MISSOURI	<input type="radio"/> 39 PA - PENNSYLVANIA	

Select a value

State

Next CAPI CAWI

Back Save and Continue Later

Selecting a state from an alphabetical list shown on a laptop, is much simpler and quicker than using a drop-down list. In CAWI, we cannot control the user device, and so a long alphabetical list would probably involve scrolling.

## 5. CONCLUSIONS

For legacy projects currently using Blaise 4 for data collection, there is no single correct answer about whether and when to make the transition to Blaise 5.

Every project context is different and may need to evaluate specific factors, such as the duration of the data collection, whether there are multiple modes, etc.

Eventually existing projects will migrate to Blaise 5, and new CAPI projects may wish to start in Blaise 5. However, by use of the resource database and control-centre layout tools, it is possible to preserve the ability to use the keyboard as the primary data entry mechanism in Blaise 5 – if desired - and that may be of paramount importance for some studies.

# Rolling Out Blaise 5 – An Interesting Journey

*R. Suresh, Keith Bajura, Matt Boyce, Emily Caron, Lilia Filippenko, Preethi Jayaram, Patty LeBaron, Joe Nofziger, Jean Robinson, Gil Rodriguez, Vorapranee Wickelgren - RTI International*

**Disclaimer:** This paper is simply to report on our findings from the evaluation of Blaise 5. It should not be construed as an endorsement as RTI International does not endorse any software or product.

## 1. Introduction

RTI uses Statistics Netherlands' Blaise software (Version 4.8) for several critical survey projects. Given that Blaise 5 is a significant redesign, offering new capabilities for web and mobile, we did a very careful evaluation of its features and adapted our case management systems for CAPI and mobile to work with Blaise 5. This presentation will describe our journey to incorporate Blaise 5 into our survey software toolkit and our experience in using the newer features offered by Blaise 5.

RTI started its evaluation almost two years ago. In the first year, we focused on the instrument proper – ensuring the features we have used in Blaise 4 are still available and working as expected in Blaise 5, and that these features worked correctly in windows, web, mobile-web and mobile-app modes. In the second year, we switched our focus to adapting our case management systems to work with Blaise 5. Somewhere in between we used Blaise 5 web version in production for a project as well.

## 2. Key Features and Requirements

As with any evaluation of CAI software, we decided to ensure that Blaise 5 had all of the features we needed for a questionnaire and met RTI's other key requirements.

### 2.1 Questionnaire Features

We tested the software on a Windows desktop, a web-browser on a desktop and a small device, and on the Android and iOS apps. The results were very promising. The summary results are shown in Figure 2.1. *Note that these tests were done using version 5.2.5 and it is possible some issues might have been corrected in later versions.*

#### 2.1.1 Single and Multiple responses question

Single and multiple responses questions with radio buttons displayed as expected.

#### 2.1.2 Question with an Image

Images were sized correctly and displayed nicely in all devices. We did not test videos but suspect that they will look fine as well.

#### 2.1.3 Grid Question

Grid questions are always difficult to display on small screens, but Blaise 5 does a decent job with it. On handheld web browsers, in portrait mode it requires us to scroll to the right; on the app versions it truncates/wraps the header so that the grid is fully visible. Given the screen limitations, the questionnaire designer should be creative with the header text so that the resultant display looks better.

#### 2.1.4 Slider Question

Sliders are tricky to display on small devices as well, but Blaise 5 does a good job for the most part. Of the three variations of slider questions, the only issues were that the headers were truncated, or it did not display the number selected but the visual cue was there as part of the slider.

### 2.1.5 Question with Lengthy Response Options

Lengthy response options wrapped nicely in all devices.

**Figure 2.1:** Feature evaluations summary. X indicates there were some issues with the display.

	Radio	Image	Grid	Slider	Lengthy R.O.
Desktop Browser	✓	✓	✓	x	✓
Mobile Browser	✓	✓	✓	✓	✓
Apps Portrait	✓	✓	x	x	✓
Apps Landscape	✓	✓	✓	✓	✓

## 2.2 Compatible with Blaise 4 source code

Given that we have been using Blaise for a couple of decades, we have developed quite a few modules that are re-used over and over again. We did not want to reprogram everything from scratch. In addition, we need to be able to transition long running projects from Blaise 4 to Blaise 5 without too much effort. *Blaise provides a tool to convert Blaise 4 code to Blaise 5 and it really works very well.*

## 2.3 Logic engine needs to be as good or better

Blaise's logic engine is probably one of the best, if not the best, of all CAI software that's available on the market. We did not want to give up on any of the features of the logic engine. *The logic engine is still pretty much the same as Blaise 4 with one subtle exception. Blaise 4 would allow us to enter a numeric answer that is out of range but Blaise 5 automatically truncates it without prompting the user; this could be problematic.*

## 2.4 Program once, use in all modes/devices

This is the ultimate goal of any good CAI software. Even though we program it only once, it needs to have hooks in it so that response options can be customized based on mode. For example, don't know/refused options show up in CATI/CAPI mode but not in self-interview modes. *In Blaise 5, response options, question text, layout and logic can be customized by mode.*

### 2.4.1 Good user experience on all devices

We have no control over the devices that the respondents use to complete a self-interview over the web. We needed the software to support pretty much any device out there, large and small. *Through the use of CSS files, Blaise provides the mechanism for customized user experience for all devices.*

### 2.4.2 Easy deployment

Even if we have the best software in the world, if it took too much effort to deploy it, that sort of defeats the purpose of a CAI software. *The initial server configuration takes a bit of time, but once we master its*

*features, actual deployment is relatively easy for web and mobile apps. It might be useful to make this even more user friendly in the future.*

## **2.5 Cloud**

Our clients are starting to require us to deploy the software in the cloud - specifically GovCloud. The software should be deployable in GovCloud. *RTI was able to deploy Blaise 5 in GovCloud with minimal effort.*

## **2.6 Feature Support**

Given that no one can predict all of the needs of the future, we need Blaise to provide hooks or support the use of special features that are critical. For example:

### **2.6.1 Mechanism for calling custom code & external applications (like alien router/proc)**

We need to be able to call external programs for a variety of reasons. It might be that the clients require us to call a software they provide for a specific section, for collecting an online consent form, or for an appointment scheduling program. Regardless, there are situations where we need to call external programs and the CAI software has to support it. *Blaise provides a mechanism for calling external programs. We might need additional features in this regard.*

### **2.6.2 ACASI and CARI (audio self-interview and recorded interviews)**

Twenty years ago, ACASI and CARI were a novelty. Now, it is pretty much a standard feature in good CAI software. The challenge is that we need these features on the apps for small devices, as well. *These features are built into Blaise 5; of course, some of them are yet to be released.*

### **2.6.3 GPS and other sensors**

GPS coordinates have become an invaluable tool for authenticating interviews. In addition, other sensors and gadgets are used to collect bio-specimens, etc. The software should allow us to connect to these external devices and gather the results in a reliable fashion. *We haven't tested this yet, but the hope is that the feature to call external programs will allow this as well.*

## **2.7 Audit Trail**

RTI uses the audit trail files for salvaging data in case of database corruption, authentication of interviews, and on occasion, for timing of sections. Blaise 5 changed the way audit trails are created and the format is different, as well. *RTI needs to revise its processes and systems to adapt to the new methods and formats for audit trail data.*

## **2.8 Integrate with RTI's case management systems**

Last but not least, Blaise must work with RTI's case management systems (CMS) for CATI, CAPI, and handhelds. Even if Blaise develops a case management system in the future, it probably will not include the organization specific features that are critical; hence, it is important that the software can work seamlessly with our case management systems. *As described later in this paper, we have been successful in integrating Blaise 5 with our laptop and handheld CMS systems.*

## **3. Look and Feel**

One of the major changes in Blaise 5 is that the Mode Library has been replaced with the Resource Database. Given that Blaise 5 must accommodate multiple modes and devices, this change is a welcome move. Having said that, given the amount of flexibility offered by the Resource Database to configure and customize the look and feel of the screens in varied environments, the task of setting up the database is daunting and tedious.

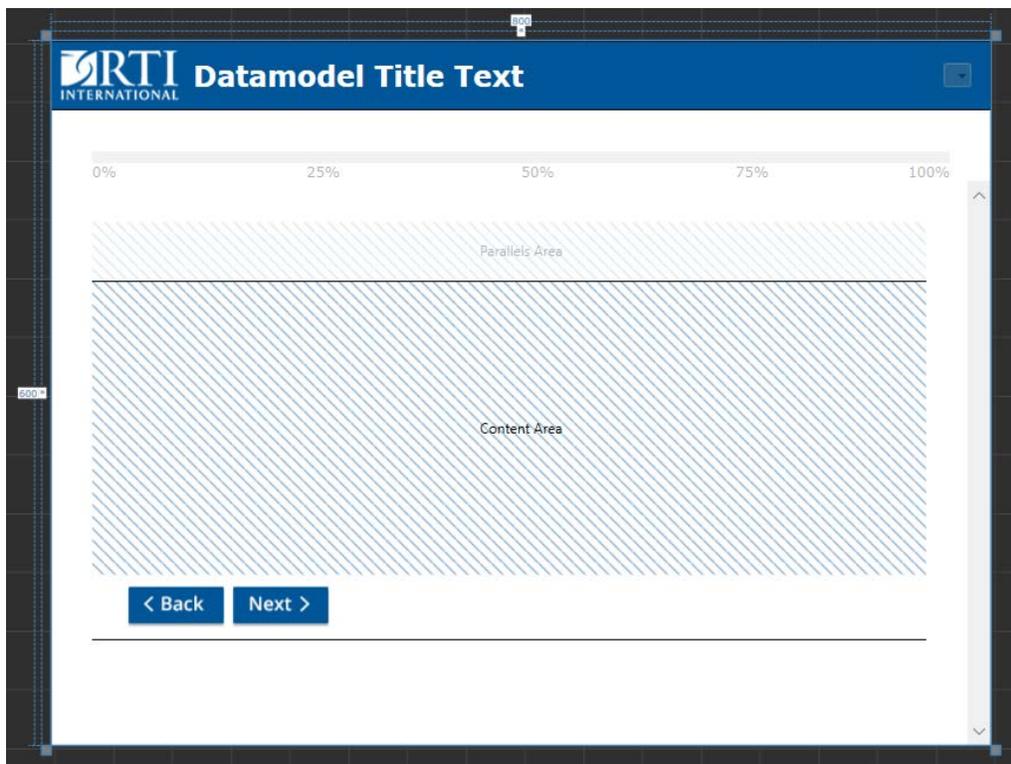
RTI initially looked at the Resource Database and worked on creating an RTI template, but soon decided to use the default environment provided by Blaise until the software was truly production ready. Based on the feedback from other organizations, this approach might have saved us time and effort. Even the default configuration provided by Blaise was sufficient for the most part.

### 3.1 RTI's default setting

Developers enlisted the help of an in-house User Experience / User Interface expert to review and make suggestions on forming an RTI master page template using one of the provided Blaise 5 templates. The goal was to create a look and feel similar to other RTI data collection systems which had been created based on best practices.

The review resulted in a list of specific items to modify, making the process less overwhelming than coming up with a new template from scratch. The requested modifications were doable, but as many have experienced there is a steep learning curve with the Resource Database and making the various changes took time.

**Figure 3.1:** RTI's current master page template.



Examples of some of the Resource Database modifications made:

- Overall page style (color, font tweaks, logo)
- Reformatted and moved Back/Next buttons
- Made updates to table column headers, row selection behavior
- Programmed instrument exit to be done using window 'x' instead of special button
- Modified error text size and location
- Modified watermark style and behavior

RTI's master page template has been tested to work for large and mid-sized screens. A future task is to create a working template for small handheld devices.

### 3.2 Touch Screen version

We experimented with the touchscreen capabilities of Blaise 5 on an MS Windows laptop. We enlarged the size of radio buttons and checkboxes so that the user can select them easily with their fingertips. We also tried using images for response options and the user simply taps on the image to make selections as shown in Figure 3.2.

A few ergonomic issues were discovered in this process and they are not necessarily Blaise issues. Namely, since it is a touchscreen, the user might think that they can:

- swipe up/down on a screen for scrolling, like one would do on a phone/tablet
- expand or shrink the screen display using their fingers

Furthermore, the virtual keyboard took up more screen space than one would expect and so a physical keyboard might be better.

**Figure 3.2:** “All that apply” question for selecting types of lightbulbs. User can tap on the check boxes or lightbulb images to select the corresponding response.

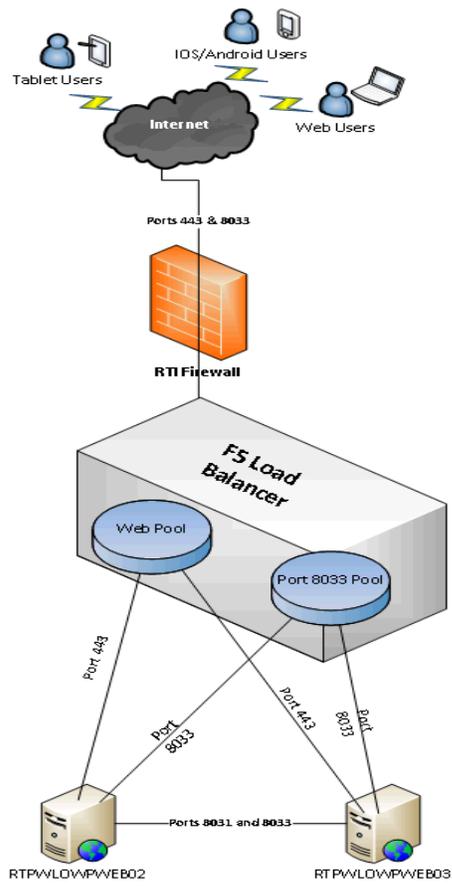
The screenshot shows a survey question interface. At the top, it says: "Please look at the names and pictures of the lightbulbs shown below. Please note that some forms of these light bulbs may look different from the pictures, but you should include any form that you have used." Below this is the question: "Which type of lightbulbs do you currently have in your home?" and the instruction: "Select all that apply." There are three columns, each with a lightbulb image and a checkbox below it. The first column is labeled "Incandescent" and shows a standard incandescent bulb. The second is labeled "CFL" and shows a compact fluorescent bulb. The third is labeled "LED" and shows a modern LED bulb. Below these three columns is a radio button with the text "I don't have any of these types of lightbulbs in my home". At the bottom of the screen are two buttons: "Back" with a left arrow and "Next" with a right arrow.

## 4. Setting Up the Blaise Server Environment

### 4.1 Network Configuration

Figure 4.1 below shows a general diagram of the network connections required to allow Blaise 5 to host both web surveys and mobile device surveys.

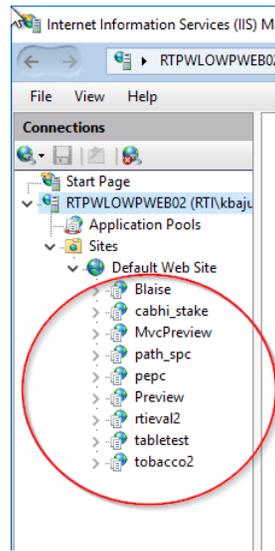
**Figure 4.1:** Network connections for Blaise 5 web and mobile device surveys.



## 4.2 Server Configuration for Web Surveys

Blaise 5 uses a standard IIS web server to host its web surveys. Aside from installing an SSL certificate, configuration of the website is automatically handled either in the initial installation of the Blaise 5 software or by the application itself. Each web survey gets its own subsite under the default web site as shown in Figure 4.2.

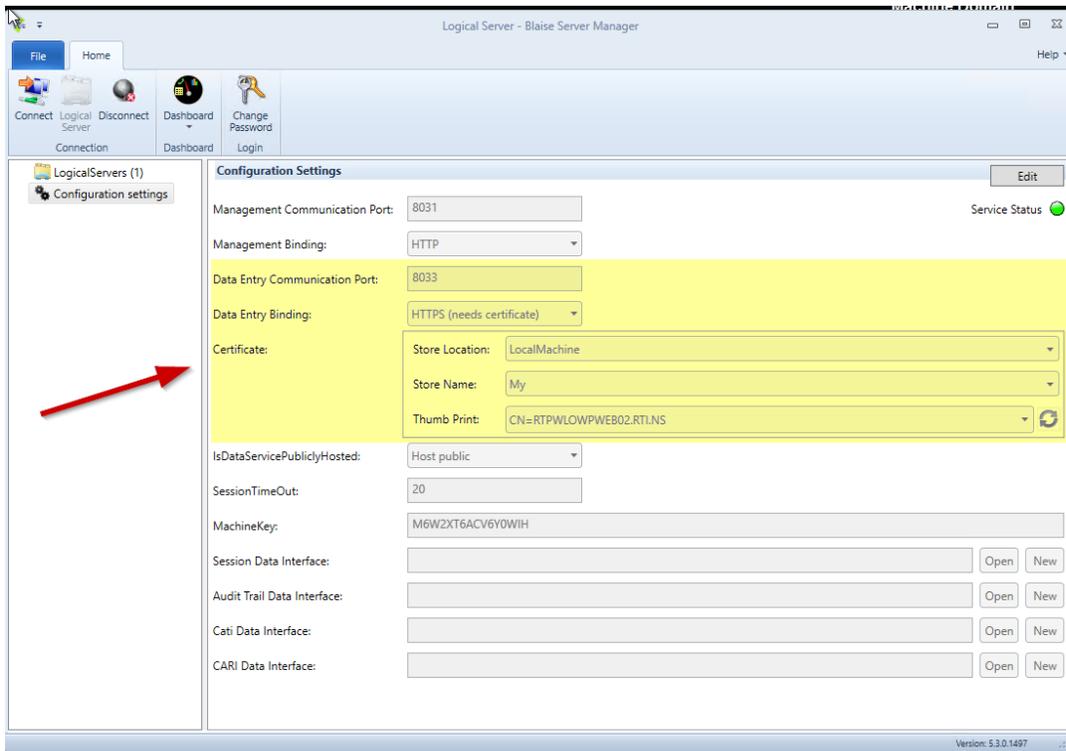
**Figure 4.2:** View of default web sites for Blaise 5 web surveys.



### 4.3 Server Configuration for Mobile Device Surveys

Blaise 5 uses a single port, 8033 (or any non-standard port), for communication to mobile device apps. This is configured by default as port 8033 using HTTP. For security, this setting should be changed to HTTPS and the appropriate server certificate selected as shown in Figure 4.3 below.

**Figure 4.3:** Recommended configuration settings for mobile device surveys.



#### **4.4 Server Firewall Configuration**

If the Windows Firewall is turned on, you will also need to create rules to open ports 443 and 8033 (data). While port 8031 does not need to be opened for internet traffic, it is however used for management between servers in the same “server park” so they can communicate amongst themselves and needs to be opened for this purpose.

#### **4.5 Lessons Learned**

While the configuration is straightforward, it involved multiple people within our IT department to configure different parts of the network as it involves everything from configuring the server to opening ports to firewall changes. We quickly realized that having access to a computer on the internet which is outside of the organization’s network is crucial for testing connectivity. Also, testing connections over various ports can easily be done using telnet.

### **5. Windows version**

#### **5.1 Adapting to RTI’s Integrated Field Management System**

RTI’s Integrated Field Management System (IFMS) is a standard system to use for CAPI projects on laptops for instruments developed in Blaise, RTI’s Hatteras, and other CAI software. The laptop Case Management System (CMS) is configured to launch the specific CAI software as required by the project. Since Blaise 5 uses a different approach compared to Blaise 4 to install and launch Blaise instruments on laptops, we had to adapt our CMS to support Blaise 5. We tweaked our CMS and upgraded the associated Manipula scripts, and now RTI’s IFMS can support Blaise 4, Blaise 5 and other CAI software as needed. A detailed description of this topic is provided in the paper titled, *Blaise 5 with RTI’s Integrated Field Management System on Field Interviewer Laptops*, in this year’s conference proceedings.

#### **5.2 Using the Data Entry API for a WPF survey application**

Blaise 5 replaced the alien router concept with an API that allows it to work with external programs and routines. RTI developed a text to speech application that is used for ACASI sections a few years back and we depend on that heavily for some major projects. Blaise provided a sample C# project called SpeechWPF which served as an excellent starting point for us. We were able to modify that project to implement RTI’s text to speech application in the Blaise 5 instrument. This was successfully tested on a prototype project to everyone’s satisfaction. RTI also used the API technique for other functions such as replaying the question audio, playing audio corresponding to question help, and managing the display of onscreen keyboards for particular questions on touchscreen laptops.

### **6. Web Version**

#### **6.1 Web version in the cloud**

RTI conducted four production Blaise 5 web surveys between fall of 2017 and spring of 2018 on behalf of a client. The surveys had a combined list sample of 889 individuals across the United States. The questionnaires were hosted in on two Amazon EC2 servers behind a load balancer and were backed by a SQL Server database.

##### **6.1.1 Setup and Testing**

In preparation, we initially tested Blaise 5 mock web surveys in an Amazon AWS test environment as well as on local servers. We installed and configured Blaise 5 on the servers. Knowing that our client would be managing the production servers, having our own test EC2 instances gave us the ability to understand the port settings as described in section 4.3, and to provide our client with an informed,

detailed request. In addition, we used these servers to settle on our desired configuration of Blaise services and server parks. Blaise North American Support and Statistics Netherlands were very helpful in responding to our queries during this process.

Installation and configuration of Blaise 5 were done using remote desktop of server instances. In contrast, cases could be loaded into the remote database from an RTI desktop via a VPN connection. The VPN service was provided by our client. Data extraction was done over the same connection.

### **6.1.2 Administration**

Invitations were distributed by email including a link to the survey and unique login credentials. A custom web portal provided login and account management and displayed links to available Blaise surveys per user.

To track the status of unopened/partial/complete for each case, we created indicator variables on different paths in the Blaise questionnaires. A scheduled task regularly exported the variables and updated a control system accordingly.

### **6.1.3 Outcomes**

Each of the four surveys was completed successfully and datasets were delivered. Rarely, a user had a problem with a survey hanging with a page load spinner. Exiting and resuming the questionnaire usually resolved the problem.

These surveys preceded our work on updating instruments in production, described later in this paper. Owing to our emphasis on up-front testing and the short time each of the four surveys was fielded (a few months at most), we avoided such updates.

## **7. Android App Version**

### **7.1 Adapting to RTI's Mobile Case Management System**

Over the years, RTI has developed a powerful set of tools to manage complex surveys that may require tracking cases across multiple waves and varying modes such as mail, in-person, and telephone interviews. Additionally, studies often require that interviewers in the field work offline for extended periods of time which makes using Android tablets running the standalone Blaise app a natural choice. However, in order to make efficient use of this configuration, RTI needed to wrap a comprehensive Case Management System (CMS) around the Blaise Survey engine.

RTI chose an existing CMS created in-house that runs natively on Android devices. This system fully integrates with RTI's existing backend software and allows for many necessary features such as complex case events and histories, managing related cases such as separate screeners and interviews, locking or deleting cases based on predefined responses or remote input from supervisors or RTI, and the ability for interviewers to record events, such as no response to the doorbell, without needing to enter the survey directly.

Statistics Netherlands provides a straightforward Application Programming Interface (API) that allows for the Blaise data entry app to be invoked from another app. This allowed RTI to quickly and efficiently program our CMS to perform the necessary functions in the Blaise app, such as installing an instrument, downloading cases, starting an interview, and uploading data. Additionally, the Blaise app can return the results of an interview to the calling program, allowing RTI to update the status of a case dynamically based on responses and outcomes of a particular survey. Finally, the Blaise app can be programmed to

return directly to the calling program after every action, meaning interviewers never need to enter the Blaise app manually. This provides a single point of entry and simplifies the interviewer's workflow.

## **8. Updating Instruments in Production**

As we are all aware, we have to plan for instrument changes after the start of data collection. Updating an instrument in production without impacting the integrity of the instrument is a challenge.

Blaise 4 manipula scripts provided the mechanism for converting the collected data from the old format to the updated format and this was a very powerful, robust and dependable feature. With Blaise 5, there are even more challenges:

- the database can be in SQL databases rather than file-based structures; and this means that one might have to automatically make backups of the existing SQL database tables prior to converting them to the new format
- the instrument might have to be updated while respondents are actively using the web instruments
- offline versions of handheld apps need to be updated, but the data might still come in the old format

RTI has not had enough time to fully evaluate these scenarios, partly because the feature to update instruments in production came later in the year.

## **9. Features Still Needed and/or to be Tested**

As mentioned earlier, we are eagerly awaiting a few more features that are critical to RTI. Most of them are already on the roadmap. In this section, we highlight some of those features.

### **9.1 CARI**

CARI is a key requirement for RTI studies. We have briefly tested the laptop versions, but additional testing is required. We also need this feature on Android apps which is slated to be released in the near future.

### **9.2 Android Data Entry Client**

We need the ability to delete specific cases via the API. This is necessary to remove cases that should no longer be on the device and may contain sensitive information.

We would also like the ability to easily transfer information, such as responses, from a partially completed case to another interviewer. This will allow field staff to stand in for others without the need to start an interview over from the beginning, which can be frustrating for interviewers and respondents alike.

### **9.3 Instrument Updates**

As mentioned earlier, we still need to evaluate this feature in depth.

### **9.4 Range Checks**

As mentioned earlier, Blaise 4 allowed us to enter a numeric answer that is out of range, but Blaise 5 automatically truncates it without prompting the user. So, if a person wanted to enter the value 105 for age, and the age was restricted at 0-99, Blaise 5 would have registered it as 10 without prompting it as an error. This would have huge implications for data correctness.

## 9.5 Double Key Verification

It seems like a lot of web interviews these days provide the option of hardcopy questionnaires, as well. Therefore, it has become critical to provide a mechanism for double key verification for data entry from a hard copy questionnaire.

## 9.6 Randomization of Responses and Questions

Randomization of responses is useful so that respondents do not simply select the first few options that are provided. Given that quite a few CAI software provide this feature, we are seeing this requirement come up more frequently lately.

Randomization of questions is also critical for questionnaires that collect attitudes and opinions through a series of questions. If the questions are not randomized, there is a potential that the respondents will not carefully read the questions that are presented later in the series.

## 9.7 Automated Testing Tool

Testing the instrument has always been a tedious and expensive task. It would be nice if Blaise 5 provided an automated testing tool where the system automatically selects one of the responses for each question and routes through the instrument. The system needs to provide an interface through which the developer can specify values for preloaded variables, restrict the options that can be selected for gate questions, and adjust the probability of selection of some of the options for some of the questions. Such a tool will help reduce the level of effort for testing the instrument and will generate a test database that also would be very useful.

## 10. Next Steps

The next steps for RTI can be summarized as follows:

- Finalize RTI's standard version of the Resource Database
- Test ACASI and CARI on all devices
- Test updating instruments after the start of data collection
- Tweak manipula scripts for bringing back audit files from laptops
- Do production testing of Blaise 5 for laptops on a project
- Do production testing of Blaise 5 for Android apps on a project

## 11. Conclusion

We all have been striving for a CAI software that:

- is feature-rich
- has a powerful logic engine
- provides hooks for external programs
- supports customization for better user experience
- can be programmed once for all modes/devices

Blaise 5 might be the answer we have been looking for. Not all the bells and whistles are in place yet, but there is a solid plan to get them in place in the next few months and that's comforting. As someone mentioned in one of the conversations, this is just the beginning; there will be requests for more features, more flexibility, and definitely newer modes/devices/technologies/methodologies in the future that have to be accommodated. It is going to be an interesting journey for sure, but we have a great starting point.



# Developing Accessible Blaise Surveys

Karen Brenner, Richard Frey - Westat

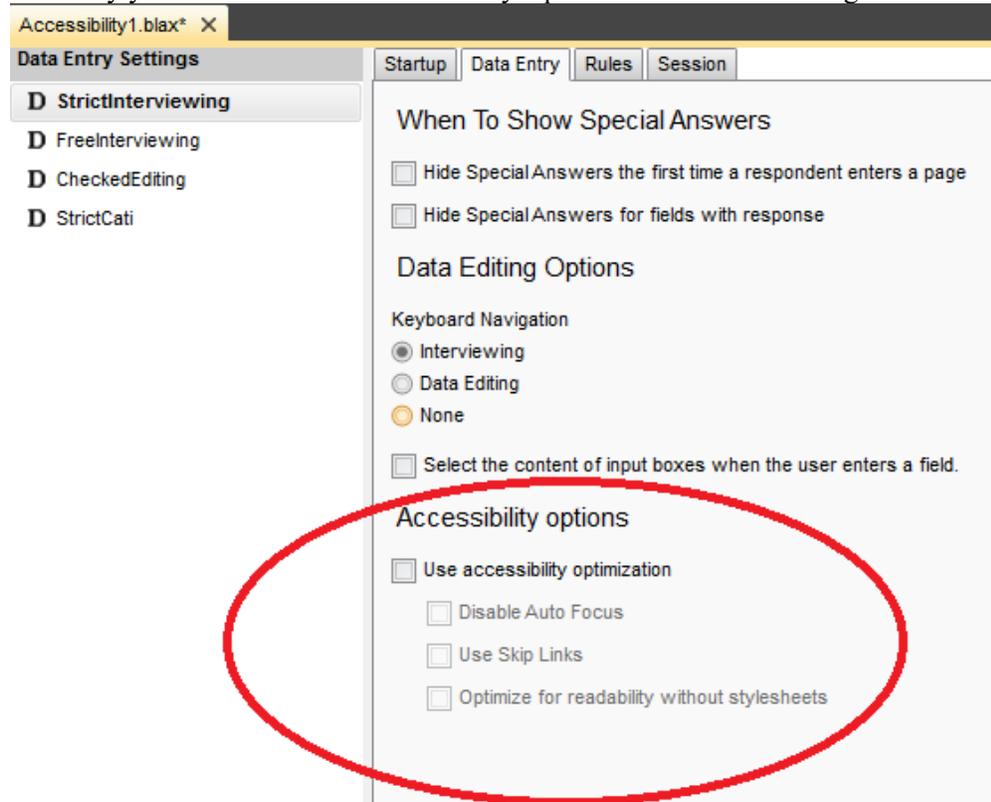
## Accessibility and Universal Design

The ultimate assurance of accessibility, usability, and inclusivity across any system is Universal Design. Any survey requirements document can specify universal design to ensure accessibility for all users. Likewise, the Blaise programmer can develop the survey to meet these basic requirements. WCAG 2.0 are the international conformance requirements for accessibility, and the Section 508 Refresh (of the Rehabilitation Act) are the equivalent of those requirements in the United States.

Accessibility is coding your survey to the widest possible audience, including those with physical, cognitive, neurological, visual or auditory disabilities. Usability is the extent to which users can achieve specified goals effectively, efficiently, and with satisfaction. Universal Design is creating surveys that all people regardless of age, size, and ability can easily access, understand, and use to the greatest extent possible. All users benefit when a survey is accessible, usable, convenient, and enjoyable.

## Techniques

Currently you need to turn on Accessibility Options in the Blaise Settings:



(Screen shot from v5.4.3.1675)

Activating the Accessibility options will turn on certain features within Blaise, and the developer adds the rest. To follow your requirements to meet the conformance requirements for Section 508/WCAG 2.0, follow all aspects of POUR: Perceivable, Operable, Understandable, and Robust. You can follow the techniques below to resolve some of the most common errors that we see in surveys:

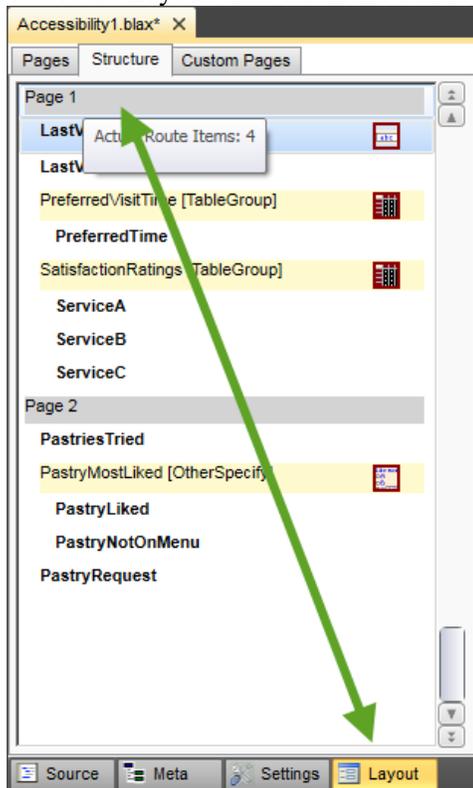
## Color

The contrast of foreground text on a background must conform to minimum contrast requirements, and you cannot use color to convey meaning.

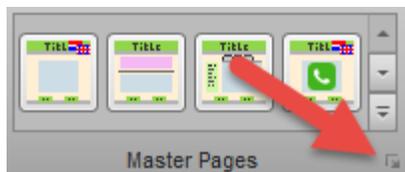
### Added Colors

Set your design colors to pass contrast requirements.

1. Select the Layout view tab in the control center.

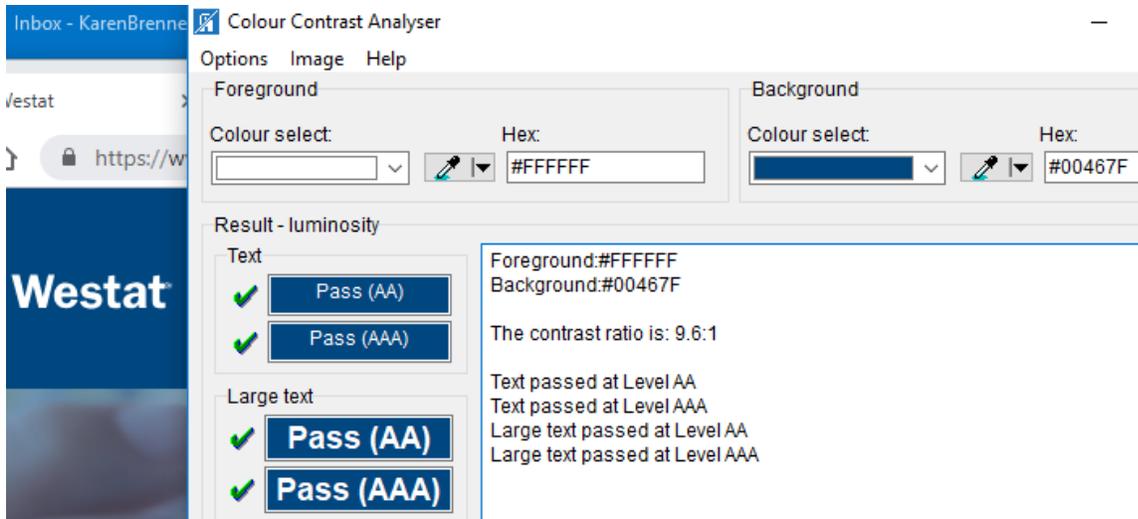


2. Find all color settings for the survey in the Resource Database. In the ribbon bar of the Layout view, most of the groups have a tiny little button in the bottom right corner. Click that to open the Resource Database Editor. For example, in the Master Pages group, the red arrow points at the little “open the Resource Database Editor” button.



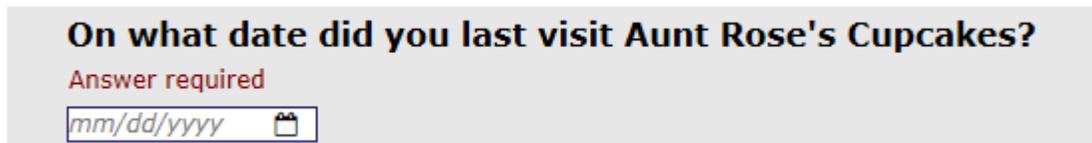
After selecting the Master Pages button, the Resource Database will open to the layout templates for the Master Pages. Each master page will have color properties such as the background color of the survey pages. The colors can be tested either by analyzing the hex value from the property's value or the display of the page in a browser. There are many color analyzer tools available to test your colors; one tool is the Color Contrast Analyzer v2.5, June 2017

<https://www.paciello.com/resources/contrastanalyser/>



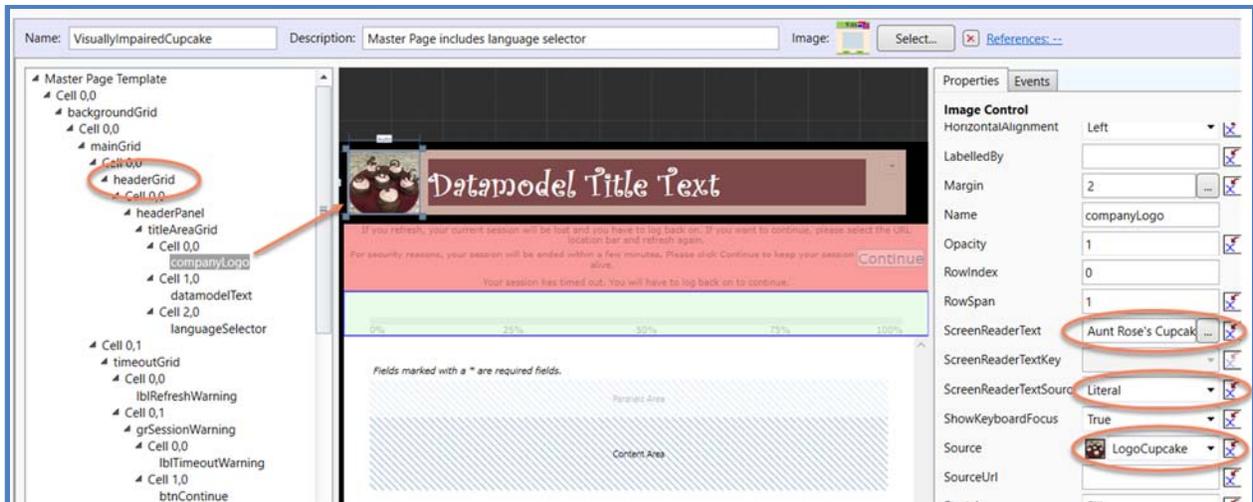
### Default Colors

You may need to adjust, in the Resource database, the default colors for displayed text such as error messages. Navigate to [Blaise Resource Database | Font Definitions | Controls | Error Text](#). On the Font Definition tab, change the Color property from #FFFF0011 (the default red) to “DarkRed” (one word - #8B0000).



### Alt Attributes

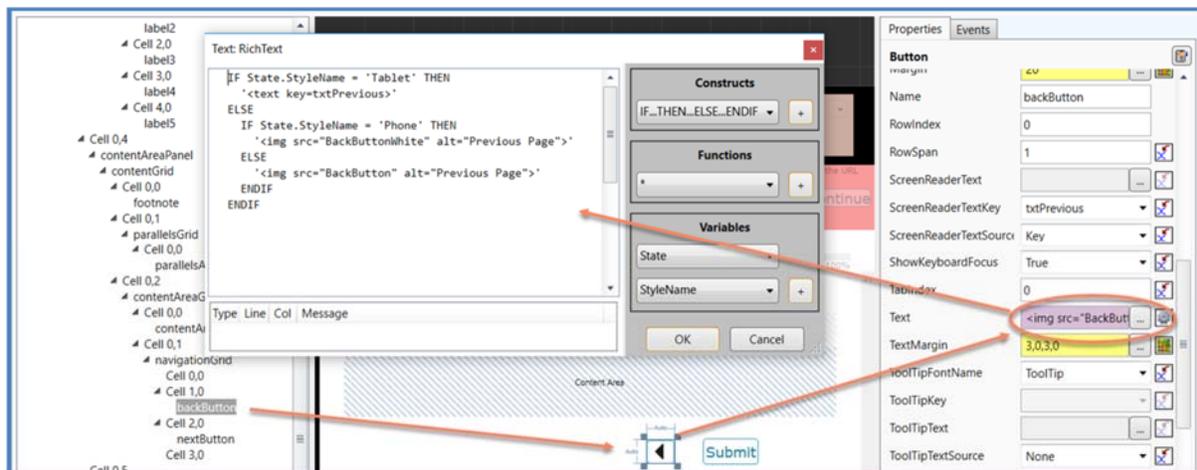
Images used in web surveys must have a description. This is accomplished by using the alt="" attribute. Accomplish this in Blaise by opening the Master Page in the survey’s Resource Data Base. To add a description for the company logo, for example, select the companyLogo Image Control in the headerGrid. Select the ScreenReaderTextSource dropdown property and pick Literal. The ScreenReaderText is now be enabled. Enter text, and save the resource database. In the example below, our alt attribute is alt=“Aunt Rose’s Cupcake Company Logo”.



Filling the alt attribute for other types of images, such as navigation buttons, may require a different method than the Company Logo. If you will only be using images for the navigation buttons then following the steps for the Company Logo will work. However, there may be a need to switch between layout styles, such as from a tablet style to a phone style, where the tablet might have a button using text and the phone using images. If this is the case, then you need an Expression to display the appropriate navigation button.

To add an Expression, first open the survey's Master Page in the Resource Database. Select the backButton Button Control in the navigationGrid. Select the Text property, right click on the  icon and select Expression. Now add an expression similar to the one below or build your own by using the Expression Editor to the left of resulting expression text.

The resulting expression will display the appropriate button as either predefined text or an image. The user needs to add the proper Alt text if the button is an image.



Each style shown in the expression will be associated with a Layout Set defined in the Layout Designer.

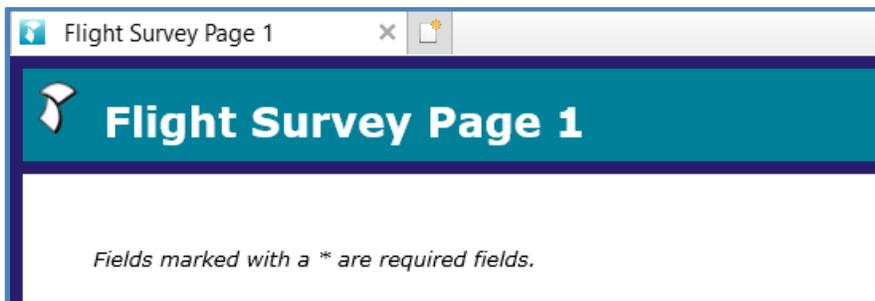
## Headers

There are usually three header types in a survey: the page header, a section header and a sub-section header. Code header elements `<h1 . . .h6>` to the header types in the source. Code the header element for the page header in the Datamodel description text.

```
DATAMODEL FlightSurvey "<h1>Flight Survey ^PageTitle</h1>"
```

The code translates to the page header element as shown in this example from the Flight Survey:

```
<div class="text-container no-auto word-break">
  <div class="">
    <span>
      <h1>Flight Survey Page 1</h1>
    </span>
  </div>
</div>
```



Add header elements to the fieldname with a block type or a Groupname description for both section and sub-section headers.

## FIELDS

```
Person "<h2>Personal Information</h2>" : BPerson
Flight "<h2>Flight Information</h2>" : BFlight
```

```
<div class="text-container no-auto word-break">
  <div class="">
    <span>
      <h2>Personal Information</h2>
    </span>
  </div>
</div>
```

## Personal Information

?
**What is your first name?**

?
**What is your last name?**

The section and sub-section headers are considered a Group in the layout designer and would be assigned the Section group template and the Subsection template respectively.

### Properly Set Properties

Some users might take a survey using the **assistive technology (AT)** of a screen reader. When a screen reader encounters an interactive element, for example a radio button, it needs to find all information related to that button. If not, the screen reader will repeat something like,

“Very satisfied radio button not check  
 Satisfied radio button not checked  
 Not satisfied radio button not checked  
 Very satisfied radio button not check  
 Satisfied radio button not checked  
 Not satisfied radio button not checked  
 Very satisfied radio button not check  
 Satisfied radio button not checked  
 Not satisfied radio button not checked”

In this example, the user cannot distinguish between the different choices as they arrow through a grid. Whether using Aria or Fieldset/Legend, it is important that the Name of the property include the question text, so the AT transmits all relative information to the user.

Note that this topic is one that exemplifies how there are different methods you can use when coding to conform to accessibility standards. You can always find the best techniques for accessibility conformance at <https://www.w3.org/TR/WCAG20-TECHS/>.

Blaise has provide three properties on many controls to assist with providing details about interactive elements.

You can identify sections of a page (landmarks) using the **AccessibilityRole** property. Landmarks help AT users orient themselves to a page and help them navigate easily to various sections of a page. Use the drop-down list to select a section of a page.

Use the **LabelFor** property to explicitly associate the label to a form control and improve accessibility.

The **LabelledBy** property of a control may contain a name of another control within the same template associated with this control. The mentioned control can contain, for example, an explanatory text.

The following enumeration question shows how the Labelfor and Labelledby set up the readable text needed by a screen reader in the Flight Survey:

### Would you like to have an aisle seat next time?

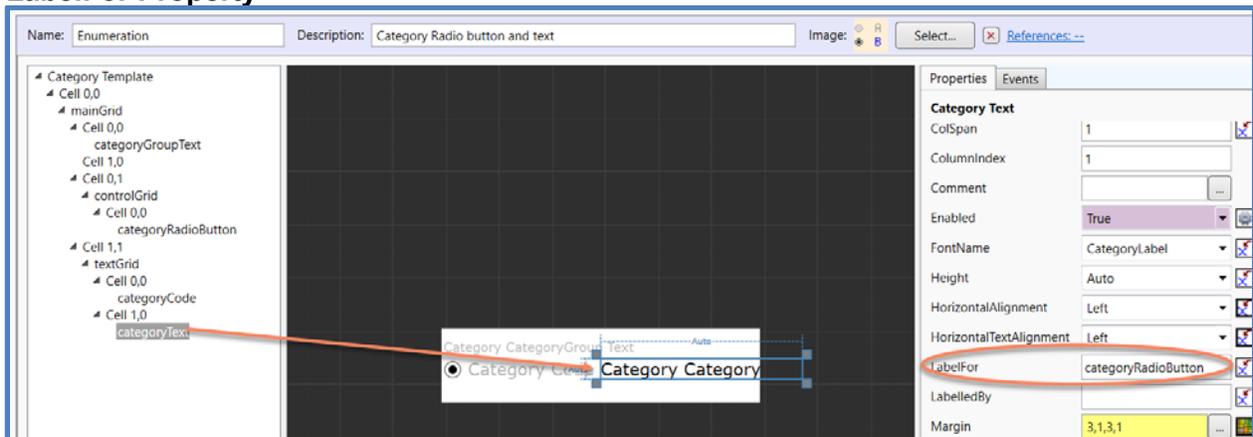
- Yes, please I would like to have an aisle seat next time.
- No, thank you I do not want an aisle seat next time.

The following is the resulting aria-label with the question and category text:

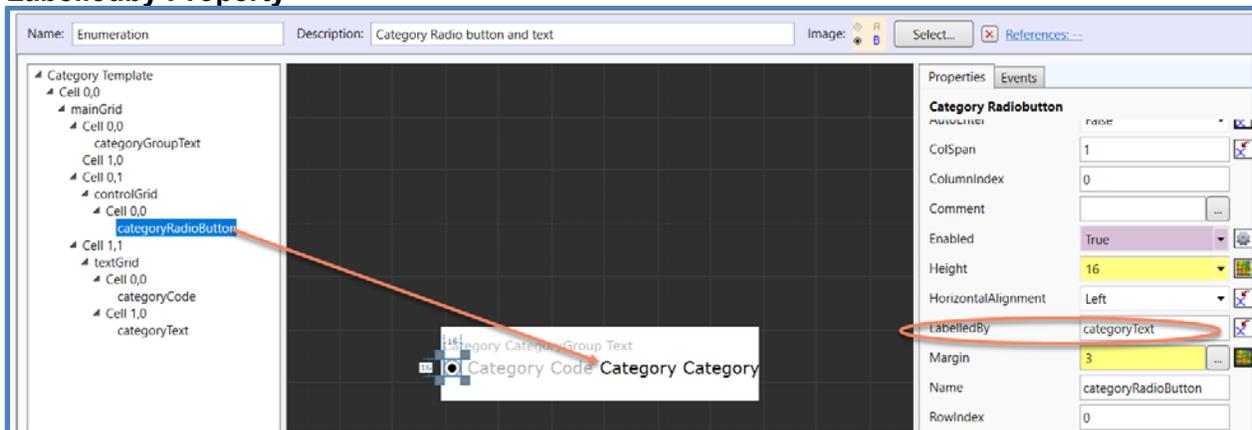
```
<input name="Flight.FirstClass" tabindex="0" class="Category  
RadioButtonComponent visibility-visible enabled focus-outlin  
e width-pixels height-pixels" id="aga_1ca_2gaba_1e" role="ra  
dio" aria-checked="false" aria-hidden="false" aria-invalid  
="false" aria-required="true" aria-labelledby="aga_1ca_2gaba  
_1h" aria-live="assertive" style="margin: 3px; width: 16px;  
height: 16px; visibility: visible;" aria-label="Would you li  
ke to have an aisle seat next time? Yes, please I would like  
to have an aisle seat next time." type="radio"  
data-fieldname="Flight.FirstClass" />
```

To achieve these results the Enumeration template must have the Labelfor and Labelledby properties set:

### LabelFor Property



## Labelledby Property

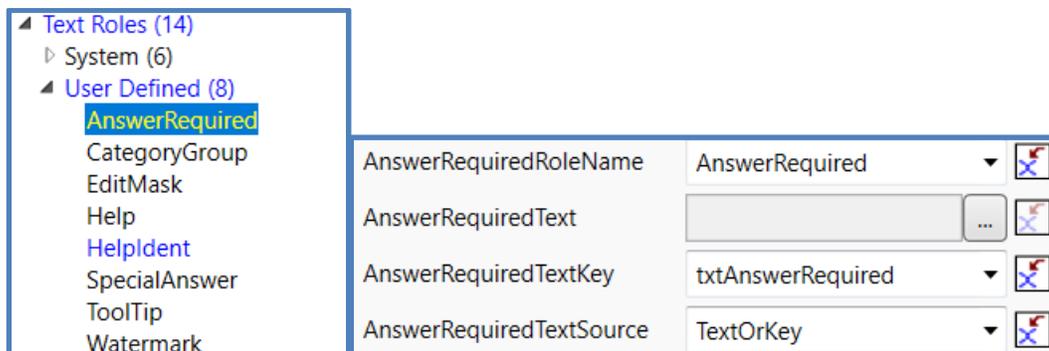


## Required Field Error Messages

Another interactive element an AT user will encounter is the “Answer Required” error message. This standard message displays when a respondent does not enter a value in a required field. A sighted person would easily associate the displayed message “Answer Required” to the question that was not answered. A respondent listening to the survey with AT will need to understand which question was not answered if there are multiple questions on a page.

Blaise provides the ability to customize “Answer required” messages for each field that has a required attribute. Accomplish this using the AnswerRequired role text and a set of AnswerRequired properties of the ‘error text message’ found in many of the FieldPane templates.

The AnswerRequired role text is located in Role Texts|User Defined and comes with the default resource database. The AnswerRequired properties are found in the ‘Error Message Text’ of a field pane template.



To customize the text of error messages, the following properties are used:

- With the *AnswerRequiredRoleName* property, you can decide which control's Text Role you want to use as an error text.
- You can provide literal text that displays as soon as an answer is required with the *AnswerRequiredText* property.
- The *AnswerRequiredTextKey* property allows you to provide a user-defined translatable text.

- You decide whether there should be an answer required error text with the *AnswerRequiredTextSource* property. If so, you choose which bound element should be its source.

Next, add AnswerRequired to the Roles section in the datamodel.

```
ROLES = Help "The Help role provides a help button that the user can click to display question help text.",
HelpIdent "HelpIdent helps to identify the help button for blind and partially sighted people.",
Watermark "The Watermark role provides a hint to the user as to what input elements are used for.",
Tooltip "The Tooltip role provides a popup hint when the user hovers over a control.",
AnswerRequired "The AnswerRequired role can be used to point out to the respondent that a question has not yet been answered."
```

Then add the AnswerRequired role text to the questions with the NoEmpty attribute for which you want to clarify the AnswerRequired message.

```
LastName "What is your last name?"
AnswerRequired "Your last name has not yet been entered. To continue, your last name must be entered."
Help "Please enter your family name."
<newline>Your privacy is guaranteed. Your name is only important to provide good service."
HelpIdent "Help button for the question about your last name."
: STRING[20]

Age "What is your age?"
AnswerRequired "To continue, your age must be entered."
Watermark "0..150" : 0..150

Town "Where do you live?"
AnswerRequired "Your hometown is not yet known. Enter your hometown." : STRING[20]
```

The screen shot below shows an example of different Answer Required messages.

**What is your last name?**  
Your last name has not yet been entered. To continue, your last name must be entered.  
=

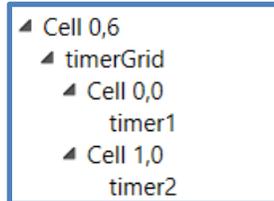
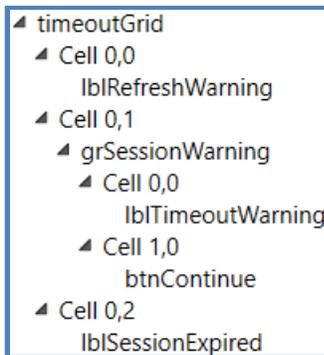
**What is your age?**  
To continue, your age must be entered.  
=

**Where do you live?**  
Your hometown is not yet known. Enter your hometown.  
=

### Setting notification of session expiration

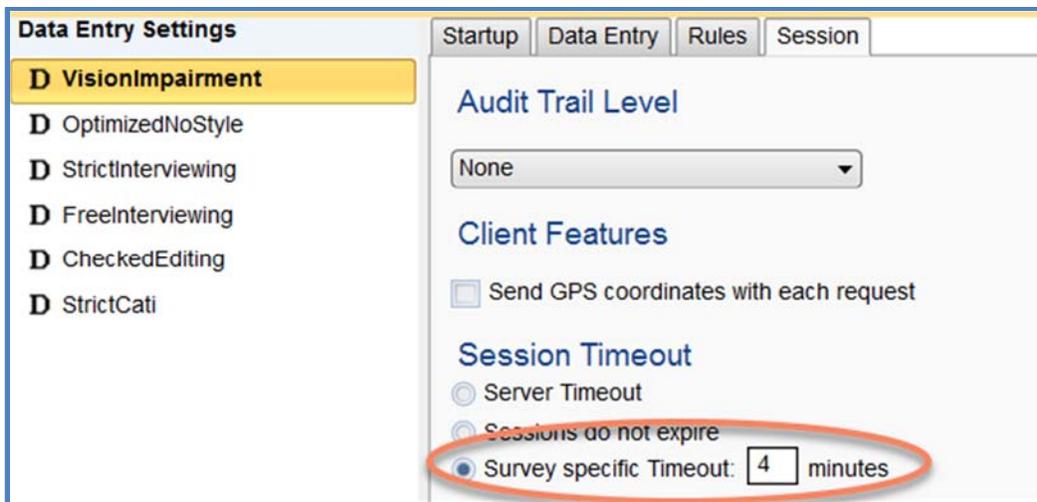
AT users need a warning when their session is about to expire. In some cases, they may need to re-authenticate to continue their session, and they want to avoid any loss of any data. If someone reads slowly or has trouble navigating, they may require additional time to complete an activity.

Blaise 5 provides an example of warning messages and the use of timers to warn users about the status of their current session. You can copy the session warning messages and timer structures from the VisuallyImpaired resource database and inserted into a survey's Master Page.



Copy the time Grid(Timeout grid) and the Grid(TimerGrid) from the VisuallyImpaired sample's blrd into your master page. To do this, add a row after the header in your master page. In the VisuallyImpaired master page, right click on Grid(TimerGrid), click copy, then go to your new row in your master page, right click and paste. Repeat the same step for the Grid(Timeout grid) but add the new row below your Navigation grid.

If the server session timeout is too long, the user can adjust the timeout in the Session tab of the Control Centre Settings.



Extend the Timeout depending on the length of your page to accommodate a slow reader, someone using AT, a keyboard only user, etc.



The warning messages should appear indicating the action the respondent can take, allow the user to take action using their keyboard, and provide enough time for all users to take action.

## Summary

These are some examples of the many ways a developer can make surveys accessible. After addressing the accessibility issues described here, our survey passed Section 508/WCAG 2.0 testing. The tester used

a combination of manual testing and tools recommended by the Department of Homeland Security Trusted Tester certification training. The only outstanding issues were those related to readability without style sheets. Although Blaise relies heavily on style sheets, the Access Board proposed 1194.22(d) related to style sheets be removed from the conformance criteria since assistive technologies now support CSS very well. Blaise has made many improvements in v5.4 that allow designers and developers to create accessible surveys.



# Enhancing Blaise 4 Surveys for JAWS Screen Reader Compatibility

*Mecene Desormice, Michael Mangiapane, Erin Slyne, Richard J. Squires Jr. - U.S. Census Bureau*

## 1. Introduction

There are a number of initiatives within the U.S. Government to build a more representative and diverse workforce by hiring employees with disabilities. Some disabilities require accommodation by providing hardware or software that assists a person in using a computer. We have several interviewers that work in the U.S. Census Bureau's CATI call centers that have impaired vision and use such assistive devices. When conducting interviews, it is important that they are able to accurately read questions to the respondent and enter the given answers.

Job Access with Speech (JAWS) is a screen reader program for Microsoft Windows that allows visually impaired users to read the screen with a text-to-speech output or by a refreshable Braille display<sup>1</sup>. JAWS operates by creating a virtual "JAWS Cursor" that may be moved around the screen via the keyboard in order to read text or describe certain text attributes such as color or the use of bold or italics. The visually impaired CATI interviewers at our call centers use both text to speech and the braille display with Blaise.

When the Census Bureau started to write instruments in Blaise 4, they developed a set of screen standards for Blaise instruments. Those screen standards created a consistent survey experience for interviewers and simplified their training. The standards also streamlined instrument programming by providing a known format for each screen. They included font settings for question text, interviewer instructions, and enumerated answer lists. We continue to use these standards for all of our Blaise instruments.

The JAWS users at the call centers reported to the Blaise authors that there were elements and screens in our instruments that could be better when used with JAWS. We decided to develop screen standards specifically for JAWS users to create a more consistent experience for the interviews and reduce potential confusion while using JAWS with Blaise.

## 2. Overview

### 2.1 Reviewing current survey instruments with JAWS

Our team was responsible for evaluating how well JAWS works with Blaise for three CATI instruments and making any necessary improvements. These instruments were the Telephone Point of Purchase Survey (TPOPS), the Medical Expenditures Panel Survey (MEPS) Pre-screener, and MEPS Telephone Follow-Up.

All of us were new to using screen readers and JAWS so we obtained feedback from our call center interviewers and supervisors who were already using the software. We arranged Skype meetings with a call center supervisor and watched her navigate through our instruments with JAWS. She identified screens where JAWS was not reading text accurately and screens that were difficult for the visually impaired interviewers to navigate. During the meeting, we learned about different settings in the JAWS software that the interviewers utilize to help with their interviews. At a later meeting, we listened to an interviewer conduct a mock interview while they were using JAWS.

<sup>1</sup> JAWS (screen reader). (2018, July). In *Wikipedia*. Retrieved from [https://en.wikipedia.org/wiki/JAWS\\_\(screen\\_reader\)](https://en.wikipedia.org/wiki/JAWS_(screen_reader)) 07/30/2018

Observing the supervisor and interviewer use the instrument with JAWS helped us conceptualize better solutions for our instruments. After gathering their feedback, we created a spreadsheet listing each of the comments we received from the call center and our notes from the observed interviews. Our team met on a regular basis to discuss a plan of action for each item.

## **2.2 JAWS users at Census**

During our observations, we discovered which JAWS features our interviewers utilize. We learned that the “JAWS cursor” is used to navigate the visible screen space, and the “PC Cursor” is used for normal keyboard navigation. At the start of every screen, the interviewers pressed Ctrl + Home to move the JAWS cursor to the top of the screen. Then they pressed the down arrow key to move the JAWS cursor so that JAWS spoke the text line-by-line. The interviewers prefer to use the arrow keys to move the JAWS cursor rather than having JAWS read the entire text at once so they can control the speed and in case they need to repeat any text. We also observed how the interviewers switched to PC cursor mode from JAWS cursor mode when encountering Blaise tables and lookup tables.

## **2.3 Prototype development**

Due to the complexities of our instruments, we developed a small-scale prototype to focus on the identified JAWS issues. We gathered “problem” screens from the different instruments we were evaluating. Each issue included an example of a current “before” screen and a corresponding “after” screen that implemented a suggested solution. At our meetings, we presented the prototype to the call center supervisor (with JAWS turned on) via Skype to obtain her feedback.

## **3. Proposed Issues and Resolutions**

The team has incorporated resolutions to many issues in the instruments we are evaluating. We plan to incorporate these fixes into future instrument development.

### **3.1 JAWS User Flag and Parameter**

It is possible for multiple interviewers to work a case before its completion. In order to minimize the impact of screen modifications for non-JAWS users, one requirement specified that screens should have the ability to display with either JAWS friendly or regular (non-JAWS) formatting. Our CATI system should have the capability to launch cases in JAWS mode or non-JAWS mode.

We added a flag to our CATI systems, set by supervisors, to indicate if the current interviewer is a JAWS user. By having a standard flag, our programmers know what variable to use when programming changes to the instrument. The flag passes in as a parameter when we call the instrument in CATI. If the flag’s value equals “1” then the JAWS-friendly screens display. If the flag has no value, the standard screens display.

### **3.2 Interviewer Instructions**

Our Blaise screen standards include interviewer instructions, which the interviewer does not read to the respondent. A blue diamond precedes the instruction, and the instruction text is in blue Arial 14pt font. Since Blaise 4 displays any OpenType font, we use the Wingdings font (lowercase S) to display the diamond (see Figure 1).

We learned during the initial feedback that JAWS does not interpret the diamond as expected. By default, JAWS does not say or display that character as “diamond.” Instead it says or displays the code that is set for the Wingdings character itself, resulting in the interviewer hearing or reading “**Wingdings sixty one thousand one hundred fifty five**” (hereafter referred to as the Wingdings code). It is similar to the alt-text attribute for HTML images that is used by screen readers to say or display the text associated with the image. Whenever JAWS would read an interviewer instruction, the user would hear or read the Wingdings code followed by the instruction text, which slowed down the interview.

Another scenario was a screen with interviewer instructions displayed before question text, such as in Figure 3a. Additionally, other screens may contain multiple interviewer instructions after the question text but before JAWS would reach an answer list for an enumerated question. In both situations, we needed JAWS to alert the interviewer that they had reached the end of the instruction(s) and were moving on to a new type of text.

Initially we tried looking at the JAWS settings and even the JAWS dictionary to see if there was a way we could tell JAWS that if it read the Wingdings code to say or display a different phrase such as “diamond” or “interviewer instruction.” However, we were unable to use either method to change how JAWS was speaking or displaying the character as JAWS continued to read the Wingdings code.

We resolved the issue by substituting the diamond with a different character for JAWS users. After listening to how JAWS reads several different characters, our team decided that we would use the left brace, {, to denote the start of interviewer instructions and the right brace, }, to denote the end of the instructions. The braces also take up less space on the Braille reader and are easier for the interviewers to read. We do not use these characters regularly so they became the best choice for our requirements. For screens that have multiple interviewer instructions grouped together, we decided to use only one set of braces to eliminate redundant instruction notifications.

To display the braces for a JAWS user or the diamond for a non-JAWS user, we programed text fills for all interviewer instructions. The fills were simple to program but every question needed to be reviewed and updated if it contained interviewer instructions. We used the “find” option in the Control Centre to locate all text with the “diamond” code, which we program as “@Zs@Z.” We modified the instruction coding like this:

```
IF JAWSFLAG = 1 THEN
    FR_Instruction := '{ '
    FR_Instruction_2 := ''
    End_Instruction := '}'
ELSE
    FR_Instruction := '@Zs@Z'
    FR_Instruction_2 := '@Zs@Z'
    End_Instruction := ''
ENDIF
```

```
@L^FR_Instruction Press Insert key to add to/edit field contents.@L
```

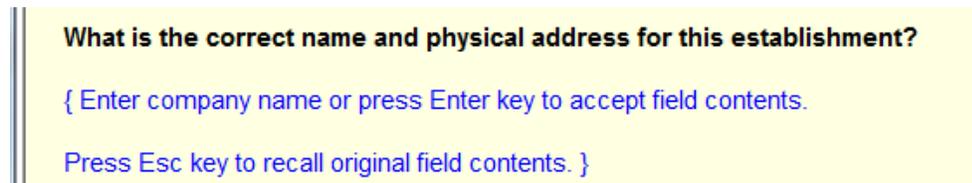
```
@L^FR_Instruction_2 Press Esc key to recall original field contents.
```

```
^End_Instruction@L
```

Figure 1 - Interviewer Instructions for non-JAWS users



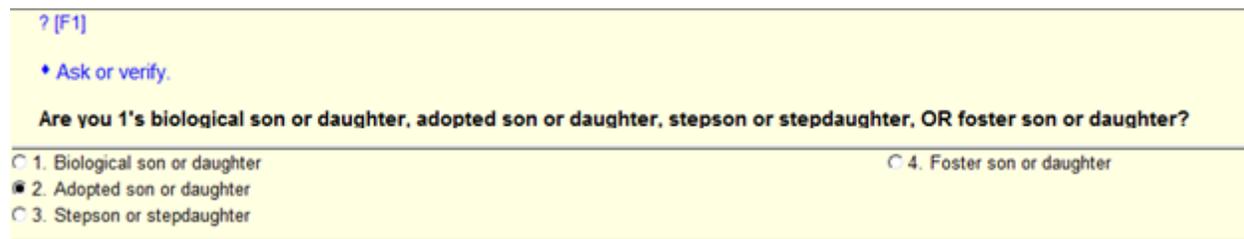
Figure 2 - Interviewer Instructions for JAWS users



### 3.3 Multi-Column Answer Lists (Answer Choices)

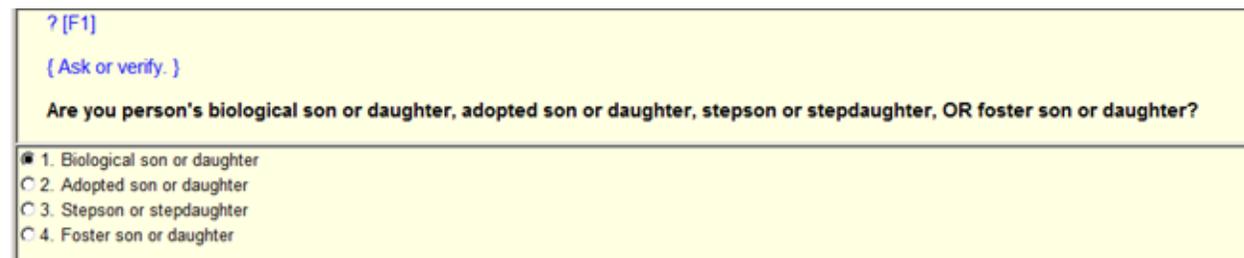
If an Info Pane allows for more than one column of possible answer choices in an enumerated question, JAWS will read the answers out of order since JAWS reads each line from left to right. In the figure 3a below, JAWS says or displays the choices as “**1 Biological son or daughter 4 Foster son or daughter 2 Adopted son or daughter 3 Stepson or stepdaughter.**”

Figure 3a - Multi-column answer list (before)



Our recommendation was to modify the layout instructions to use a one-column Info Pane for this screen so all options would align in one column. However, it is important to verify that the change does not introduce scrolling as JAWS does not have the capability to auto-scroll and it does not always alert the user that there is a scroll bar. We also made sure not to affect the display of other fields on the same pages.

Figure 3b – Multi-column answer list (after)



### 3.4 Multi-Column Form Panes

In Blaise 4, we can specify which questions to group together in a form pane with layout instructions specifying the grids and field panes to use for each page of the instrument. For most of our instruments, we tend to have multiple columns of questions in the form pane area:

Figure 4 - Multi-Column Field Pane

Name1	ANYTOWN COUNTY NURSING HOME	Street1	877 CEDAR BLUFF RD
Name2		Street2	
		City	ANYTOWN
		State	AL
		Zip5	99997
		Zip4	0000

As mentioned above, JAWS will typically read items on the screen from left to right. In Figure 4 above, JAWS would say “**Name one Anytown County Nursing Home Street one eight seven seven Cedar Bluff Road.**” We can program JAWS to pause on the field where the cursor is located, but that does not always tell the interviewer exactly where they are in the instrument.

We decided to update the instrument layout so that all questions in the form pane are in one column per page. As a JAWS user moves their cursor, it will be more apparent which question they are currently answering. This change does not adversely affect non-JAWS interviewers.

Figure 5 - Single Column Form Pane

Name1	ANYTOWN COUNTY NURSING HOME
Name2	
Street1	877 CEDAR BLUFF RD
Street2	
City	ANYTOWN
State	AL
Zip5	99997
Zip4	0000

### 3.5 Multiple Question Questions / Multiple Answer Questions

Some screens present multiple answer inputs based on one question.

Figure 6 - Asking for the name and title with different input boxes for each



May I have your name and title please?

- ◆ Press [Insert] key to add to/edit field contents.
- ◆ Press [Esc] key to recall original field contents.

Respondent Name	Mary Citizen Supervisor
Respondent Title	

We learned that our JAWS users were not always clear that they needed to type this information into two different response items. In the above example, they would type the name, “Mary Citizen,” and the title, “Supervisor,” into the name field.

As we have identified these screens, we have worked with the survey sponsors to modify the screens so clarifying instructions are displayed on the screen for JAWS users. The instructions were updated for both JAWS and non-JAWS users.

Figure 7 - Same question as Figure 6 with additional clarifying instructions



May I have your name and title please?

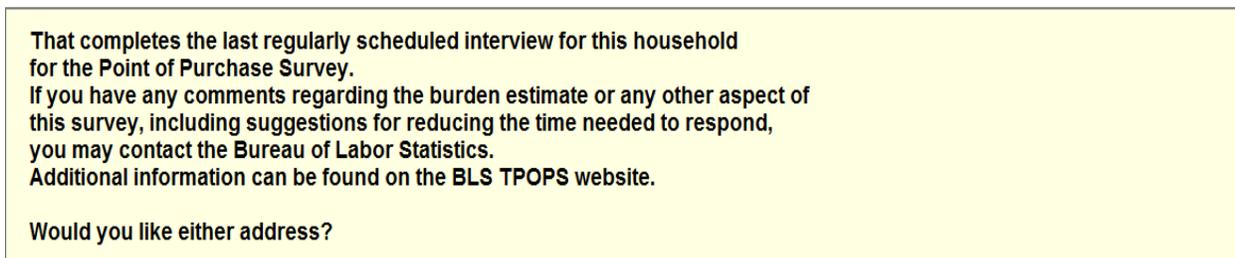
{ Enter name or press Enter key to accept field contents.  
Press Esc key to recall original field contents. }

Respondent Name	Mary Citizen
Respondent Title	

### 3.6 Hard Returns in Question text

We use hard returns in our surveys to add spacing to questions, interviewer instructions, and to split long questions or fills into multiple lines. This helps the interviewers or respondents follow the flow of the survey and makes it less likely for interviewers to misread or overlook questions or instructions.

Figure 8 - A screen with extra hard returns for the question text

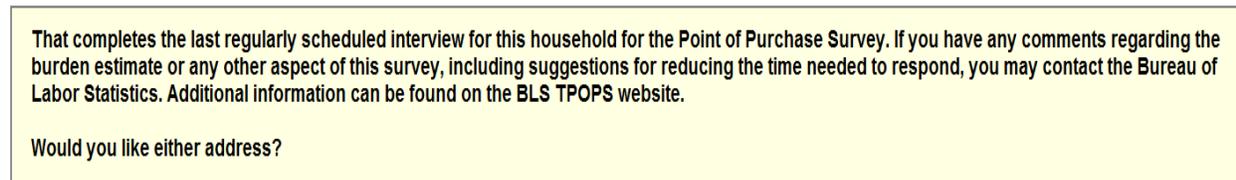


That completes the last regularly scheduled interview for this household for the Point of Purchase Survey.  
If you have any comments regarding the burden estimate or any other aspect of this survey, including suggestions for reducing the time needed to respond, you may contact the Bureau of Labor Statistics.  
Additional information can be found on the BLS TPOPS website.

Would you like either address?

Despite the usefulness of hard returns in formatting some of our surveys, this practice does not work as well for JAWS users. The issue with using hard returns in question text is the more lines the interviewer has to navigate using JAWS, the more “pauses” they hear from JAWS. To alleviate the burden on the JAWS users we decided to remove all unnecessary hard return “@/” from within the question text and fills in the CATI instruments.

Figure 9 - The figure 8 screen with extra hard returns removed



The effort to remove all the hard returns was tedious. We could not do a search and replace for “@/” in the code as we might inadvertently delete hard returns that were intended to be included in the question text. Together, sponsors and authors identified each screen to be modified and thoroughly tested the changes to make sure the screens work for both JAWS and non-JAWS users.

### 3.7 Info Pane and Form Pane Scrolling

The JAWS software does not alert the user if there is scrolling in the info pane or form pane. Since many of the interviewers are exclusively using their keyboard, they may not be aware that there is more question text or even additional answers for an enumerated question. As we made changes to the layouts, we tested each screen to make sure they displayed all text without a vertical scrollbar.

Another factor we also must keep in mind is that the interviewer may be running at a different resolution so our instrument should be able to display without scrolling for multiple resolutions. The best way to design to avoid scrolling for JAWS users is to make sure the screens do not scroll at the lowest possible resolution of the hardware that is used.

### 3.8 Exiting Household Rosters and similar Tables

As a standard in our Blaise 4 instruments when collecting a household roster of names, if there are no other members in the household to collect, the user enters “999” in the first name field to exit the data collection screen.

We found instances where JAWS users accidentally entered “99” or “9999” in first name, which added an additional person to the table. Once added, it can be difficult to remove this accidental person from the table. It proved to be even more difficult for a JAWS user.

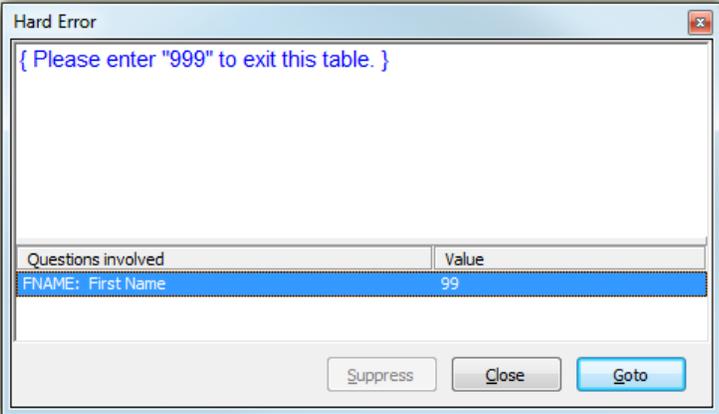
To mitigate this problem, we added an edit for all users to prevent them from entering “9,” “99,” “9999,” or “99999.” The edit message displayed upon those conditions is:

“Please enter “999” to exit this table.”

Figure 10 - Edit check to prevent exiting a roster if user did not enter "999"

Solution: Set up checks to ensure the user does not enter "99" into the FName field.

{ Enter First Name  
Enter 999 if no more persons }



Questions involved	Value
FNAME: First Name	99

Stat	Stat	First Name	Last Name	Mem	Ure	Sex	Rel	Birth Month	Day	Year
1]		Mary	Citizen	1	2	2		1	1	1990
2]		99								

## 4. Challenges Encountered

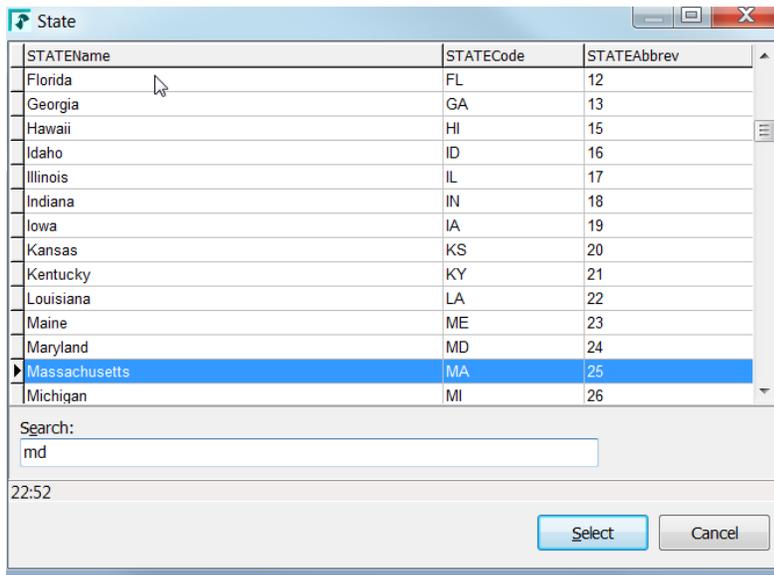
There are still a few requirements that are being actively developed and refined to create the best experience for our JAWS users. As we get feedback, we will incorporate their recommendations into our JAWS screen standards.

### 4.1 Lookup Tables

By default, Blaise displays all the fields listed in the datamodel of a lookup table. Some reports from our JAWS users are that they experience some potential issues when accessing lookup tables, especially when there are multiple columns displayed.

- JAWS reads the text that the user types into the lookup table, but it does not read the highlighted selection in the table.
- JAWS reads the first line of the table automatically before starting to read the search line that shows up when the first few characters were typed.
- Interviewers do not know immediately that they are in a lookup table.
- Interviewers cannot move through the list with the JAWS cursor. They must switch to the PC cursor to do this and then switch back to the JAWS cursor when they are finished.

Figure 11 - State lookup table with default view



STATEName	STATECode	STATEAbbrev
Florida	FL	12
Georgia	GA	13
Hawaii	HI	15
Idaho	ID	16
Illinois	IL	17
Indiana	IN	18
Iowa	IA	19
Kansas	KS	20
Kentucky	KY	21
Louisiana	LA	22
Maine	ME	23
Maryland	MD	24
Massachusetts	MA	25
Michigan	MI	26

Search:  
md

22:52

Select Cancel

Ideally, JAWS should read the highlighted selection in the table so the interviewer knows they selected the correct option. Otherwise, there could be a mistake and the interviewer would not know.

We tried different approaches to remedy this issue such as writing a Manipula program and creating a custom data viewer. The Manipula program activated a box that looked like the lookup table but it had a Title Bar that included text to tell JAWS users to switch to the PC cursor. The contents of the table are shown to the user with only the necessary fields, such as State Name and State Code in Figure 11. The Manipula attempt looked promising but we discovered that the Manipula script treated the first character keyed as what activates the lookup so it is not carried into the search bar and would not be spoken by JAWS.

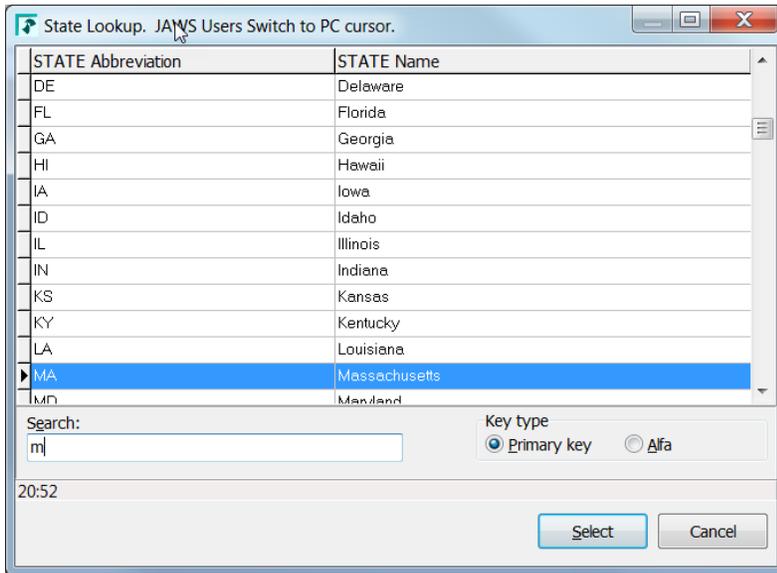
We decided to proceed with a custom lookup table using the Blaise Data Viewer. The goals are to alert the JAWS users that they have accessed a Lookup table and they need to switch to PC cursor to navigate through the Lookup table and to remove any unnecessary columns in the Lookup table to eliminate extra text read to the user.

We created a Blaise Data Viewer (.BDV) file for each lookup table to filter and customize content displayed to the user. We can reuse this file for other survey instruments that use the same lookup tables. However, if the database structure of the lookup table changes, then the BDV file needs to be re-created. This change to the lookup tables will be available to all users.

A JAWS-friendly BDV file should:

- Add a Lookup Table Title and Standard wording: <Table Name> + “Lookup” + “JAWS Users Switch to PC Cursor.”
- Modify Column Headings so that JAWS can clearly read them to indicate the information contained.
- Remove any unneeded extra columns from the view.

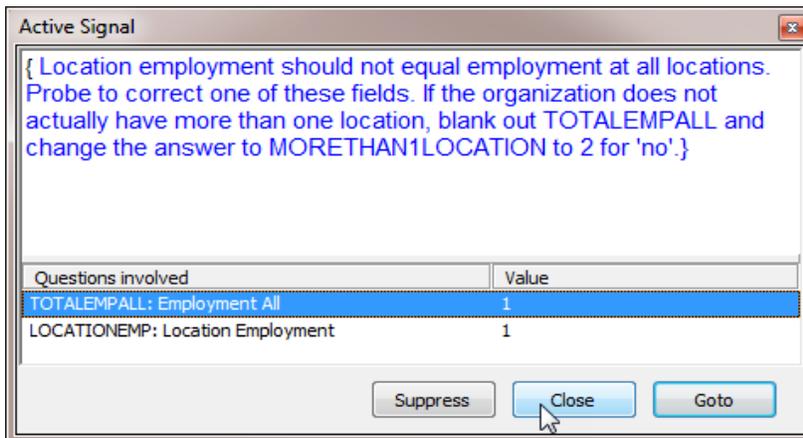
Figure 12 - State lookup with BDV for JAWS



## 4.2 Edit Checks

JAWS users have encountered issues with edit checks that are similar to the issues with lookup tables. When the edit check window pops up, JAWS reads the title of the edit check (Active Signal or Hard Error) and then skips to the bottom of the screen without reading the text from the edit check. Like the lookup tables, the users do not always know immediately that a popup window has appeared and they should use the PC Cursor to navigate.

Figure 13 - Active Signal with JAWS interviewer instructions

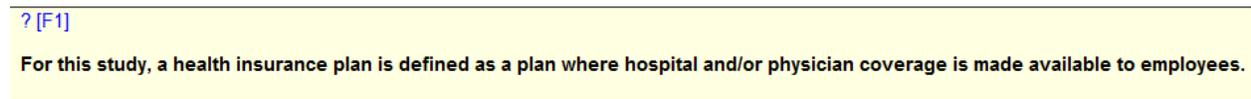


Unlike lookup tables, we do not have a setting to customize the edit check windows display. To provide an indicator for the JAWS user, we are considering using the JAWS dictionary so that if it encounters the phrase “Active Signal” or “Hard Error” JAWS speaks or displays the title of the edit check window plus “JAWS Users Switch to PC Cursor.”

### 4.3 Help Screens

Some survey questions have the option to provide additional information if the respondent requires more help. This is displayed to the interviewer as ? [F1], which alerts them that they can press F1 to open a window with the additional information.

Figure 14 - Display of F1 help available for this screen

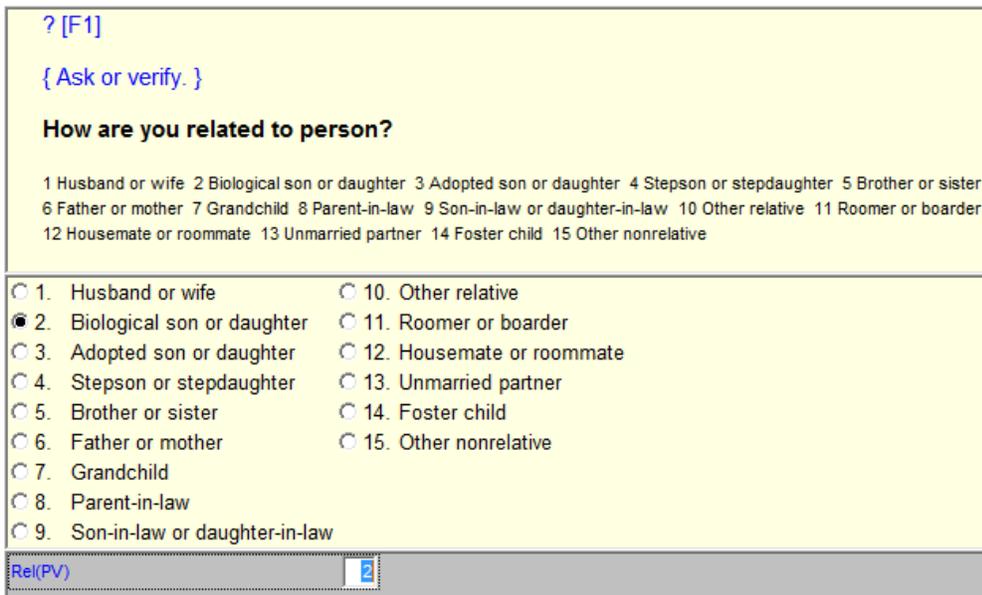


JAWS reads ? [F1] as: **“Question Mark Left Bracket F1 Right Bracket.”** We found that the description read was not ideal. By adding a new entry to the JAWS dictionary, we could translate the above characters, so that when encountered JAWS reads **“Press F1 for Help.”**

### 4.4 Large Response Lists

Some survey questions have a large enough set of responses that it is impossible to put them into one column for JAWS users. One solution that our team made available to help JAWS users is adding a fill to the info pane that lists the answer set in the left to right order that JAWS reads the screen.

Figure 15 – Additional fill for a large response list (for JAWS users only)



Another solution in the early stages of prototyping is to create a lookup table that contains the answer list. This would be useful for very large lists such as one containing the list of languages primarily spoken in the household. However, implementation of this solution requires buy-in from our survey sponsors and interviewers as it would change the display and implementation of the question.

## 4.5 JAWS Settings

We have described the majority of changes to accommodate screen reader compatibility by modifying the Blaise 4 code and redesigning the screen presentation. However, we can also customize JAWS by using screen reader parameters.

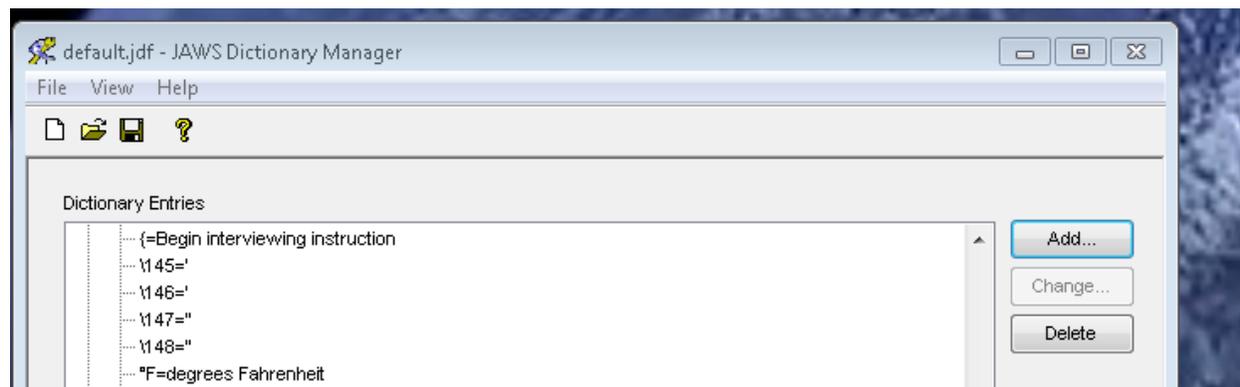
### 4.5.1 Customizing JAWS Settings

During our initial review of JAWS use at our call centers, we learned that interviewers have the capability to personalize their own JAWS settings. These features include text presentation on the screen, the question reading speed if they are using text-to-speech, the voice used for text-to-speech, and the buttons used for navigating their screens.

A few of the solutions that the team came up with are standards for how certain elements such as edit checks are presented to the interviewers and how certain characters or phrases are translated. The JAWS screen reader has its own dictionary feature that can be refined to update what it speaks or displays to the user. By default, when JAWS sees CATI, it says it with a much different inflection than what is commonly used (Cah-Tee). We updated the dictionary so that JAWS uses the expected inflection (Cat-ee).

Besides being able to tweak the pronunciation of words for text to speech, we used the JAWS Dictionary Manager to customize how JAWS reads a particular character, word, or phrase. By default, JAWS reads the braces we would use for interviewer instructions as **“left brace”** or **“right brace.”** We added an entry in the JAWS dictionary so when a **“{“** is encountered JAWS speaks or displays **“Begin interviewing instruction”** or **“End interviewing instruction”** for **“}”**. Not only can we create custom text, we can program JAWS to play a sound if it encounters this character or phrase.

Figure 16 - Translating characters to words



### 4.5.2 Centralizing JAWS Settings

One feature that is offered by JAWS is the capability of having one centralized repository for settings that can be used by all interviewers. By having one common dictionary and other settings, a centralized configuration file can be stored to a common directory on the network. Every time an interviewer starts JAWS, it imports this configuration file for their use. Using this file would ensure consistency of what is said or displayed to the interviewers. This also eases the burden of troubleshooting when issues arise on JAWS settings, since we are not dealing with customized individual settings.

This is still undergoing refinement as we want to be able to capture the settings and dictionary entries the interviewers will definitely need, plus create a baseline that all interviewers can use that would need little to no tweaking.

## **5. Next Steps**

Besides the refinements mentioned in Section 4, we will be adding these enhancements to the instruments on a rolling basis as scheduled. The MEPS and TPOPS surveys currently have the layout changes and clarifying instructions in their instruments. In 2019 we are planning to add the updated interviewer instructions and lookup table enhancements to MEPS and any other approved changes. The team is still meeting regularly to get feedback from our interviewers on the planned changes.

As we continue our research into Blaise 5, we will review our changes to the Blaise 4 instruments and determine how to implement them in Blaise 5. Blaise 5 represents an opportunity to write new screen standards that can take into account the needs of users of JAWS and other hardware or software so that they are able to conduct a survey without a need to overhaul the instrument.

## **6. Conclusion**

Blaise 4 works well with JAWS even if we need to revise some of our screen standards to make our instruments more JAWS-friendly. We have learned several ways to improve our instruments so that anyone who uses JAWS or similar hardware or software will be able to conduct surveys with minimal issues. However, as technology changes it remains important to get feedback from our users at every stage in the development lifecycle. This ensures delivery of the best quality instruments that work with any hardware or software that they will need in order to be able to complete their tasks.

## **7. Acknowledgements**

The authors wish to thank Heidi Casey and the staff of the National Processing Center for their feedback on using JAWS and helping suggest improvements to our instruments.

*The views expressed in this paper are those of the authors and not necessarily those of the U.S. Census Bureau.*



# A Process for Blaise Data Emulation of Complex Instruments

*Todd Flannery, Ed Dolbow and Curtis Cooper, Westat*

As instruments grow in complexity, difficulties arise in testing routes due to the sheer number of combinations of data items and the relationships among those items. This complexity makes it difficult for specification testing to cover every one of thousands of routes through an instrument. It limits the efficiency of prescribed scenario testing, in that only those planned scenarios can be developed and tested, resulting in potential gaps in coverage regarding some novel combinations of data items. Although automated regression testing can identify the effects of changes to an instrument from one test or round of tests to another, it will not account for persistently problematic routes or data combinations. Data emulation of many records can assist in producing a test base that is more comprehensive and thus more representative of possible combinations that can occur in a field study, however unlikely.

Large scale testing involving multiple instruments with many interdependent inputs, or longitudinal data collections with multiple rounds of administration add to the difficulty of ensuring full coverage of the instrument during testing. These testing efforts also require additional coordination among different participants within an organization. Each of these factors increases the difficulty of ensuring full coverage during testing.

This paper will present an approach implemented on a large national longitudinal study to test widely varied routes through a complex Blaise instrument. Westat applied two data generation tools and scripting to a large survey instrument containing 1700 items and more than 2500 unique variables (multiple response and other specify categories). Blaise software offers two utilities to emulate data in an instrument called BTEMULA.EXE and BLEMU.EXE. The application of the emulator tools warrant the use of scripting to address integer variables, date comparisons, hard edits and so forth. The National Standards of Institute and Technology (NIST) in the US offers a software product called ACTS (Automated Combinatorial Testing for Software) to generate data based on combinatorial theory. These generated data were one of three sources of inputs to the emulator along with the variable values produced by the scripting process. Data from previous rounds of data collection and scenarios were the other two sources. The use of inputs, scripting and emulation produced large numbers of randomized data records and provided a method to enable more robust analyses of questionnaire items.

## Theory

Generating large quantities of records with randomized routes through a Blaise instrument can identify errors that may go undetected via more traditional approaches to testing, such as specification testing, scenario testing, and regression testing. The capabilities of the Blaise Emulator Tool, as applied to the test data within an iterative process cover more routes and data combinations than these traditional approaches, ultimately result in increased integrity of instruments, and study data.

## Methodology

Westat applied several methods to preparing the preloads for an instrument. One set of inputs (preloads) were based on responses to the survey in a prior year as well as responses to other instruments during the same administration. The latter category typically involved management variables intended to avoid repetition. For example, several instruments had contact sections at the end that one needed to be answered only once during the administration.

Virtually all of the preloads were derived variables based on a surprising number of questions asked in prior years. These variables were based on specifications approved by the client and extensive programming. The process was time consuming, and the results were not available until later in the development process for the next wave. Nevertheless, these data were a valuable source for the emulation testing and testing performed by the corporate test team.

The second source of data were the variables generated by combinatorial logic of the ACTS software. NIST has performed extensive research on the use of combinatorial methods to uncover system failures. Many system failures are caused by a single factor or a combination of two factors. The interaction effects of three or more variables cause a smaller but significant number of failures. NIST research indicated that single and two-factor testing accounted for 67 to 93% of the failures but that four to six way testing accounted for virtually 100% of failures. Developing test data to meet requirements for interaction testing requires many observations. 10 input variables each with two values generates 1024 test cases if each test case covers a single three-way test. Figure 1 demonstrates how the NIST software uses trios of the data items (the columns) to cover several combination groupings (red, yellow, blue) with each three-way test<sup>1</sup>. Each row can represent a row of input prior to emulating record-level random data.

In this example, the ACTS software requires only 13 records to cover all of the three way interactions. Westat used ACTS software to generate four-way interactions because routes through the instrument required many conditions.

	A	B	C	D	E	F	G	H	I	J
Tests	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	0	0	0	0	1
	1	0	1	1	0	1	0	1	0	0
	1	0	0	0	1	1	1	1	0	0
	0	1	1	0	0	0	1	0	0	1
	0	0	1	0	1	0	1	1	1	0
	1	1	0	1	0	0	1	0	1	0
	0	0	0	1	1	1	0	0	1	1
	0	0	1	1	0	0	1	0	0	1
	0	1	0	1	1	0	0	1	0	0
	1	0	0	0	0	0	0	1	1	1
	0	1	0	0	0	1	1	1	0	1

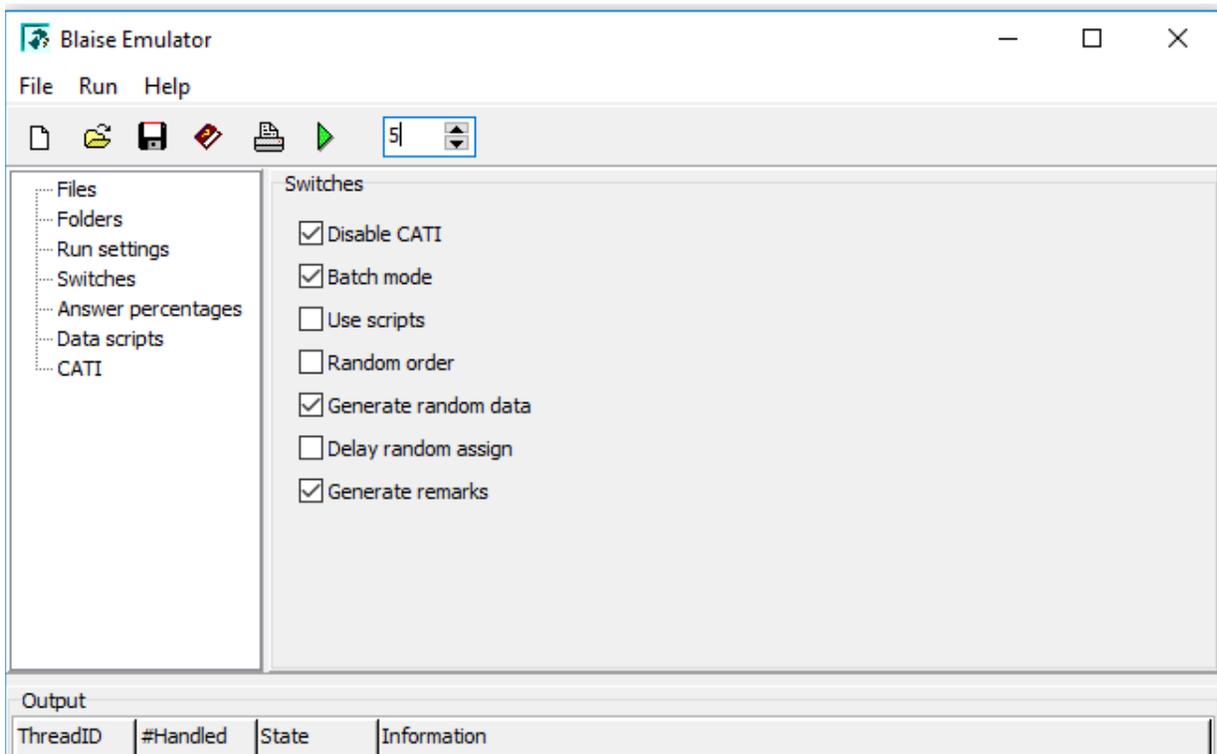
Figure 1 – Multiple tests on one row<sup>1</sup>

There are several advantages of using ACTS data rather than scenarios or longitudinal data. It ensures wide coverage of all possibilities some of which may not have occurred in the first half of the prior way. The original ACTS code took several days to produce but it was modified easily using either the ACTS

interface or edits of the XML file produced by the software. The data can be produced on a timely schedule in less than two days so the method is kind to staffing resources and scheduling during periods of intensive development.

## The Blaise Emulator

Statistics Netherlands provides two tools to emulate data, BTEMULA.EXE and BLEMU.EXE. The BTEMULA program is single-threaded, meaning it handles a single record at a time, and the BLEMU program is multi-threaded. The programs can be used to stress-test CATI instruments, but we were more interested in the capability to generate randomized values for all FIELDS on the route of an instrument. The tools execute via a command-line with an option file or a user interface with settings (Figure 2).

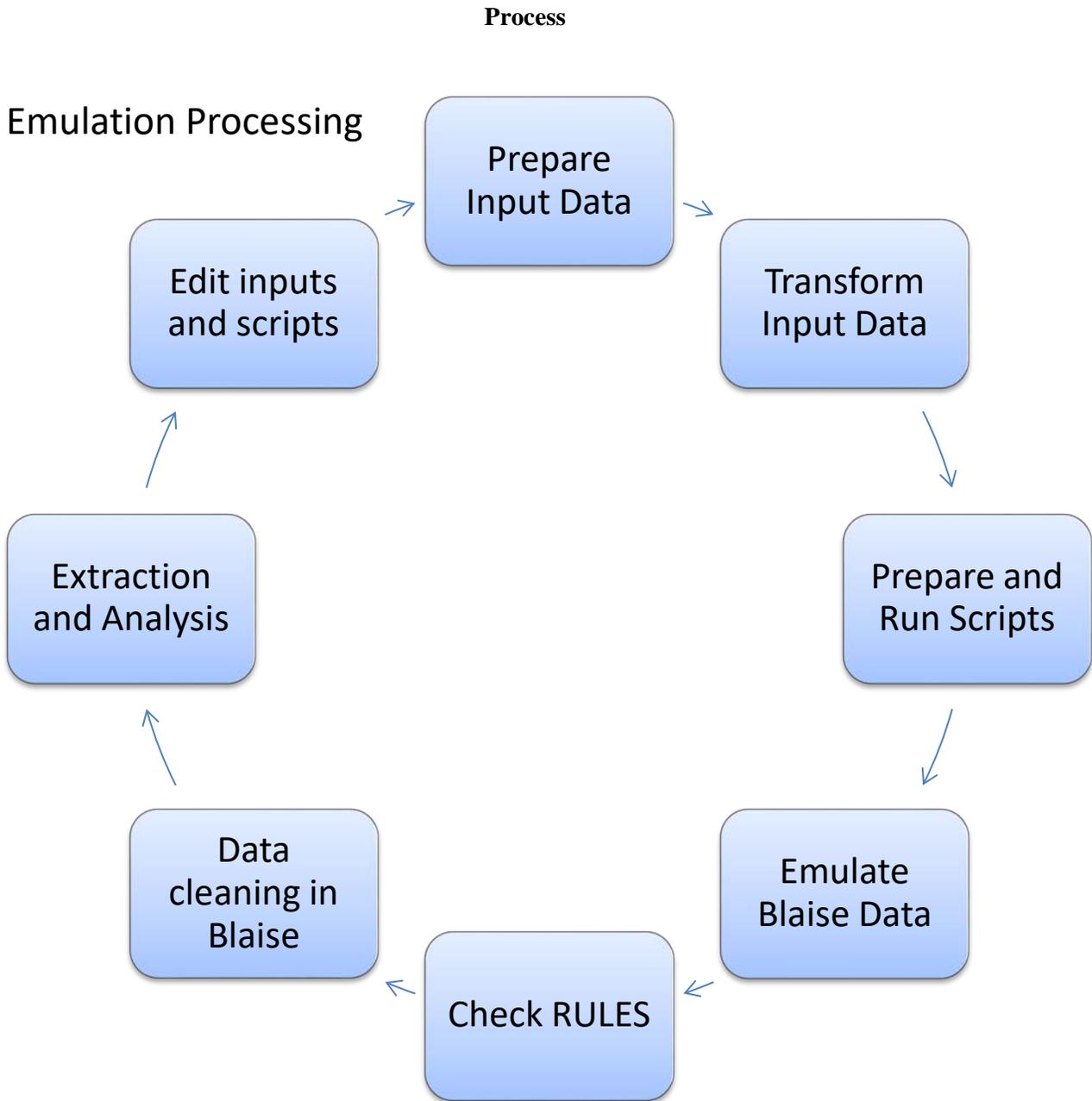


**Figure 2** – Blaise Emulator Tool Interface

As we begin to emulate data for a complex datamodel, some problems arise due to the nature of data and relationships among the data items:

- Inputs are often related to one another, and randomization of values for these inputs does not consider constraints that may exist in the imported data items.
- Randomizing values for certain items, like dates, times, or statuses that are stored as strings will likely create data that is not usable or causes problems with routing in the instrument.
- Eligibility determined by screening questions or sampling algorithms may result in few or no eligible respondents, and this will limit the effectiveness of randomization on most routes.
- Values that are created in randomization are based on the *definition of the FIELD*. The emulator does not use hard or soft edits (CHECKs and SIGNALs) in determining the random values. This can result in setting many outrageous values for FIELDS defined with wide ranges of integers.

We have tools that mitigate each of these issues and produce large-scale data that can provide valuable information regarding the integrity of our code or specifications. To begin, we isolate any inputs to the randomization process from the emulator, since the emulator tool does not overwrite existing data with emulated values. This will control and enforce data integrity for those inputs. We later discuss some methods for creating values in the input data. Additionally, we can avoid some of these errors in emulated values by scripting particular combinations of fields and prescribed values. We apply these tools as steps in the iterative emulation process.



**Figure 3** – Processing steps for emulation of data

## Preparing the input data

Inputs can be comprised of any data from external sources, such as management data (statuses or demographic characteristics of a respondent), sampling rates or flags, or data collected and passed in from other Blaise instruments. Sources of the input data can be:

- Derived via a combinatorial approach. The NIST combinatorial approach allows us to apply constraints to the input data in order to maintain relational integrity among the input items.
- Imported using Blaise data via prescribed scenarios – a planned set of testing inputs are preloaded into the database via Manipula.
- Imported using existing Blaise data from a prior data collection – prior wave data are preloaded into the database via Manipula.

The data preparation involves creating, in Blaise, any field values external to the instrument that could affect the routes. Next, we copied these records over several times to associate multiple routes through the instrument with each set of inputs. For each instrument, we decided to create about 15 thousand records to pass through the emulation process.

## Transforming the input data

The data transformation involves a simple Manipula import to bring our input into the existing datamodels that we are testing via emulation. At this point, we do not need to check the rules because data integrity is assumed intact when the input values are imported.

## Preparing and running scripts

Scripting populates the instrument with additional values to provide constraints and limit outlandish values. To minimize the impact of datamodel changes while an instrument is being used to collect data, we intentionally allow broad ranges for continuous values like age. This ensures that we can maintain compatibility with existing data in the case that we decide to update our ranges. Scripting limits these broad ranges to a discrete number of meaningful values. We often use hard or soft edits to handle errors in data collection. In emulating data via BTEmula or BLEmu, these edits are ignored, and we can sometimes end up with some values that would not be allowed via the DEP so we apply scripts, by section. Applying scripts by instrument section allows us to assign discrete values that avoid unusual data that can arise due to broad definitions. This enforces data integrity that the emulator tool cannot achieve without using the scripts. Designing scripts can be an intensive process, based on the complexity of edit checks, so we use experts who know the instruments to determine the discrete values that would affect routing for critical and/or integer fields.

In a sense, the scripting process may belie our stated goal of producing great variation in the data, and some prudence should be exercised in the development of scripts. We use scripts to limit particular data items to exclusive values or combinations of values, excluding these values from being randomly selected by the emulator. Although the selection of any particular script among many is randomized, the values within the script will be assigned to the data. Given this, we recommend keeping the number of FIELDS referenced in a single script limited to a few. Note that we create some scripts to increase *access* to routes that we know could be disproportionately skipped if randomization alone were used. Additionally, scripting can prevent range errors that may skew the data analysis from emulated output.

Once the scripts have been created, we can use the emulator tool to apply them to the data, as is described in the Blaise help for emulation. In our case, we determined that scripts were required for each section of

the instrument and, since we already had some of the existing data in place, we did not want scripting to overwrite those values. So we handled scripting via a Manipula batch process, and this allowed us to have greater control over how the scripts were processed into data values. Our process randomly selects a single script per section and only writes the scripted data where no data currently exists in the instrument for those values. Note that individual scripts can contain logically inconsistent data, as the data will be cleaned in a later stage of processing.

## Emulating everything else

After the scripts have been applied to the now mostly-empty data set, we can run the emulation step via BTEmula or BLEmu. Emulating all of the remaining data in the datamodel can produce some inconsistencies and/or errors that may not occur in the real world. In emulation, all possible values are equally likely, and this can produce some unexpected results. Here are some considerations:

- Continuous problem – as described above, defining broad ranges produce many outrageous results (e.g., the 500-year-old man or someone who smokes 900 cigarettes each day.)
- Strings can take on any values depending on their length, and results may not be useful.
- Randomized dates can lead to date combinations that make little sense or generate errors.
- Sampling algorithms or screener questions can produce few or no results (or respondents) due to randomizing field values that tend to almost always have a positive result in the real world (e.g., enumeration for “Does the respondent consent?” having values “1=yes, 2=not now, 3=soft refusal, 4=hard refusal”) will result in a lot of missing data.

Through successive iterations of the process, we can address issues that arise via creating scripts that will address these data issues individually.

## Checking the RULES

We ensure clean routes by running a Manipula script with these settings:

```
CHECKRULES = YES  
DYNAMICROUTING = YES
```

At this point, we can separate records where hard or soft errors are found. Trapping these errors informs future iterations of scripting.

## “Cleaning” the data

Prior to extracting the data from Blaise, we need to clean some values created as a result of randomization in the emulator. This process requires another Manipula program that uses recursion to parse both the datamodel and the data values. This customization is largely dependent on project needs. Some tasks:

- Cleaning up strings to use a single value, like “ZZZ”, since these are not often useful for analysis and don’t affect skips in the instrument.
- Setting limits on SET-type responses to accept only 1 or 2 selections, if preferred.

- Iterating through pairs (or trios, etc.) of values that are producing hard or soft edits.
- Removing values where the enumerated response text was intentionally blanked-out.

We can use results from the cleaning process to inform scripting for a subsequent processing cycle.

## **Extracting and analyzing the data**

For our project, Westat converted the data to SAS for analysis. In the SAS analysis, our project and clients define populations and routes that have known probabilities. The emulated values are compared to the expected values to determine if routes are being missed, administered improperly or expected ranges in the data are exceeded. In this way, we can determine if an error exists, and then once determined, we can investigate whether these errors are related to specifications or coding.

## **Editing inputs and scripts**

Instrument experts, Blaise systems staff, and data experts use the results of this analysis to:

- Identify problems in the routing or specifications.
- Determine how to construct or edit inputs and scripts that will allow for the most value in terms of coverage of routes in the data.

These edited inputs and scripts are then applied to our edited datamodel as we begin the emulation process with a new set of test records.

## **Results**

Our first pass through the emulation process utilized an approach where an expert tester entered scenarios, and the remaining data was emulated. This produced about 15,000 test records, each containing 4000 columns of Blaise data. The iteration took about 3 working days, with processing steps for building and running scripts and the single pass through emulator program requiring the most. As a result of the SAS analysis, we were able to identify over one hundred areas of interest for further investigation within the instrument. Since our focus is in improving performance of the process through successive iterations, it is likely that this first pass will help most in determining where emulation scripting and cleaning can be improved.

Our second pass through the process utilized the longitudinal approach, using collected data from an earlier wave of data collection to populate the inputs and perhaps produce results likely to be encountered in an upcoming field collection. We generated 12,500 records, and the analysis again provided many results for further investigation.

We also were able to run the NIST ACTS tool to generate two and four-way combination records. The records were imported as inputs to our data, and we generated x records for analysis. For the 2 factor test, ACTS generated 85 records covering tens of thousands of combinations. For the 4 factor test, ACTS generated 4717 records covering tens of millions of combinations. The software is very efficient and produced even the 4 way combination in minutes.

## Considerations

We needed to make adjustments regarding the scripting and editing some emulated values. Scripting for data where values already exist will overwrite the existing value with the scripted value, and we preferred the pre-existing values to take precedence over the scripts. In addition, the scripts needed to be in the same folder as the emulated data, and our number of scripts per section was getting burdensome enough for us to use a batch process with a Manipula program to be able to have more control over the scripting process. Our customized process randomly selects a script from each scripting folder to write its contents to the active record. This allows us to script more data in a single run through all of the emulated records.

Emulation via the multi-threaded BLEMU.exe or the single-threaded BLEMULA.exe allows values to be emulated where response text has been suppressed. In coding, programmers sometimes pass blank strings as response text in order to conditionally hide or disallow an inconsistent response. The emulation tool ignores these restrictions and may use the value in the emulated data, so we reset these values to non-response when we come across them, as part of our data cleaning.

## Recommendations

The ability to assign weights to scripts or emulated values could potentially provide better coverage of targeted combinations or groupings. As it stands, each script or value is treated equally, and this does allow for variety covering a full range of responses. It is possible to mimic the behavior of weighting scripts by creating many scripts with preferred data. However, it would be better to assign weights to particular scripts to allow a higher prevalence of that combination. The weighted scripts could be used to produce data more in line with known prevalent trends.

Using experts with different backgrounds improves how the scripts or input data can be constructed between cycles of emulation. We need to restrict the inputs or scripts to provide meaningful results, but we also seek to produce a multitude of varied data in order to maximize coverage of the routes through the large-scale method of data emulation. Instrument experts have intimate knowledge of the desired combinations that provide insight regarding prevalent routes or key fields that determine crucial skips in an instrument, whereas data experts use the analysis of the emulated data to develop frequencies that detect route errors.

We can further adapt the process of emulating random data for special situations by using Manipula features to handle some of the exceptions we noted. Being able to access error information will allow us to re-randomize when a hard or soft edit has been raised, and we can determine if responses are excluded conditionally to reduce some of the emulation-specific errors that we discover in the analysis stage. Additionally, developing Manipula processing could allow us to adapt the process for Blaise 5 instruments without having to convert between Blaise versions.

## Conclusions

Applying a process for emulation of data allows greater coverage of combinations than standard scenario testing or data entry testing. Although the preparation of scripts, processing and cleaning of individually emulated test records can be initially intensive and time-consuming, the iterative nature of the process and subsequent refinement of inputs for cycles of processing can provide high-quality test data for longitudinal or cross-sectional instruments. The more burdensome tasks associated with data entry testing can be alleviated via the emulation process, allowing more time for testing staff to focus on targeted scenarios. Additionally, since emulation can begin as soon as a datamodel is prepared, the data can be

analyzed while testing is taking place, and this reduced “lag time” between data entry and analysis eliminates duplication of error reporting that occurs due to a lengthy test cycle followed by an analysis cycle.

## **References**

D. Richard Kuhn, Raghu N. Kacker, Yu Lei. “Practical Combinatorial Testing” NIST Special Publications 800-142, October, 2010



# An approach to unit testing

Rod Furey, Statistics Netherlands

It is often mooted that it would be nice to be able to unit test various bits and pieces of a datamodel. One obvious section that can be tested stand-alone is a block. The concept presented here suggests selecting a block, extracting it (in various ways), defining a controller for the block and checking various things inside the block.

## Test IDE

The basic Test IDE is based on various items that have been assembled over time and which have subsequently been collected in one place. The main concepts are as follows:

- display the asked fields in the instrument
- check the route by running the rules on a datarecord
- check for any errors that may occur
- extract the block and create the block controller
- perform the items 1, 2 and 3 on the extracted block
- if necessary, correct the block and re-prepare the instrument
- compare the results of the various runs
- repeat as necessary

The first 3 items are well-known and have been presented before.

The screenshot shows a Test IDE interface with a flowchart on the left and a data table on the right. The flowchart consists of four boxes: 'MemNo (1)', 'Forename (2)', 'Surname (3)', and 'PreferredMode'. Arrows indicate a flow from 'MemNo (1)' to 'Forename (2)', and from 'Forename (2)' to 'Surname (3)'. The 'PreferredMode' box is positioned below the others but has no incoming or outgoing arrows.

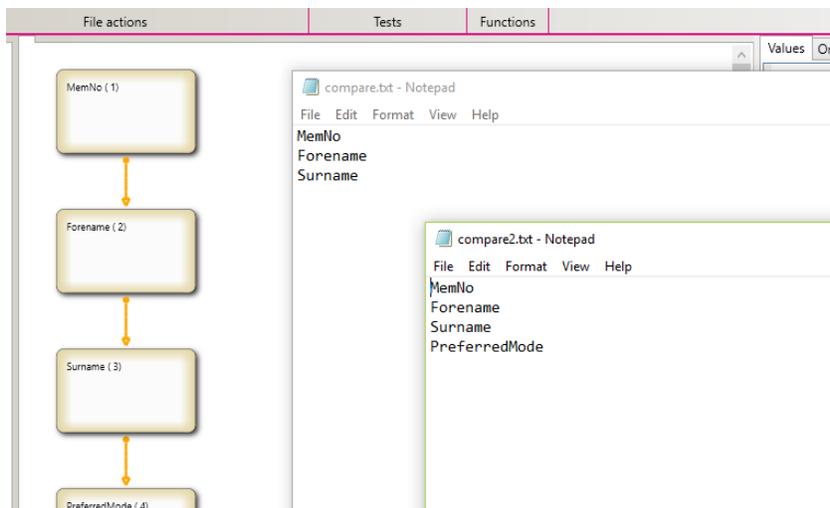
The data table on the right has the following columns: VarName, StartingValue, CurrentValue, and VarType. The data is as follows:

VarName	StartingValue	CurrentValue	VarType
DoAge.Adult			ENUMERATIO
DoAge.Age			INTEGER
DoAge.DOB			DATE
DoAge.Junior			ENUMERATIO
DoAge.Senior			ENUMERATIO
Forename	fred	fred	STRING[255]
MemNo	1234	1234	INTEGER
PreferredMode			ENUMERATIO
Surname			STRING[255]
WhichEvents			SET

At the bottom of the interface, there is a table with columns: VarName, ErrorType, ErrMsg, Surname, Route, EmptyButRequired.

Extraction of the block can be accomplished in a number of ways, depending on how the test IDE suite is constructed (e.g. directly from the sources or reconstructed out of the .bmix). The basic controller for the block doesn't have to be complicated and can be constructed by analyzing the source or retrieving FIELD information out of the .bmix. The problem with constructing the controller is working out what dependencies the block may have (e.g. ALIENS, references to things outside the block, PARAMETERS etc.) Dependency determination can be accomplished in a number of ways depending on what information is available to the programmer.

Once the controller has been created and compiled, the results of a test run can be checked in any number of ways, most usually by checking the results of executing the rules on a data record. Comparison of results can be achieved reasonably easily by marching the route and writing the output to a text-file which can then be fed into a text-file comparison program. In this way the output can be compared against a previous run or against an expected result. Again, if things don't match up, the analysis and repair steps can be repeated.



## Data generation

Instead of creating the test datasets by hand, it is possible to generate them programmatically. There are various approaches to this, from randomly generating the data values for each field through supplying data for each field on the various routepaths to analyzing the flow of the source code and supplying values that exercise each route through the program code. Some of these possibilities are easier to program than others.

Once the datasets have been generated, they can be stored away and re-used at a later date. If a dataset is not valid any more, it will either produce an error when value assignment takes place (e.g. value out of range) or an error will have to be trapped when trying to assign to a field that no longer exists.

## **Conclusion**

Creating the advanced items that facilitate unit testing (e.g. exercising every path through the code) costs time. Creating the basic IDE and supporting routines is relatively cheap and can deliver useful results relatively quickly whilst the more expensive items are analyzed and programmed.



# Experiences with Blaise 5 CATI and multimode at Statistics Norway

*Jan Haslund and Trond Båshus, Statistics Norway*

## 1 Introduction

Statistics Norway is the national statistical institute of Norway and the main producer of official statistics. We are responsible for collecting, producing and communicating statistics related to the economy, population and society at national, regional and local levels. Statistics Norway has collected data for social surveys for over 50 years. We started using Blaise in social surveys in 1991, and are currently working on a transition from using a mixture of Blaise 4.8 and Blaise 5 to Blaise 5 only.

## 2 Current situation

Statistics Norway is using Blaise 4.8 for CATI (online) and CAPI (offline), and has used Blaise 5 since 2014 for CAWI. While Blaise 4.8 is tightly integrated with our case management system (SIV), there is only a very basic integration with Blaise 5. The system allows for simultaneous data collection on CATI and CAWI. If a respondent starts a questionnaire in CAWI-mode (Blaise 5), the corresponding case will be excluded from daybatch in Blaise 4.8. This is handled by SIV. While this provides us with a very flexible framework to conduct multi-mode surveys, it is very labour intensive. The surveys must be prepared twice and data is stored in two separate databases. Mature support for CATI in Blaise 5 will make it possible for us to save time and reduce costs. A more sophisticated synchronisation of events and data between SIV and Blaise 5 will have to be developed before we can fully replace Blaise 4.8.

## 3 Transition to CATI in Blaise 5.x

### 3.1 Why

While Blaise 4.8 works very well for CATI and CAPI, it hasn't been particularly usable for CAWI for quite some time, especially when we consider the small screens of mobile devices. We decided to switch to Blaise 5 for CAWI as soon as we perceived Blaise 5 was mature enough for this purpose. Having all modes in Blaise 5 will make it less time consuming to create and maintain multimode surveys, and will obviously make the data processing simpler. We judged that CATI functionally to be complete enough during the spring 2018.

### 3.2 How

To implement Blaise 5 for all modes, we decided to use real surveys: Governing and Experiencing Citizenship in Multicultural Scandinavia (GOVCIT) and a pilot survey for the Labour Force Survey (LFS). The LFS pilot was to see if we could use CAWI in the Labour Force Survey for some or all the respondents, from the second to eight waves. The framework for CATI, such as the appointment and dial questionnaires, were based on the CATI sample provided with Blaise.

### 3.3 GOVCIT

The first survey out was GOVCIT. The survey has a sample of almost 8000 people aged 20-36 years. 3000 have immigrated after the age of 16 years. 3600 have immigrated before the age of 11, or were born in Norway with parents who have immigrated. The rest is the control group where most people are born in Norway. The questionnaire is in seven languages (Norwegian, Somali, Arabic, Polish, Turkish, English and Urdu). The questionnaire has quite simple routing and took about 15 minutes for the respondent to complete. It has a few mode specific questions (different text for CAWI and CATI). We made a specific layout for smartphones and one for laptops and tablets see illustrations below. The surveys mode design was to first conduct CAWI for two weeks and then have a follow up on CATI and CAWI. Since we knew that most of the respondents would complete the survey on CAWI, it was less risky to follow up on CATI

in case something technical went wrong. At the time of follow up on CATI, 2265 respondents had answered on CAWI. CATI and CAWI follow up resulted in 1065 additional completed questionnaires. Some of these respondents had already started the web questionnaire, but not completed it. SMS and email reminders were sent to the respondents at the same time as the interviewers would call them, and gave them a choice to either wait for the interviewer to call them or to do it themselves on web. We also made a button in the CATI-questionnaire which the interviewer could use to send an e-mail to the respondent with a link to the CAWI questionnaire including username and password.

### **3.3.1 What went wrong on CATI?**

The first issue we experienced was that the select field include/exclude functionality did not work properly for string fields. This was reported to the Blaise developers, but it couldn't be fixed in time for the survey. We decided to mark all cases we wanted to exclude from the daybatch as completed instead.

The first two weeks with only web questionnaire everything worked fine. The exciting part started when we started CATI. We have our production data in a MySQL database. We have tested on our laptops with data in a .dbx file. And it seems to work as expected. When the interviewers started to call and interview, we soon experienced that there was a major problem with the survey. The dial questionnaire came up and the interviewers phoned, but when they tried to start the questionnaire they just came to the dial menu for the next respondent. This was not for all respondents, but approximately 80 to 90 per cent. We had to stop the CATI interviewing and find out what to do.

### **3.3.2 Emergency solutions**

To salvage to the rest of the data collection period, it was decided to use Blaise 4.8 for the CATI part of the follow up. The new CATI-questionnaire was very simple with only a couple of questions and contained a link to the web questionnaire including the respondents' username and password, The interviewer could cut and paste the link to the browser and reach the GOVCIT CAWI questionnaire . The interviewer completed the interview in Blaise 5 in a browser and at the end switched back again to Blaise 4.8 to answer a question if it was a new questionnaire, an already started questionnaire or an appointment. Every morning we ran some Manipula programs to synchronize new CAWI interviews to the CATI questionnaire in Blaise 4.8 so that we could exclude them from the daybatch.

Doing it this way was additional work, but questionnaire data were stored in one single database and the interviewers could use Blaise 5 and see how it worked, although they didn't use the CATI layout or use the t CATI specific layout. Even though this first attempt at CATI interviewing in Blaise 5 was a failure, we gained valuable experience in creating a multimode survey in Blaise, including setting up mode specific questions and layout, setting up the CATI specification and learning about the CATI dashboard. We also learned how the new dial and appointment questionnaires worked, which is a major change compared to Blaise 4.8.

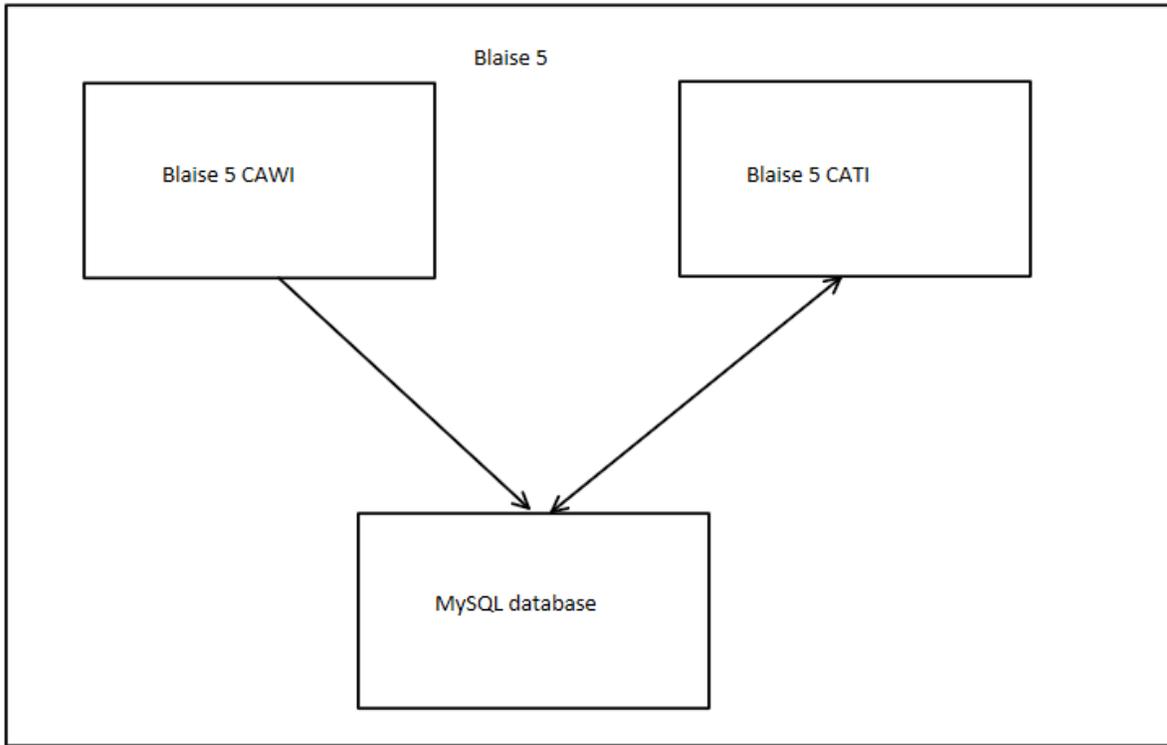


Figure 1 Masterplan for GOVCIT SIV??

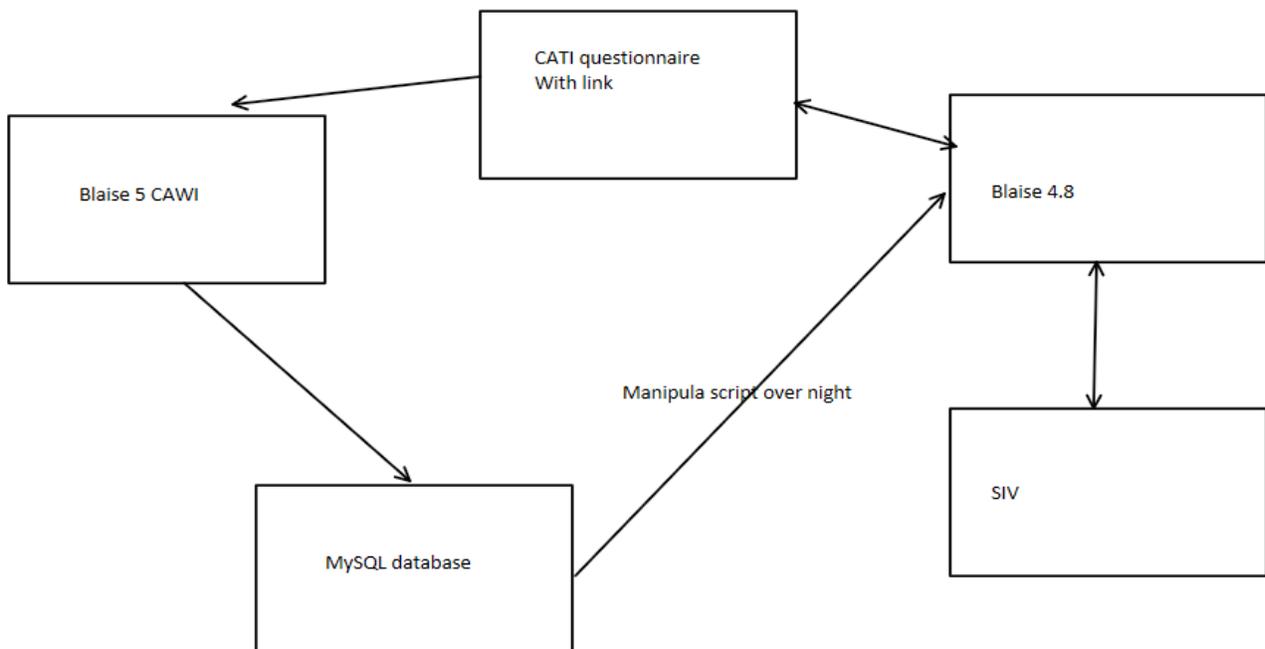


Figure 2 Emergency solutions for GOVCIT

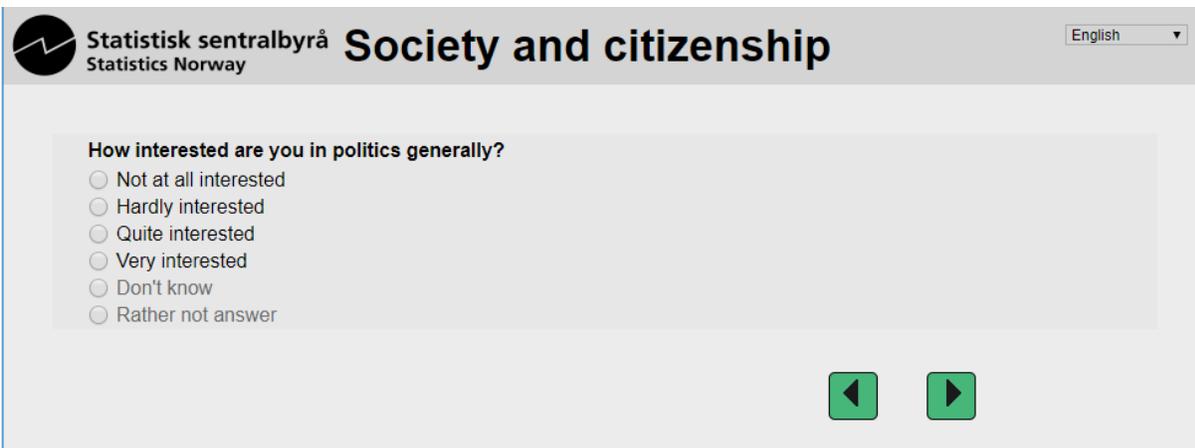


Figure 3 GOVCIT CAWI laptop, tablets

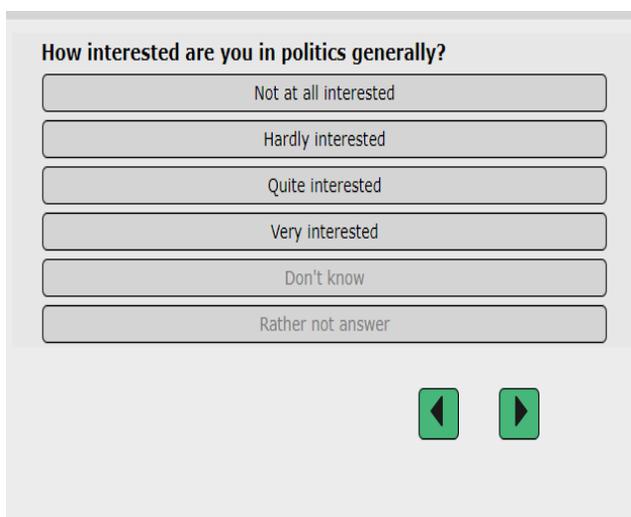


Figure 4 GOVCIT CAWI smartphone



Figure 5 GOVCIT CATI

### 3.4 Labour Force Survey (LFS) pilot

The labour force survey (LFS) is currently CATI only, but a pilot to develop a more flexible LFS is underway. A multimode Blaise 5 questionnaire has been developed, and the project will also participate in developing a more sophisticated integration between Blaise and SIV.

It was decided to convert and adapt the existing LFS Blaise 4.8 questionnaire to Blaise 5, rather than writing a new tailor made multimode questionnaire. The pilot could have been a good opportunity to create a less complex questionnaire. But it was decided that a conversion should be attempted because of time constraints, and because it would make comparison to the ordinary LFS easier. The conversion of the Blaise 4.8 code went well, and only minor changes were necessary to make it work in Blaise 5. Most of the remaining work was to make CAWI layouts for desktop and mobile, and further developments of CATI-solutions developed for GOVCIT.

#### 3.4.1 Developments after GOVCIT

The CATI interface was updated slightly based upon feedback from interviewers. Changes includes the possibility to return to the dial from the appointment form and similar functionality in the main questionnaire.

Some screenshots follow below.

Ringebilde

Statistisk sentralbyrå Statistics Norway **Ringebilde** Norsk

IO\_nummer: 10001

Telefon 1:

Telefon 2:

Telefon 3:

Avtale til en bestemt tid  Foretrekker et bestemt tidsrom  
 Avtale til en bestemt tid  Foretrekker noen dager i uka

Bokommune: 0101

Adresse:

Postnummer: 1767

Poststed: HALDEN

E-post:

Kontaktperiode:

Periodenummer: 24

IO tilhører delutvalg: 1

Brukerkode:

Passord:

Intervjuer/gruppe: tth

Meldinger til intervjuer:

Når kan vi ringe deg tilbake?  
 Ikke en bestemt tid  Foretrekker et bestemt tidsrom  
 Avtale til en bestemt tid  Foretrekker noen dager i uka

Avtale startdato:

Avtale starttid:

Avtale slutt dato:

Avtale sluttid:

Valgte ukedager:  
 Mandag  Onsdag  Fredag  
 Tirsdag  Torsdag  Lørdag

Melding om avtalen:

Ringemeny:  Start skjema  Avtale  
 Ikke svar

Figure 6 Dialform

Lag en avtale

Statistisk sentralbyrå Statistics Norway **Lag en avtale** Norsk

Du skal nå lage en avtale.

Når kan vi ringe deg tilbake?  
 Ikke en bestemt tid  Foretrekker et bestemt tidsrom  
 Avtale til en bestemt tid  Foretrekker noen dager i uka

Spørsmål: Appointment.AppointInfo.AppointType

Figure 7 Appointment form

Figure 8 Main questionnaire

### 3.4.2 What went wrong?

Our experience is that some problems first become obvious when a survey has started in the field, mainly because volume of test cases are too small when a questionnaire undergoes testing. When the first round of the LFS-pilot was put in production, it soon became apparent that something was amiss as the information in the CATI Dashboard showed too few completed cases. After some investigation it turned out that information was missing from the RegCalls block, and that many cases did not get a completed status.

### 3.4.3 Emergency solutions

It turned out that the system worked fine despite the error, in that completed cases were not delivered again by the scheduler the same day. Since we had the foresight to make our own finished variable, we could use this to exclude cases from the daybatch with select fields. The bug in Blaise has been fixed (in Blaise 5.4.4).

### 3.4.4 Future incremental updates

For the future waves we have planned updates, which includes more relevant information in dial questionnaire (information from previous waves) and minor usability changes after feedback from the CATI-interviewers. Better support for keyboard navigation in the latest version of Blaise 5 also helps.

## 4 Integration with our case management system (SIV)

### 4.1 4.8 solution

Our current data collection system is based primarily on Blaise 4.8 for CATI and CAPI, and SIV as our case management system. Blaise 5 is used for CAWI (excluding the two surveys mentioned in this paper), and is very loosely integrated with our systems. SIV is continually updated with events from Blaise 4.8,

and there is also a two-way link to update SIV or Blaise with changes in address and telephone numbers. From Blaise 5 SIV only receives events such as start session events and end questionnaire events, which are sufficient for CAWI.

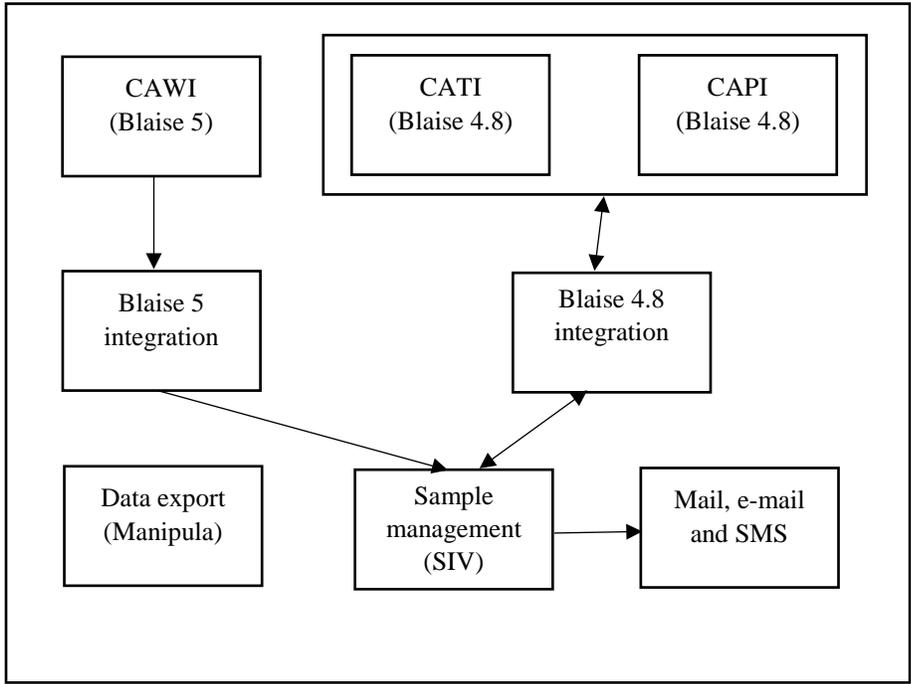


Figure 9 Blaise 4.8 og SIV (old)

#### 4.2 5.x solution

The data collection system we are working towards will be without Blaise 4.8, negating the need to prepare the same questionnaire twice, which hopefully will save us some time, and make the system more robust. The synchronization mechanism will be totally rewritten, and this work is currently underway. We expect that we within six months will have a tightly integrated multimode data collection system, where it's possible to run all CAWI and CATI questionnaires in Blaise 5.

It is also planned to extend the integration between Blaise 5 and SIV significantly, compared to Blaise 4.8 and SIV. We wish SIV to read appointment information from Blaise, so that this information is available in SIV. In Blaise 4.8, information about all contact attempts were stored in the Blaise database itself. In the future we wish to store this information in SIV instead, which will make this valuable information much more accessible.

We are still considering whether our future CAPI solution should be online or offline, or if it should run on a PC or for example a pad. Until we have a satisfactory solution for CAPI in Blaise 5, we will run Blaise 4.8 in parallel. The most likely outcome is online CAPI, either using a PC or tablet, in contrast to our offline CAPI system of today.

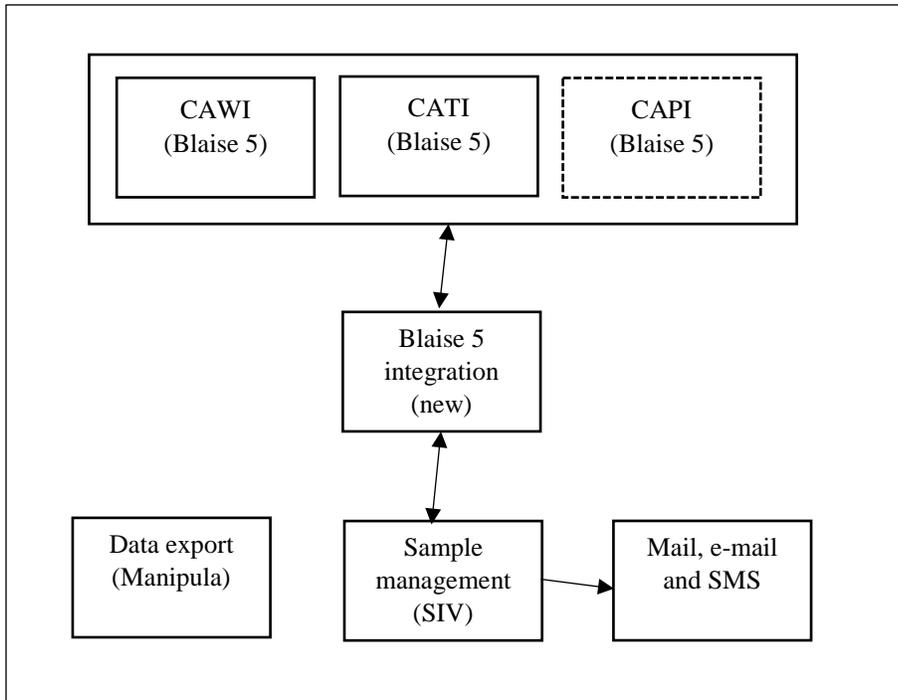


Figure 10 Blaise 5 and SIV (new)

## 5 Conclusion

With the work we have done on multimode in Blaise 5 in relation to the GOVCIT survey and the LFS, it has become apparent that Blaise 5 now has reached a level of feature completeness which makes it possible to replace Blaise 4.8 completely. The main obstacles to make this possible are dependent on internal development work at Statistics Norway. Although we have encountered errors in the Blaise software during the implementation, we feel that our CATI experience has been a success. The errors have been reported back and fixed, and sometimes real-world usage is the only way to iron out glitches in software as complex as Blaise.



# Implementing a Blaise 5 Mixed-Mode Solution

*David Kinnear, Office for National Statistics UK*

## 1. Introduction

The Office for National Statistics (ONS) has been conducting the Opinions survey for several years as a face-to-face interview. It is a regular omnibus survey conducted on behalf of sponsoring organisations, who use it as a vehicle to obtain opinions on topics of interest. A decision was made to move the survey to a mixed-mode CAWI (Computer Assisted Web Interviewing) and CATI (Computer Assisted Telephone Interviewing) survey, and a phased approach was planned to achieve this. In October and November 2017, a CATI pilot exercise was conducted and a fully mixed-mode CATI/CAWI survey was then carried out in February and March 2018.

Resource pressures meant that the existing telephone interviewing systems using Blaise 4.8 would be used, as opposed to using the Blaise 5 call scheduler. Regarding the web element, the current corporate position is that the electronic questionnaire tool (eQ) which the ONS is in the process of developing should be used for web survey. However, delays in the rollout of the eQ tool meant that an alternative solution was required to meet the February deadline. The decision was made to use Blaise 5 for the web-collection instrument.

## 2. Conducting the survey

### 2.1 Methodology

The Opinions survey sample consisted of 2010 household cases, drawn from the fifth wave of the Labour Force Survey (LFS) and based on age and gender. A named adult from each of the household cases was identified, and this person had to be the individual who completed the survey.

One issue that was identified early on was the treatment of personal data brought forward from the LFS. The telephone survey would continue to reference previous LFS responses provided by individuals, and the interviewer would ensure as far as practicable that the correct respondent was being spoken to. The web survey required a new approach, due to concerns regarding personal data security and who had access to responses previously collected as part of the LFS.

The approach taken for the web survey was to display the name of the sampled person invited to complete the survey. A statement was included in the questionnaire requesting whoever was completing the form to confirm that they were indeed the sampled person. No previously collected data from the LFS was pre-loaded into the web survey database, avoiding any data confidentiality issues.

### 2.2 The collection period

There was a three-week survey period; for the first week respondents were invited to complete the survey online. Postal reminders were sent towards the end of the first week to try and maximise online response. If they had not completed the survey online after the first week, then the second week would see the interviewers phoning respondents. The online questionnaire would then become unavailable for the third week, when all interviewing was conducted by phone.

### **3. Technical solution**

#### **3.1 The web survey**

Blaise 5 was the only serious contender for conducting the web survey. The ONS has used Blaise for many years, and its staff have become proficient in coding survey questionnaires using the software. However, the layout template designer was a new concept that we were using for the first time on a live questionnaire. The biggest challenge was to create effective layout templates in the time available, particularly as staff capability in using it was still being developed.

The web survey was being hosted by colleagues at the Northern Ireland Statistics and Research Agency (NISRA). Due to Information Assurance concerns, the ONS did not have direct access to the Blaise server. Whilst not ideal, NISRA implemented a workable solution which proved effective for the small-scale pilot. A scheduled task would copy the Blaise web data from the server to a secure ftp site, from which the ONS could retrieve the data to be incorporated into its systems.

The Opinions survey database pre-populated with the sample was installed on the NISRA server, with each case having a unique identifier. An accompanying login database, with the same case identifiers and their unique respondent access codes was also installed. The access codes had previously been mailed to the sampled individuals.

A link to the Opinions survey was provided on the ONS website, from which respondents would be redirected to the questionnaire hosted by NISRA. The Opinions questionnaire then called the login questionnaire. On the successful input of the user access code that had previously been sent to the respondent, the corresponding case would then be retrieved from the pre-filled Opinions database for completion.

#### **3.2 The telephone survey**

The ONS Telephone Operations (TO) currently uses the Blaise 4.8 call scheduling system. The Opinions survey had previously been conducted in the TO as part of the transition in moving the survey from a face-to-face to telephone exercise. Time constraints in the mixed-mode project meant that the only practical option was to continue using Blaise 4.8, rather than utilising the Blaise 5 call scheduler.

The web element of the mixed-mode pilot would increase the number of respondents contacting the Survey Enquiry Line (SEL) team in the TO. The SEL deals with any number of calls from respondents, ranging from queries, complaints, refusals to arranging appointments. For a previous unrelated pilot SEL operatives would record outcomes of respondent calls relating to the web survey in a Blaise 4.8 instrument.

For the Opinions mixed-mode the decision was made to incorporate the SEL instrument into the main telephone survey questionnaire, ensuring that the TO Blaise database would always have the latest case information available to the scheduler. This would include, for example, those respondents contacting SEL after receiving the advanced letter to confirm their refusal to take part.

### **3.3 Bringing the modes together**

One of the more significant challenges was ensuring the CATI database was consistent with the latest responses from the web. Hosting each mode on separate databases in different organisations was always going to introduce a time lag between the consistency of the web and phone data.

On opening the web questionnaire, after reaching a specific point in the questionnaire a status field is set to partially completed. At the end of the questionnaire the status is set to completed. If the respondent breaks off from the web questionnaire without completing it, the status will remain as partially completed. This ensured that the case would be available for telephone interviewing if not subsequently completed by the respondent on the web.

The web data collected in Blaise 5 was downloaded from the NISRA web server, and then converted using Manipula to XML format. To enable the conversion from XML to the Blaise 4 CATI database, the Blaise5to4Source.exe was used to create a Blaise 4 structure of the web questionnaire.

A Blaise 4 Manipula script was then used to copy the XML data into the Blaise 4 CATI database. All web cases, irrespective of whether they were completed or partially completed, were copied. This was done at regular intervals during the day, reducing the risk of completed web cases being selected by the call scheduler.

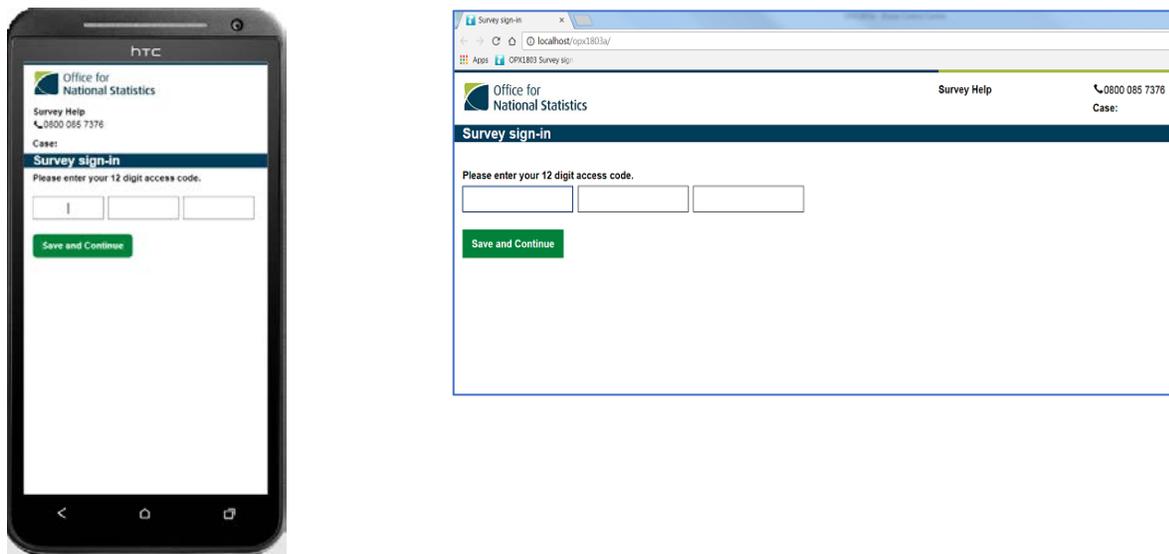
As previously mentioned the LFS data was not pre-loaded into the web database. Consequently, the Manipula transfer of web data to the CATI database would empty the pre-filled LFS data in that database for the relevant case. The reasoning for this was that any data collected on the web would be timelier than data collected in the last LFS wave.

### **3.4 Designing the web questionnaire**

We decided to separate authentication into a separate sign-in questionnaire. On successful validation of the 12-digit access code previously sent to the respondent, the Opinions questionnaire was then launched.

The screen design of the Blaise 5 web pages was heavily influenced by the look and feel of the standards employed by the corporate eQ tool. These in turn refer to the UK Government Digital Standards, which most UK Government departments adhere to. The Blaise 5 layout designer made it possible to replicate the screen designs across different device screen sizes.

Figure 1. Mobile and laptop survey sign-in screens

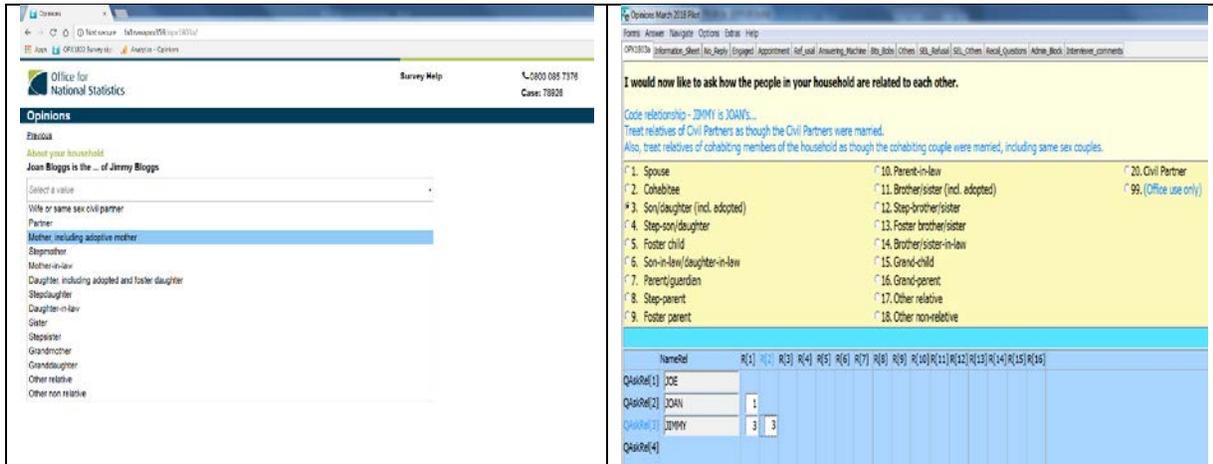


As this was the first time that layout templates designed in Blaise 5 would be used in an ONS web survey, considerable effort was made to ensure that they displayed as intended across different platforms. A group of volunteers from within ONS brought their devices to drop-in sessions to see how the templates rendered on a variety of different screen sizes and browsers. These sessions proved invaluable, and the findings influenced further changes to the templates.

The structure of the web questionnaire was often influenced by the display requirements to maximise the respondent experience when completing the survey. This was primarily driven by the research done as part of the development of another online survey using the eQ tool. This often conflicted with the structure of the CATI questionnaire, which had been designed over time to enable the interviewers to conduct the survey quickly.

The family relationship questions are a good example of the design having to reflect the needs of the questionnaire user. The web questionnaire was designed to collect relationships between household members; these relationships were collected one page per person. The CATI questionnaire has historically had a grid approach to collecting the household relationships, with interviewers being able to navigate quickly through many repeating questions for all the household members.

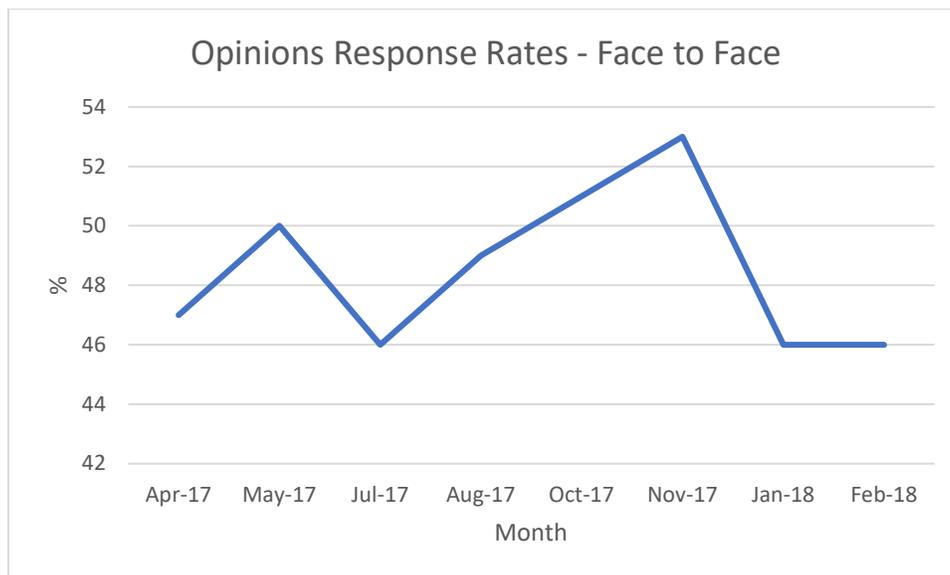
Figure 2. Web vs. Phone - Comparison of Relationship Questions



#### 4. The results

A key driver for conducting the mixed-mode survey pilot was to evaluate the effect on response rates, which have been showing a decline over the last few years. Prior to the mixed-mode pilot, response rates for the Opinions CAPI survey ranged from 46% to 53% over a 12-month period (see Table 1).

Table 1. Opinions Response Rates – Face to Face



The mixed-mode pilot saw a substantial increase in response rates when compared with the face-to-face interviews. The sample was drawn from the last wave of the Labour Force Survey (LFS), and consequently a slightly higher response due to more compliant respondents might be expected. However, it could also be

argued that respondents at this stage of survey cycle (LFS has five waves) may be suffering from survey fatigue. Nevertheless, the response rates for the mixed-mode pilot were encouraging, as Table 2 below shows.

Table 2. Response as percentage of issued sample

Completion	February	March	Total
Online	24.5	27.6	26.1
Partial	2.7	1.7	2.2
Telephone	34.1	31.4	32.8
Total	61.3	60.7	60.8

Of all those who responded to the survey, telephone was the most successful mode for achieving completed interviews. A small number of respondents only partially completed the web survey, but a healthy number of respondents completed the web form.

Table 3. Response as percentage of all respondents

Completion	February	March	Total
Online	40.0	45.8	42.9
Partial	4.4	2.2	3.3
Telephone	55.6	52.0	53.8

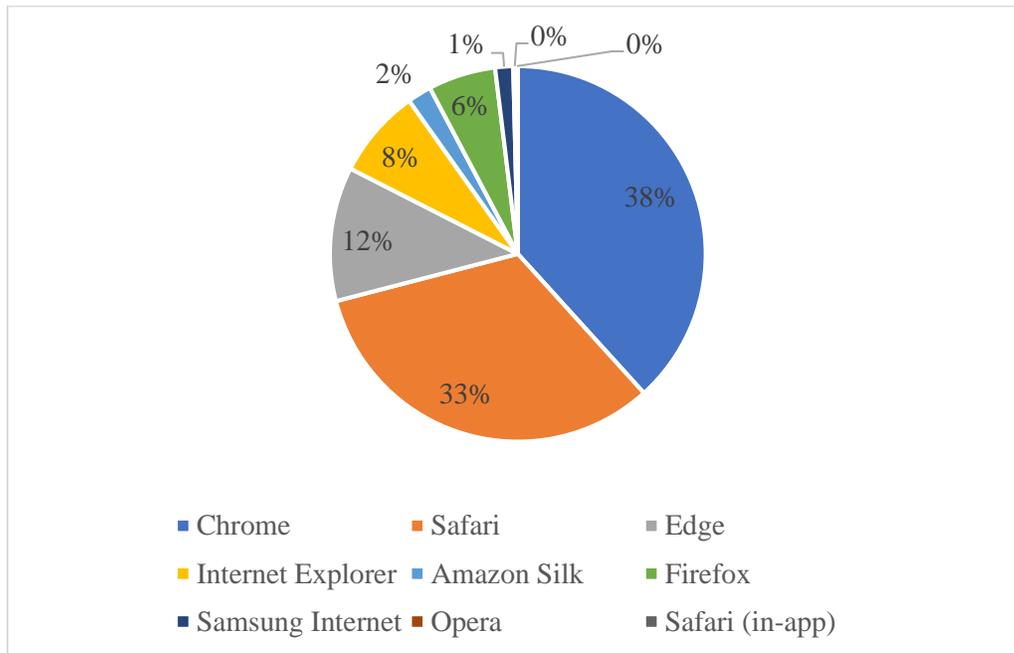
The Opinions questionnaire is comprised of several distinct sections. The first section, collecting demographic information about the household, saw the largest respondent drop-off in both collection months. The sections collecting responses to questions provided by sponsoring clients saw smaller drop-off rates as Table 4 demonstrates.

Table 4. Response as percentage of online respondents

Completion	February	March	Total
Full	90.1	94.2	92.2
Internet Access (section 3)	1.8	2.3	2.1
Smoking/e-cigarettes (section 2)	0.8	0.7	0.8
Demographics (section 1)	7.3	2.8	5.1

A variety of browsers were used to view the Opinions survey landing page on the ONS website. Table 5 shows that over the two survey periods, Chrome and Safari were clearly the most popular browsers, ahead of the Microsoft products. The unique pageview data was collected using Google Analytics.

Table 5. Opinions web page viewed by browser



## 5. Reflections on the mixed-mode project

Initial training and support from Statistics Netherlands was essential to using the Blaise Resource Database effectively. It is an extremely powerful tool, and for the beginner it can be daunting. Features such as automatically applying templates based on type names has led us to devise new questionnaire standards. One Resource Database and standard naming conventions will be used across all surveys. Users can then create questionnaires without the need to manually apply templates in the layout designer.

Statistics Netherlands were quick in resolving issues. Some were genuine bugs with the software, but the majority were due to programmer inexperience, particularly in creating the layout templates.

The advantages of a questionnaire that truly supports mixed-mode cannot be overstated. The Opinions questionnaire undergoes a lot of code changes during the testing phase. Duplication of effort replicating these changes across two questionnaires was time consuming, introducing a significant risk of inconsistency between the versions. A single database associated with the single questionnaire instrument, pre-loaded with all sampled cases available to any collection mode, is essential. This would avoid the timing issues that are inevitable when two modes are conducted with two separate questionnaires and databases.

The pilot web questionnaire demonstrated a lot of the layout functionality that is available in Blaise 5, but there were elements that could have been approached more effectively if time and resource had allowed. One such area was accessibility, and future work will include this essential work.

We did not fully realise the potential of paradata. We know that Blaise 5 offers a lot of functionality, but a restriction for us in NISRA hosting the survey was access to the audit data. The Opinions audit data shared the same audit database table as surveys conducted by NISRA. Unfortunately, there was not enough time to instigate a system to extract the relevant details for Opinions, and data security issues prevented the

whole database table from being copied. Obviously there was a lot of interest in the timings of the completion of individual questionnaire modules.

# Blaise 5 with RTI's Integrated Field Management System on Field Interviewer Laptops

*Lilia Filippenko, Preethi Jayaram, Joe Nofziger, Brandon Peele, R. Suresh - RTI International*

## 1. Abstract

For many years RTI's Integrated Field Management System (IFMS) is a standard system to use for CAPI projects on laptops for instruments developed in Blaise and other CAI software used by RTI. The laptop Case Management System (CMS) is configured to launch the specific CAI software as required by the project. Since Blaise 5 uses a different approach compared to Blaise 4 to install and launch Blaise instruments on laptops, we needed to adapt our CMS to support Blaise 5. We tweaked our CMS and upgraded the associated Manipula scripts, and now RTI's IFMS can support Blaise 4, Blaise 5 and other CAI software as needed. The paper will describe the process we followed and the challenges we encountered during this development.

## 2. Overview of Integrated Field Management System (IFMS)

IFMS is used for almost all field studies conducted by RTI and is a web-based application responsible for electronic assignment and transfer of cases to field staff, standard case status reports, data transmission, field monitoring, interviewer production, and laptop case management. IFMS is set up for every project regardless of the CAI software used on Field Interviewer (FI) laptops.

The general approach for field studies is as follows:

- A Master database for all cases of the study resides at RTI and is used to create a zip file for every case in the study with preloaded information
- Each case is assigned by a Field Supervisor to a Field Interviewer using the IFMS website, so cases are distributed among laptops
- When an FI laptop connects to RTI, cases assigned to him/her are transmitted from RTI and loaded in the laptop's database. It could be MySQL, Blaise, or SQLite database.
- If a case needs to be assigned to another interviewer, the Field Supervisor initiates an order on the IFMS website to transfer a case and its data files are transferred between laptops.
- When an interview for a case is completed, its data file is sent back to RTI for loading into the Master database.

Although many special functions are used by IFMS, just a few of them are specific to a software package and need adjustments, such as export/import case data from/to Master/Laptop databases; update status of the case on laptop set by the instrument or entered by FI; and launch the instrument.

## 3. Overview of Case Management System (CMS)

Before the start of data collection for a project, all necessary software and instrument files are loaded on laptops at RTI. At RTI International a .Net Case Management System (CMS) is used on laptops. The CMS application allows Field Interviewers to update case status, enter comments, launch instruments, and synchronize the status of cases with a centralized SQL server database. Various software packages or languages are used to develop instruments, including Blaise, Hatteras, and others. The type of software

package is stored in configuration files and is used by the CMS to create and invoke an appropriate CAI package object to manage work with the individual package. The CAI package is a software abstraction that hides the details of the language/software from the CMS, allowing it to perform essential operations on it without knowledge of the underlying implementation. Creating a new CAI Package class allows the CMS to work with new or custom survey administration tools with minimal change.

Methods implemented by the CAI package and applied for every case on the laptop include:

- **Import** (load a case into the laptop database)
- **Update event and status** codes for a case to the instrument database
- **Invoke** an instrument for a case
- **Export** (extract a case from the laptop database for transfer)

To work with Blaise 5, a new CAI package was added to the CMS. It was copied from the Blaise 4.8 CAI package and changes were made to pass modified or new parameters to accommodate requirements for Blaise 5. The goal for moving to Blaise 5 was, and still is, to maintain the same functionality as before with Blaise 4 while take advantage of many new Blaise 5 features. By maintaining the Blaise 4.8 CAI package alongside the new one for Blaise 5, the CMS continues to support both versions.

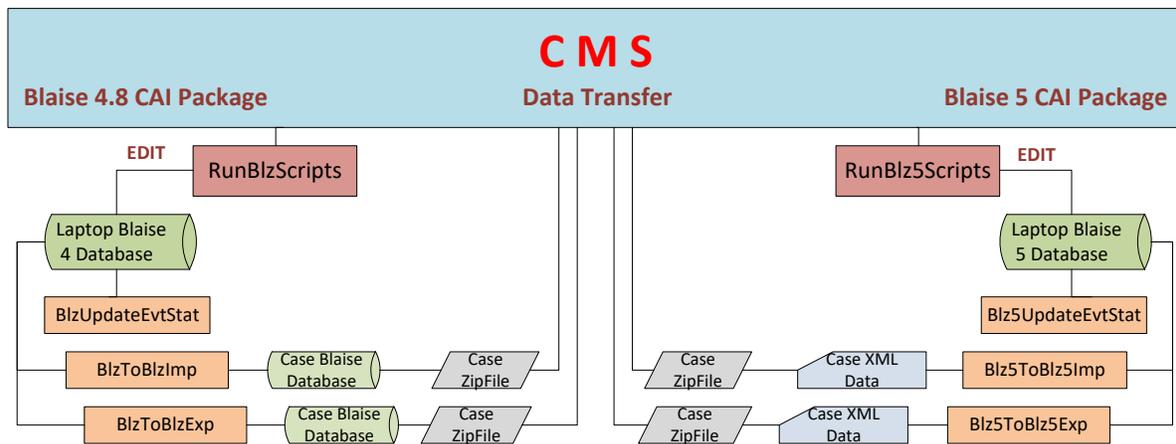
This paper will describe implementation of these methods with Blaise instruments and challenges with applying them to Blaise 5 instruments.

## 4. CMS with Blaise Instruments on Laptop

### 4.1 Manipula Process and Setup Scripts

For Blaise instruments, the CAI package invokes a single main Manipula process script, passing a parameter to execute the requested method by calling corresponding Manipula scripts. Our Manipula-driven architecture is illustrated in Figure 1. The CMS handles transfer of encrypted case zip files to and from FI laptops. The Blaise 4.8 CAI package is shown on the left side and the Blaise 5 CAI package on the right. Each executes the appropriate main Manipula process script, which in turn calls other Manipula setups. The same set of Manipula scripts is used by the CAI packages, with small differences in parameters and type of input/output files.

Figure 1. CMS with Manipula Scripts on Laptop



The main Manipula process script is developed in a general way so that it can be prepared independently of the Blaise instrument with which it is used, and without changes to the CMS. This approach gives us the flexibility to adjust requirements for different studies as needed. An example of this is the ability to call an external application before conducting the interview or to spawn new cases after completion of the interview without returning to the CMS.

An abbreviated example from the main Blaise 4 Manipula process script used for current projects in the field is provided in Appendix A. It shows the calls to Manipula setups to load a case, extract a case, update event and status code in the Blaise database, and a direct call of the Edit function to start Blaise instrument. In the CMS a command line is constructed to start Manipula.exe with parameters where the type of the method to use is defined as the first parameter. The name of the data model and database is also passed as a parameter. Other parameters are the case ID, a study identifier, and a folder name to use.

## 4.2 Challenges in converting Manipula Scripts from Blaise 4 to Blaise 5

We successfully converted all our Blaise 4 Manipula scripts and just modified “EXPORT”/“IMPORT” Manipula setups to use file type “XML” for output and input respectively. We also did small changes to the list of parameters and how they were passed to Manipula scripts. To use the EDIT function a name of the package file was used instead of an instrument name, and run mode was specified as “ThickClient”. But we discovered a few unexpected things and found that we needed to make more changes to Manipula scripts for Blaise 5. Some of challenges encountered are described below.

### 4.2.1 Additional Data Model Files

When we built a solution containing all our Manipula scripts, we noticed that for data models defined inside Manipula scripts in Blaise 5, prepared data models ending with “\$\$\$bmix” were created. For example, to create an ASCII file to log a result of different steps during the execution of RunBlz5Scripts, we use the data model “Info” defined in the RunBlz5Scripts.man as shown in Figure 2. After building RunBlz5Scripts.msu, we noticed that the file “runblz5scripts\_Info\$\$\$bmix” was created. If the same datamodel was used in a few scripts, a new “\$\$\$bmix” was created each time.

Figure 2. Example of Data Model to Create Log File

```
DATAMODEL Info
FIELDS
    date : string[8]
    sp1  : string[1]
    time : timetype
    sp2  : string[1]
    name : string[50]
    sp3  : string[1]
    success : string[7]
ENDMODEL
```

As it turns out these files are needed on the laptop to successfully run Manipula scripts. We decided to create data model files and use them directly in Manipula scripts as shown in Figure 3. They are also added to the solution with Manipula scripts for installation on laptops.

Figure 3. Example of Code to Output Log File

```
USES
  MoveCase 'MoveCase'
  Info 'Info'
  //---More code---
OUTPUTFILE file2:info('info.log',ascii)
  SETTINGS
    MAKENEWFILE=NO
  //---More code---
PROCEDURE InfoFile
  PARAMETERS
    Import intReslt:integer
    Import strText:string
  INSTRUCTIONS
    IF intReslt<>0 THEN file2.success:='Failed '
    ELSE file2.success:='Success'
    ENDIF
    file2.date:=DATETOSTR(SYSDATE, 'MMddyyyy')
    file2.time:=sysTime
    file2.name:=strText
    file2.write
  ENDPROCEDURE
  //---More code---
  InfoFile(Reslt,'Run Interview for ' + strCaseId)
```

Starting with Blaise version 5.4 an additional file, Manipula runtime meta “\_locals\$\$\$bmix”, is created for a Manipula script that should be installed along with the “.msux” file on laptops. The number of files to install on laptops is therefore doubled, since for Blaise 4 only the “.msu” files are distributed on laptops to call Manipula scripts.

#### 4.2.2 Blaise Data Interface File for ASCII and XML Files

During testing in the developer environment, we noticed that a Blaise Data Interface File (.bdix) was created in addition to the ASCII or XML file. It looked like it could be helpful to see the data especially for an XML case data file created by the “EXPORT” script and transferred to RTI. On the laptop all scripts were also working fine when we tested with version 5.3.0.1501.

When we started testing Manipula scripts upgraded to Blaise 5.4 on laptops, using version 5.4.2.1669, an error message was thrown that the “bdix” file cannot be created without a license. It was not clear why we needed a license with the new version and whether it was related only to this file. We contacted the Blaise support and received a suggestion to use special setting in our Manipula scripts to not create that file. As a result, all our scripts now have this setting: “CREATEBDIX=NO”. It is good, though, to have that file for in-house review of XML case data files using the Blaise Data Viewer.

#### 4.2.3 Production and Training Cases

With versions of Blaise before Blaise 5, RTI’s standard practice was to distribute instruments on laptops into two folders – a folder where the production interview data was collected and a folder where FIs could be trained to work with the instrument. Therefore, special cases were prepared and loaded on laptop in “Train” folder along with a Manipula script used to reload those training cases if necessary. With Blaise 5 only one instance of the instrument can be deployed on the laptop, so we needed to make a change to require a special field present in all of the Blaise 5 instruments – “modeID”. Manipula setups to load data

in the Master database at RTI have this field loaded as “4” (production mode), and Manipula scripts to load training cases use a value of “3”. With Blaise 4 when FIs need to have all training cases reinitialized, the training database is simply deleted by the CMS and Manipula script to reload training cases is called. To reinitialize training cases with Blaise 5, we needed to modify Manipula script to remove only the training cases from the Blaise database and then load them again.

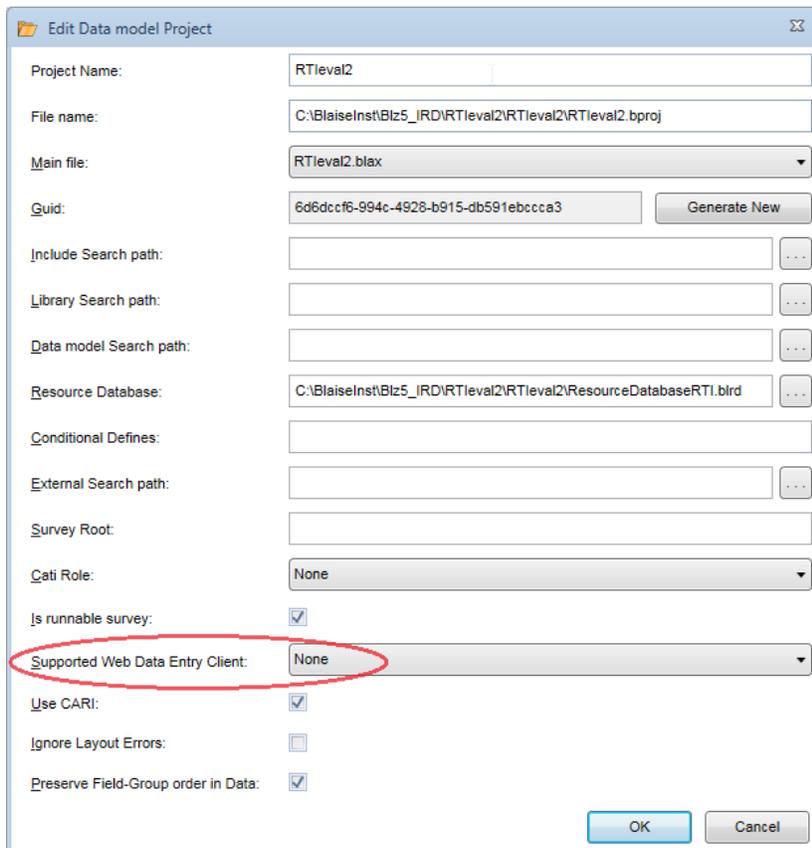
### 4.3 Blaise 5 Instrument Deployment as “Stand Alone”

When the “Edit” method to start a data entry session was added to Manipula in Blaise 5.4, we started using Manipula.exe to invoke Blaise instruments from a Manipula process script. Blaise is not installed on our laptops and only a few supporting files are needed. These additional files and settings to build a Blaise 5 instrument for running as “Stand Alone” on a laptop are described below.

#### 4.3.1 Building a Package to Deploy on Laptop

To distribute a Blaise 5 instrument on Field Interviewer laptops, a package (.bpkg) should be built in the Control Center and the project settings must be changed so only needed files are included in the package. This is done by selecting “Edit Project” from the dropdown menu in the Solution Explorer and the “Edit Data Model Project” dialog will open. The value of the “Supported Web Data Entry Client” should be set to “None” as shown in Figure 4 to have only basic instrument files included in the package.

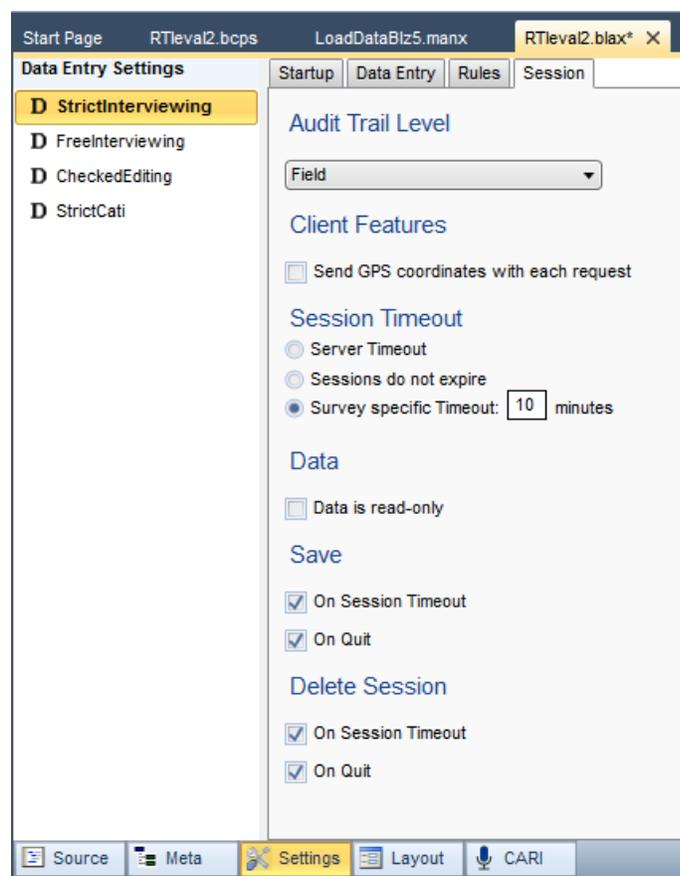
Figure 4. “Supported Web Data Entry Client” Value in Data Model Project



Although it was recommended to set the Session Timeout in the session tab of the applied Data Entry Settings to “Sessions do not expire”, we decided to use “Survey specific Timeout” as shown in Figure 5.

With these settings, collected interview data is saved in the Blaise database and deleted from the Session database regardless of the interview status - completed or partial. This is critical if partial case need to be transferred to a different FI and our Manipula scripts use the Blaise database to export a partial case.

Figure 5. Setting for Session Timeout



### 4.3.2 Programs needed on Laptop

During the CMS installation on laptop, folders are created as specified in configuration settings for the project. For all projects a folder is created for programs that are needed for the project, and it is used later by the CMS to run any specific project's program from that folder. The following programs and components were copied into this folder on the laptop:

- Manipula.exe
- Manipula.exe.config – Configuration file with deploy folder value as "C:\Blaise5\StandAlone"
- System.Data.SQLite.dll – SQLite library
- Msvcr100.dll – Microsoft Visual C++ Runtime library
- Survey package file (.bpkg) build as described above in 4.3.1
- Batch file to start a Manipula script to install the package
- Manipula process script to select the package and install it

The first four files were copied from the Blaise installation folder '..\StatNeth\Blaise5\Bin' on the developer's computer. The .NET Framework (4.5.1 or higher) is also required for running Blaise 5 instruments and is already installed on RTI laptops, as it is needed by the CMS.

The Manipula process script (Figure 6) and a batch file to run it can be used to install any survey used by the study.

Figure 6. Manipula Process Script to Install Survey

```
PROCESS InstallSurvey "Install Survey on Laptop"  
AUXFIELDS  
Survey: STRING[100]  
MANIPULATE  
Survey:= SELECTFILE('Select survey',", 'Blaise Package (*.bpkg)| *.bpkg')  
INSTALLPACKAGE(Survey)  
END
```

The batch file has the following code:

“Manipula.exe InstallSurvey.msux”.

After a successful run, instrument files are installed in “C:\Blaise5\StandAlone\Surveys\<Survey name>”.

#### 4.4 Installation of Manipula Scripts

Our Manipula scripts to run with the CMS are standard; even the name of the instrument in them is the same. With Blaise 4 we use a special program to prepare an installation package with instrument files and Manipula scripts to install on the laptop by running single batch file. As a first step that program uses “b4cpars.exe” to prepare Manipula scripts for a data model passed as parameter. Unfortunately, there is no way right now to use the same approach with Blaise 5. We need to use the GET method in some Manipula setups, so we cannot use the reserved word VAR to pass name of the data model at run time. Although Instrument Builder can be used to prepare all Manipula scripts in the solution at once, we still need to manually change the name of the data model in such Manipula setups.

After Manipula scripts and the data models used in them are prepared, they are copied to the folder where the instrument files were installed by the “InstallSurvey” Manipula script. We hope that in the future we will have something like with Blaise 4 when scripts could be prepared without any changes.

#### 4.5 Audit data to extract

With Blaise 4 we have a configuration file on the laptop to create an audit file for every case and it is transmitted to RTI along with the interview data for a case. In Blaise 5, Audit data is saved in the SQLite database for all cases. Our goal now is to have a Manipula process script that will extract audit data for a case when the case is exported after it is completed or is partial and needs transfer to a different FI.

## 5. Conclusion

Most of our work is done and we are ready to use Blaise 5 for CAPI studies. Still ahead is to modify some ancillary tools we have developed for working with Blaise 4. These tools include defining procedures during production to do (1) changes to a data model that are harmless to data and therefore suitable for automatic adjustment and (2) changes which cause data file incompatibility. We plan to modify an existing program to apply upgrades to production Blaise 5 databases. It will use the same approach as for Blaise 4 databases, creating backup folders and Manipula scripts to upgrade the Master database and then

the database on the laptop. We will also evaluate tools in Blaise 5 to prepare SAS datasets for delivery, or upgrade our “BlaiseToSAS” application to use with SQLite and SQL Server databases.

With some reading and experimentation, we have learned several differences in laptop installation, script invocation, and storage between Blaise 4 and Blaise 5. Our existing infrastructure and componentization lessen the impact on us. We hope the implementation changes we have documented here will help others to adjust their processes.

## 6. References

Blaise 5 Help Reference Manual, Statistics Netherlands

<http://help.blaise.com/>

Blaise 5 Tutorials, Statistics Netherlands, “Blaise 5 Deploying DEP Stand Alone”, 2016

“Libraries\Documents\MyDocuments\Blaise5\Tutorials\Data Entry\Deploying\_DEP\_Stand\_Alone.pdf”

## Appendix A. Example of Blaise 4 Manipula Script “RunBlzScripts.man”

---

```
strScriptName := uppercase(PARAMETER(1))

CASE strScriptName OF

  'IMPORT':
    strWorkF := PARAMETER(2)
    strCaseId := PARAMETER(3)
    strOutFile := PARAMETER(4)
    strInstName := PARAMETER(4)
    strProjectID := PARAMETER(5)

    LogFile('BlzToBlzImp BEFORE ' + PARAMETER(3))
    strRun:= 'BlzToBlzImp /W' + strWorkF + ' /I' + strCaseID + ' /O' + strOutFile + ' /P' + PARAMETER(3) + ' /Q'
    Result:= CALL(strRun)
    LogFile('BlzToBlzImp AFTER ' + PARAMETER(3))

    IF Result<>0 THEN
      file1.success:='Failed '
    else
      file1.success:='Success'
    ENDIF
    file1.date:=sysDate
    file1.time:=sysTime
    file1.zrid:=parameter(3)
    file1.imp:='I'
    file1.write

  'EXPORT':
    strWorkF := PARAMETER(2)
    strInstName := PARAMETER(3)
    strOutFile := PARAMETER(4)
    strCaseId := PARAMETER(4)
    strXfer := PARAMETER(5)
    strProjectID := PARAMETER(6)

    strParam :='/P' + strCaseID + ';' + strXfer
    strRun:= 'BlzToBlzExp /W' + strWorkF + ' /I' + strInstName + ' /O' + strOutFile + ' ' + strParam + ' /Q'
    LogFile('BlzToBlzExp BEFORE ' + PARAMETER(4))
    Result:= CALL(strRun)
    LogFile('BlzToBlzExp AFTER ' + PARAMETER(4))

    IF Result<>0 THEN
      file1.success:='Failed '
    ELSE
      file1.success:='Success'
    ENDIF
ENDIF
```

```

file1.date:=sysDate
file1.time:=sysTime
file1.zrid:=parameter(4)
file1.imp:='E'
file1.write

'UPDATE':
strWorkF := PARAMETER(2)
strInstName := PARAMETER(3)
strCaseId := PARAMETER(4)
strProjectID := PARAMETER(9)
strParam := ''' + '/P' + strCaseId + ';' + PARAMETER(5) + ';' + PARAMETER(6) + ';' + PARAMETER(7) + ';' + PARAMETER(8) + '''
strRun:= 'BlzUpdateEvtStat /W' + strWorkF + ' /I' + strInstName + ' ' + strParam + ' /Q'
Result:= CALL(strRun)
InfoFile(Result,'BlzUpdateEvtStat for ' + strCaseId)

'INVOKE': (do not need special script)
strWorkF := PARAMETER(2)
strInstName := PARAMETER(3)
strProjectID := PARAMETER(5)
IF PARAMETER(4)<>' THEN
    strCaseId := ' /K' + PARAMETER(4)
ELSE
    strCaseID :=''
ENDIF
{Custom code to create lookup table with CW names who completed OC section}
IF (Uppercase(strInstName)='CSWRKR') AND (strCaseId<>'') AND (strProjectID='14780') THEN
    strRunTemp :='C:\NSCAW\PGMS\CreateCWnamesDB.exe'
    IF FILEEXISTS(strRunTemp) THEN
        {get value of PSUID for the case}
        strTemp := PARAMETER(4)
        strPSUID := SUBSTRING(strTemp,2,3)
        strRunTemp := strRunTemp + ' /PSU=' + strPSUID + ' /' + strTrainProd
        Result:= run(strRunTemp, WAIT)
        InfoFile(Result,'CreateCWnamesDB')
    ENDIF
ENDIF
strRun:= strWorkF + '\ ' + strInstName + ' /G /X /C' + strInstName + '.diw @' + strInstName + '.bcf ' + strCaseId
IF strInstName<>' THEN

    Result:= EDIT(strRun) (Start Interview)

    InfoFile(Result,'Run Interview for ' + PARAMETER(4))
    {Custom program to create cases for RECS - 14615 }
    IF ((Uppercase(strInstName)='RX2015') AND (strProjectID='14615')) THEN
        strTemp:= GETVALUE(strInstName,strInstName,PARAMETER(4), 'main_case.sum_stat')
        IF (strTemp = '2690') THEN
            strRun:= 'C:\RECS\pgms\CreateCasesNet.exe'
            IF FILEEXISTS(strRun) THEN
                DISPLAY('Wait one moment please...' )
                {Create RECS cases}
                strRun:= 'C:\RECS\pgms\CreateCasesNet.exe /ID=' + PARAMETER(4) + ' /' + strTrainProd
                Result:= run(strRun, WAIT)
                InfoFile(Result,'Run Create Cases for ' + PARAMETER(4))
            ENDIF
        ENDIF
    ENDIF
ENDIF
ELSE
    InfoFile(1, 'Run Interview for empty InstName')
ENDIF

'RESET':
strWorkF := PARAMETER(2)
strInstName := PARAMETER(3)
strOutputF := PARAMETER(2)
strProjectID := PARAMETER(5)
{PARAMETER(4) is a script name do load data in training DB}
IF FILEEXISTS(PARAMETER(4)) THEN
    strRun:= PARAMETER(4) + ' /W' + strWorkF + ' /O' + strInstName + ' /T' + strOutputF + ' /Q'
    Result:= CALL(strRun)
    InfoFile(Result,'Reset Train Cases')
ENDIF

ENDCASE

```



# Conducting Offline Blaise 5 Surveys in a Distributed Environment

*Marsha Skoman, University of Michigan Survey Research Center*

## 1. Introduction to SurveyTrak

The University of Michigan Survey Research Center (SRC) uses *SurveyTrak*, one of its sample management systems, to create cases on the fly in the field and to conduct Blaise 5 surveys for those cases. SurveyTrak runs on Windows laptops and the surveys run as thick client/standalone. The Blaise Control Centre is not installed on the laptops; only Dep.exe, Manipula.exe, and a few other files are on the laptops.

SurveyTrak uses SAP's SQL Anywhere database replication system. Each laptop has a remote SQL Anywhere database which replicates to a central, consolidated SQL Anywhere database. Interviewers upload and download survey and sample management data via the replication system. Data models are downloaded via an FTP-based file send utility. This upload/download process is called "send/receive" in SurveyTrak.

SurveyTrak is used for multiple studies and each study can have its own survey or set of surveys. The surveys can be compiled in any version of Blaise 4.8+ or Blaise 5.3+. SurveyTrak can also accommodate suspending and later resuming surveys as well as updating data models during data collection.

Each case is treated individually – there is one Blaise database (BDBX) and one AuditTrailData database per case. Merging of the audit data and of the survey data into one master Blaise database happens at another point in the process.

## 2. SurveyTrak Laptop Configuration

### 2.1 DEP and Manipula Files

The DEP and Manipula files are stored by Blaise version in the folder **C:\Blaise**:

- C:\Blaise\Blaise4841904
- C:\Blaise\Blaise5301501
- C:\Blaise\Blaise5431675

Each version folder (e.g., **C:\Blaise\Blaise5431675**) contains the files required to run the surveys as thick client/standalone:

- C:\Blaise\Blaise5431675\Dep.exe
- C:\Blaise\Blaise5431675\Manipula.exe
- C:\Blaise\Blaise5431675\msvcr100.dll
- C:\Blaise\Blaise5431675\NAudio.dll *This file is required even if the survey isn't using CARI*
- C:\Blaise\Blaise5431675\System.Data.SQLite.dll

### 2.2 Study-Specific Survey Files

The folder **C:\BLProj** contains subfolders for each study:

- C:\BLProj\ECHO
- C:\BLProj\GIT18
- C:\BLProj\HRS2018

C:\BLProj\MEM\MEMBaseline  
C:\BLProj\MEM\MEMScreenener  
C:\BLProj\YWC

Data models for a particular study are stored by version date/time in the folder **C:\BLProj\HRS2018\Storage**. This folder naming convention allows SurveyTrak to simply use the most recent data model without having to specify which version should be used.

C:\BLProj\HRS2018\Storage\2018-03-15,22,20,00  
C:\BLProj\HRS2018\Storage\2018-04-25,16,10,00  
C:\BLProj\HRS2018\Storage\2018-05-10,14,23,00  
C:\BLProj\HRS2018\Storage\2018-06-21,16,23,00  
C:\BLProj\HRS2018\Storage\2018-08-03,09,35,00

The data model version date/time folder initially contains only the following two files:

C:\BLProj\HRS2018\Storage\2018-08-03,09,35,00\Dep.exe.config  
C:\BLProj\HRS2018\Storage\2018-08-03,09,35,00\HRS18.bpkg

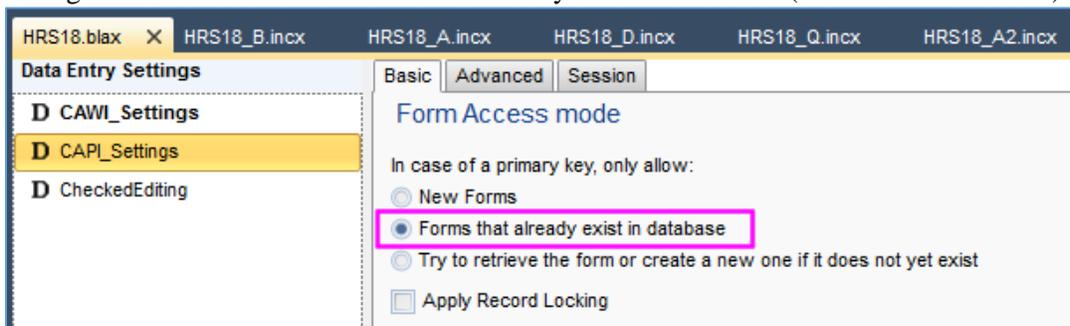
DEP requires the *Dep.exe.config* file. The Blaise programmer creates the file for each version of the data model (see step 3.5). SurveyTrak copies it to the proper Blaise folder (e.g., C:\Blaise\Blaise5431675) before DEP is called. Regarding the BPKG file: SurveyTrak unzips it the first time a particular data model is used. If an updated version of the data model is released during data collection, the new folder and these two files are sent to the SurveyTrak laptops via send/receive.

### 3. Preparing the Blaise Data Model

The Blaise programmer prepares the data model BPKG and Dep.exe.config files as outlined below and delivers them to the Interviewer Help Desk staff. The Interviewer Help Desk staff send the files out to the field and the interviewers do a SurveyTrak send/receive to download the files.

#### 3.1 Form Access Mode Setting

A critical Blaise setting for the SurveyTrak thick client/standalone environment is the Form Access Mode setting. It needs to be set to “Forms that already exist in database” (see screen shot below).



#### 3.2 Modify the BPKG File

Once the data model BPKG file is created, the Blaise programmer uses 7-Zip (<http://www.7-zip.org/download.html>) or a similar compression utility to delete the BDBX file. The BDBX file is created later on the SurveyTrak laptop by a preload Manipula script (see step 3.3).

#### 3.3 Preload Manipula Script

The Blaise programmer creates an ASCII to Blaise Manipula script which is used by SurveyTrak to create the case's BDBX file. The compiled Manipula script is added to the BPKG file using 7-Zip. When compiling the Manipula script, one or more \$\$\$B.MIX files may be created. These files must also be added to the BPKG file.

Sample Manipula script for BDBX creation (ASCII to Blaise):

```

SETTINGS
  DATEFORMAT = MMDDYY
  DATESEPARATOR = '/'
  DESCRIPTION = 'ASCII to BLAISE'

USES
  OutputMeta 'HRS18'
  DATAMODEL InputMeta
    ATTRIBUTES = DK, RF
    FIELDS
      SampleID      : STRING[15]
      BLOCK BPreload
        FIELDS
          FName      : STRING[25]
          LName      : STRING[25]
          ScriptControlID : STRING[3]
        ENDBLOCK {BPreload}
      FIELDS
        Preload      : BPreload
    ENDMODEL

INPUTFILE InputFile1: InputMeta ('preload.asc', ASCII)
SETTINGS
  SEPARATOR = '^'

OUTPUTFILE OutputFile1: OutputMeta ('HRS18', BLAISE)

SETTINGS
  MAKENEWFILE = NO

MANIPULATE
  OutputFile1.WRITE

```

### 3.4 Migration Manipula Script

If there is more than one version of the data model, the Blaise programmer also creates a Blaise to Blaise Manipula script which is used by SurveyTrak to migrate suspended cases from one data model to another. The compiled Manipula script is also added to the BPKG file. When compiling the Manipula script, one or more \$\$\$B.MIX files may be created. These must also be added to the BPKG file.

Sample Manipula script for laptop migration (Blaise to Blaise):

```

SETUP MigrationScript

SETTINGS
  DESCRIPTION = 'HRS 2018 Test Blaise to Blaise Migration Script'
  METASEARCHPATH = 'C:\blproj\HRS2018\storage'

USES
  InputMeta '2018-06-21,16,23,00\Surveys\HRS18\HRS18'
  OutputMeta '2018-08-03,09,35,00\Surveys\HRS18\HRS18'

INPUTFILE InputFile1: InputMeta ('C:\blproj\HRS2018\storage\2018-06-21,16,23,00\Surveys\HRS18\HRS18', BLAISE)
OUTPUTFILE OutputFile1: OutputMeta ('C:\blproj\HRS2018\storage\2018-08-03,09,35,00\Surveys\HRS18\HRS18', BLAISE)

MANIPULATE
  OutputFile1.WRITE
END

```

### 3.5 Dep.exe.config File

The Blaise programmer also creates a *Dep.exe.config* file for each version of the data model. This file is *not* inserted into the BPKG file; it is delivered separately.

Sample Dep.exe.config file:

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="ExternalCommunicationAddress" value="http://localhost:8033"/>
    <!-- The DeployFolder needs to be updated with every Release. -->
    <add key="DeployFolder" value="C:\blproj\HRS2018\storage\2018-08-03,09,35,00"/>
    <add key="ConfigurationDatabase" value="RuntimeConfiguration.db"/>
    <add key="ManagementCommunicationPort" value="8031"/>
    <add key="ExternalCommunicationPort" value="8033"/>
    <add key="ServerManagerDatabase" value="ServerManagerDatabase.db"/>
  </appSettings>
</configuration>
```

## 4. SuveyTrak Interview Process

### 4.1 Conduct Interview

In SurveyTrak the interviewer selects a study, then selects a case to interview. When the Conduct Interview button is clicked, SurveyTrak performs the following actions:

1. Retrieves the Blaise preload (a caret-delimited string) from the laptop database and writes it to an ASCII text file called *preload.asc*.
2. Temporarily moves the BDIX file to another directory.
3. Unzips the BPKG file if it is the first time an interview is conducted with this version of the data model. (The Survey and Settings folders are created when DEP runs.)
4. Constructs a Manipula command and runs Manipula. The Manipula script used is the Preload script from step #3.3.
5. Overwrites the BDIX file created by Manipula with the BDIX file moved in step #2. This prevents a Blaise “incompatible data” error.
6. Copies the *Dep.exe.config* file to the proper location.
7. Constructs a DEP command and runs DEP to launch the survey.

**Sample Manipula command:** C:\Blaise\Blaise5301501\Manipula.exe C:\blproj\HRS2018\storage\2018-08-03,09,35,00\surveys\hrs18\AscToBla.msux -Q:true

**Sample DEP command:** C:\Blaise\Blaise5301501\Dep.exe C:\blproj\HRS2018\storage\2018-08-03,09,35,00\hrs18.bpkg -KeyValue:<case ID> -RunMode:ThickClient -Fields:"xTimegate=1" -AssignMode:Always -DataEntrySettings:CAPI\_Settings -LayoutSet:HRS\_Iwer -LayoutSetGroup:IWERADMIN -EnableResize:true -InitialWindowState:Maximize -EnableClose:false

### 4.2 Post-Interview Processing

When the interviewer exits the survey, SurveyTrak performs the following actions:

1. Zips up the case BDBX and the *AuditTrailData.db* file and inserts the zip file into a BLOB field in the laptop database.
2. Queries the Blaise database for survey data that the study wants copied to the laptop database.
3. Updates the laptop database with the following sample management data:

- a. Interview status (suspended or completed).
  - b. Version of the data model used for the interview.
4. Deletes the file *Dep.exe.config* from the Blaise folder (e.g., C:\Blaise\Blaise5301501).
5. Deletes the following files from the Surveys folder so that the next case is treated as a fresh case:
  - a. HRS18.bdbx
  - b. HRS18.manifest
  - c. RuntimeSessionData2.db
  - d. Preload.asc
6. Deletes the following files from the Settings folder so that the next case is treated as a fresh case:
  - a. AuditTrailData.db
  - b. RuntimeSessionData2.db

### 4.3 Resuming Suspended Interviews

If the interview being conducted was previously suspended, SurveyTrak performs the following actions in addition to the actions in step #4.1:

1. Unzips the BPKG file again if necessary.
2. Retrieves the case BDBX and *AuditTrailData.db* file from the zip file stored in the laptop database and restores them to the Surveys and Settings folders respectively.
3. Determines whether the survey data need to be migrated to the latest version of the data model.
  - a. Queries the laptop database for the data model date/time of the suspended interview.
  - b. Searches the folder C:\BLProj\HRS2018\Storage to determine whether there are more recent data models.
4. If applicable for the study, writes sample management data to the Blaise BDBX.

If migration is needed, SurveyTrak performs the following actions:

1. Creates a Hold folder: C:\blproj\HRS2018\Hold
2. Copies the *AuditTrailData.db* file of the suspended interview to the Hold directory.
3. For each new data model:
  - a. Unzips the BPKG file of the new data model if necessary.
  - b. Temporarily moves the BDIX file to the Hold directory.
  - c. Constructs a Manipula command and runs Manipula. The Manipula script used is the Migration script from step #3.4.
  - d. Overwrites the BDIX file created by Manipula with the BDIX file moved in step #4.c.ii. This prevents a Blaise “incompatible data” error.
4. Moves the *AuditTrailData.db* file from the Hold folder to the new Settings folder.
5. Copies the *Dep.exe.config* file of the new data model to the proper location (e.g., C:\Blaise\Blaise5301501).

**Sample Manipula command:** C:\Blaise\Blaise5301501\manipula.exe C:\blproj\HRS2018\storage\2018-08-03,09,35,00\surveys\hrs18\BlatoBla.msux -B:False -A:True -Q:True -S:True

### 4.4 Processing of Survey Data

Interviewers are encouraged to do a SurveyTrak send/receive on a daily basis to upload survey data to the consolidated database. An automated merge process runs nightly to process the completed cases that have been uploaded that day. The merge process runs against the consolidated database. The merge also writes to a Microsoft SQL Server database which contains audit data tables for each study. The audit data tables have the same structure as the *AuditTrailData.db* database and contain all the merged audit data for the study.

The merge process performs the following actions:

1. Extracts from the consolidated database the case BDBX and *AuditTrailData.db* files.
2. Uses Manipula to merge the single case BDBX into a master survey data BDBX.
3. Uses ODBC to connect to the SQLite database *AuditTrailData.db* and copies the audit data to the study's audit data schema in the SQL Server database.
4. If migration of the master survey data BDBX is required, migrates the master survey data.
5. Migrates the single case BDBX if required before merging it into the master survey data BDBX.

Sample Manipula script for master survey data merge (Blaise to Blaise):

```
SETUP HRS2018_Merge

SETTINGS
  DESCRIPTION = 'BLAISE to BLAISE'

USES
  InputMeta 'HRS18'
  OutputMeta 'HRS18'

INPUTFILE InputFile1: InputMeta ('\\st-rpt11\Merged_Data\Blaise\HRS2018\Production\Storage\2018-08-03,09,35,00\HRS18', BLAISE)
OUTPUTFILE OutputFile1: OutputMeta ('\\st-rpt11\Merged_Data\Blaise\HRS2018\Production\MasterSurveyData\HRS18', BLAISE)

SETTINGS
  MAKENEWFILE = NO

MANIPULATE
  OutputFile1.WRITE

ENDSETUP//HRS2018_Merge
```

Sample Manipula script for master survey data migration (Blaise to Blaise):

```
SETUP MigrateMergeScript

SETTINGS
  DESCRIPTION = 'HRS 2018 Production Blaise to Blaise Merge Migration Script'
  METASEARCHPATH = '\\st-rpt11\Merged_Data\Blaise\HRS2018\Production\Storage'

USES
  InputMeta '2018-06-21,16,23,00\HRS18'
  OutputMeta '2018-08-03,09,35,00\HRS18'

INPUTFILE InputFile1: InputMeta ('\\st-rpt11\Merged_Data\Blaise\HRS2018\Production\Storage\2018-06-21,16,23,00\HRS18', BLAISE)
OUTPUTFILE OutputFile1: OutputMeta ('\\st-rpt11\Merged_Data\Blaise\HRS2018\Production\Storage\2018-08-03,09,35,00\HRS18', BLAISE)

MANIPULATE
  OutputFile1.WRITE

END
```

## 5. Sample Management System Configuration

SurveyTrak is a data-driven application. All the information required for the system to function is in the consolidated SurveyTrak database. A subset of the consolidated database replicates to each laptop's database. Preloaded data used to create cases on the fly come from the database, as do the parameters for launching a survey. The survey data are also stored in the database as a BLOB data type. The primary key for most of the tables is the study ID and the case ID.

Some of the key tables are as follows:

- *tSample\_Line*: Preloaded with sample management information at the case level. Updated with survey status and survey data as needed when the survey is exited.

- *tCapi*: Preloaded with the survey preload at the case level. Updated with the data model date/time and the survey data BLOB when the survey is exited.
- *tBlaise*: Contains one row per study and per instrument. Contains information about how DEP and Manipula commands are constructed and other instrument parameters.
- The nightly merge process has its own set of tables for merge criteria and merge tracking.

Table 1. Sample Data from the tBlaise Table

Column Value	Description
SRC.SRO.HRS2018.PROD	Study ID
SELF	SurveyTrak instrument/survey ID
C:\Blaise\Blaise5301501	The version of Blaise to use and the location on the laptop of the DEP and Manipula EXEs. The <i>Dep.exe.config</i> file is copied to this folder by SurveyTrak before DEP is called.
Dep.exe.config	Name of the config file needed by DEP
C:\blproj\HRS2018	Study's root folder
C:\blproj\HRS2018\storage	Folder that contains the date/time data models subfolders
hrs18.bpkg	Name of the BPKG file to use when calling DEP
surveys\hrs18	Used to find the files inside of the date/time data model subfolder
HRS18	Root of the names of the data model files (e.g., hrs18.bdix, hrs18.bmix, etc.)
SamplID	Name of primary key field in Blaise. Used for pulling data from the instrument.
-EnableResize:true -InitialWindowState:Maximize -EnableClose:false	The command line parameters that control the behavior of the DEP window
Preload.asc	Name of the file that contains the preload string. Written from tCapi and used by the Preload Manipula script.
AscToBla.msux -Q:true	Name of the Preload Manipula script that puts the contents of the preload.asc file into the BDBX
HRS18.bdbx*..\settings\AuditTrailData.db* ..\settings\Cari.db*	Asterisk-delimited list of files to zip up into the BLOB
BlatoBla.msux	Name of Manipula script that does data model migration on the laptop

## 6. Using the Blaise API

SRC has written a C#.NET DLL that references functions from the Blaise API in order to bring survey data into SurveyTrak. This DLL is also used to write data from SurveyTrak to Blaise if the study requires it.

Passed to the DLL from SurveyTrak are the following:

1. Path to the BDIX. For example:  
C:\blproj\HRS2018\storage\2018-08-03,09,35,00\surveys\hrs18\HRS18.bdix
2. Path to the BMIX. For example:  
C:\blproj\HRS2018\storage\2018-08-03,09,35,00\surveys\hrs18\HRS18.bmix
3. Name of the primary key variable
4. Value of the primary key (the case ID)

The API functions used to retrieve the Blaise record are as follows:

```
IDataLink dl = DataLinkManager.GetDataLink(dataBasePath);
IDataRecord dr = null;
IDatamodel dm = MetaManager.GetDatamodel(dataModelPath);
```

```
IKey key = DataRecordManager.GetKey(dm,
StatNeth.Blaise.API.Meta.Constants.KeyNames.Primary);
IField field = key.Fields.GetItem(primaryKeyName);
field.DataValue.Assign(primaryKeyValue);
dr = dl.ReadRecord(key);
```

To read the value of a specified variable:

```
string toReturn = dr.GetField(nameOfVariableToPull).DataValue.ValueAsText;
```

To write a value to a specified variable:

```
dataRecord.GetField(fieldName).DataValue.Assign(pushValue);
dataLink.Write(dataRecord);
```

**Important note:** In order for writing to Blaise to work, the Blaise license file *serialnumber.txt* must reside in the same location as the C#.NET DLL. Before the license expires a new *serialnumber.txt* must be sent to the laptops.

## 7. CARI

An upcoming SRC field study will use Blaise 5.4 and CARI. The current plan is for SurveyTrak to treat the *Cari.db* file like the *AuditTrailData.db* file. The nightly merge process will extract the blobs from the *Cari.db* and copy the recording files to a file server.

# Blaise 5 Multimode Management – A Report by the BCLUB Multimode Management Group

*Coordinator: Mark M Pierzchala / Mark M Pierzchala, MMP Survey Services, LLC*

The BCLUB Multimode Management Group formed in October 2017. Its task was to define functionality for multimode survey management. The group members represent a wide variety of experiences and practices. The eight participating institutes and 16 individuals are:

- Office for National Statistics (ONS): Mike Hart
- RTI: R. Suresh, Gilbert Rodriguez, and Lilia Filippenko
- UK National Centre for Social Research: Sven Sjodin, Nafiis Boodhumeah, and Colin Miceli
- Statistics Norway: Hilde Degerdal, Trond Båshus, and Jan Haslund
- University of Michigan, Survey Research Center: Patty Maher – lead.
- Westat: Richard Frey
- Social & Scientific Systems, Inc.: Ans Bilhorn-Janssens, and Taylor Abernathy
- Statistics Netherlands: Lon Hofman and Tim Carati

Two institutes provided answers about their use of Blaise 4 CATI. They are

- Statistics Denmark, Leif Bochis
- National Agricultural Statistics Service, Emily Caron

## 1. Executive Summary and Some Founding Principles

Blaise 5 should provide a Multimode Management System that is flexible and adaptable by the institutes that use it. It should not duplicate capabilities that statistical institutes normally provide for themselves. The system should recognize survey management methods such as responsive and adaptive design. The following summarizes the principles that came from the Multimode Management Group.

**Multimode Survey Management** is all about contacting respondents and securing their cooperation. This is done through **channels of communication** (de Leeuw 2005). A few examples are mail, email, and phone. It is also necessary to recognize inbound communications such as mail in, log in, or call in.

**Modes** concern the technology used to conduct the survey. A few examples of modes are paper, web, or CATI. **Multiple modes** mean using a **multimode instrument** or **related mode-specific instruments** to conduct the survey. Sometimes the channel is tightly connected to the mode. This is true for CATI where the call attempt and interviewing are conducted with the same system. On the other hand, a web complete may be prompted by mail, email, a phone call, or a text message.

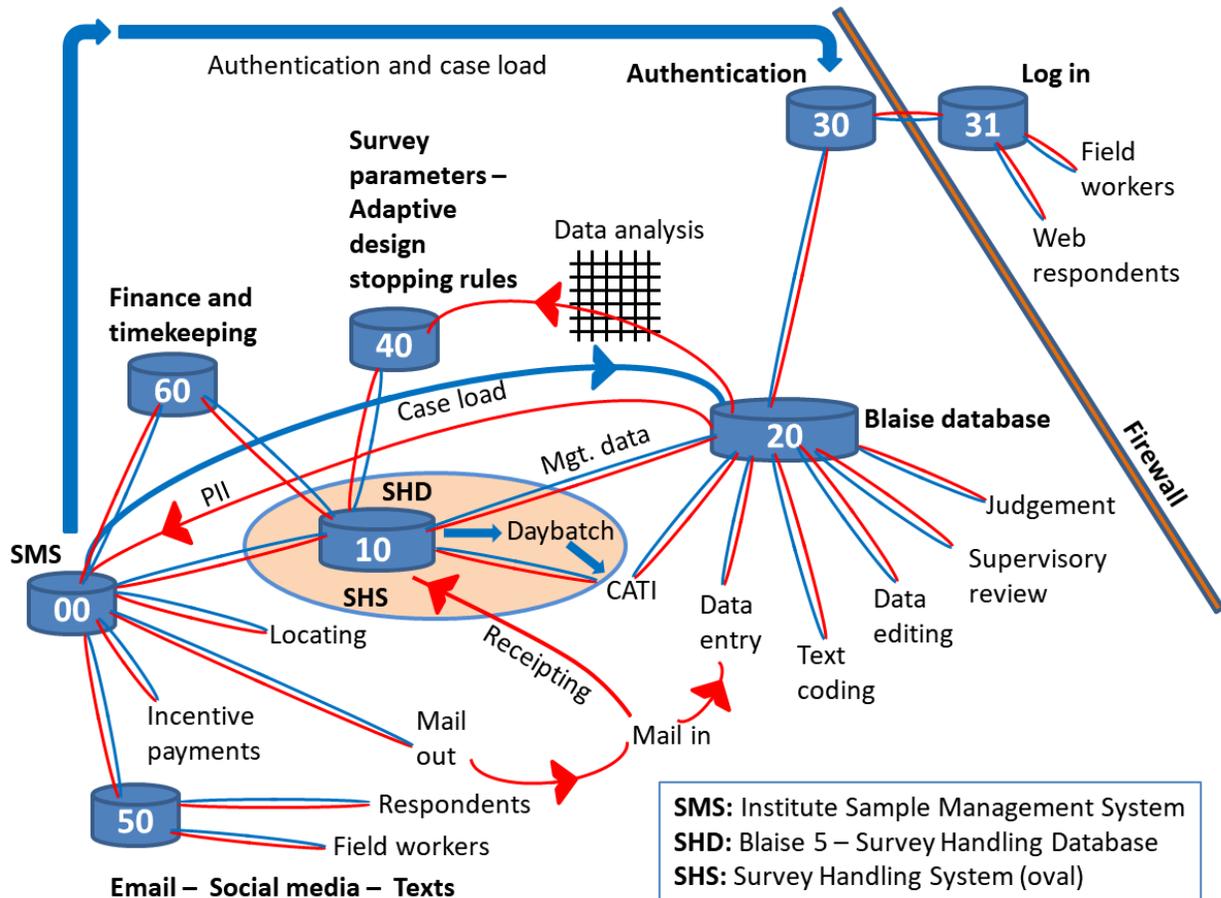
This report assumes the institute already has a **Sample Management System (SMS)**. It contains Personally Identifiable Information (PII). The SMS can batch emails, receipt paper questionnaires, or locate respondents. Blaise will not duplicate this capability.

A **Survey Handling Database (SHD)** should be provided by Blaise. The SHD database holds a representation of the sample and keeps track of what happens to each case. It does not hold any PII. The link between the SMS and the SHD is a unique Case ID. The SHD allows the institute to keep track of overall effort on a case, to prioritize strategies for a case or group of cases, and keeps track of costs. It manages status codes that represent **happenings** (events), **operational statuses**, **actions**, and maps survey history to **final disposition** codes.

A **Survey Handling System (SHS)** should be provided by Blaise. The **SHS** handles the communication between the **SMS**, the **SHD**, and other survey modules.

Figure 1 below gives a possible schematic of the SMS, the SHD, and other survey systems. The system portrayed would handle a Paper/CASI/CATI/CAPI survey. Figure 1 indicates the complexity of some modern-day multimode surveys. Subsidiary systems such as those for incentive payments or supervisory review are included because any of them can send a case back into the survey pool.

Figure 1: An Example Multimode Survey Management System<sup>1</sup>



The relationship between the **SMS**, the **SHD**, and the **SHS** is as follows:

- All Personally Identifiable Information (PII) is handled by the **SMS**, not the **SHD**.
- There is constant communication between the **SMS**, the **SHD**, and other modules via the **SHS**.
- Either the **SMS** is under the command of the **SHD** or it reports its actions to the **SHD**.

The database for **Survey Parameters - Adaptive Design - Stopping Rules** reflects Responsive and Adaptive Design. This is one of the means of parameterizing the Multimode Management System.

An institute-provided **data analysis** system assesses the stability of estimates for key variables for each designated domain. These estimates are compared to stopping rules in the **Survey Parameters** database.

<sup>1</sup> This is an idealized diagram. Details will differ for each institute.

An **action** is a term that describes what should happen to a case or group of cases. Examples of actions include putting a case on rest, stopping work on a case, or including a case in a CATI daybatch. An action can also be considered an instruction. **Actions** are determined by algorithms based on case history and survey parameters, or by manual intervention. The institute should be able to plug in its own algorithms.

A **Supervisory Review** utility allows for manual intervention. Such a utility should implement a hierarchy of supervisory levels and the actions allowed for each level.

A **happening** is 'what happened' in any of the system modules. Examples of a happening are a busy attempt in CATI, a log-in to a web authentication system, or a receipt of a paper questionnaire. This term **happening** is used to avoid confusion with the term **event** as it is used in the Blaise (and other) systems. **Happenings** can imply information that can be used to adapt survey management. For example, a web log-on, even if there is no survey progress, means that you have reached the respondent.

Four groups of status codes are at the heart of the Multimode Management System. They are **happenings codes**, **operational status codes**, **action codes**, and **final disposition codes**.

## 2. The Challenges of Multimode Survey Management

Multimode surveys have been around for decades. What has changed technologically, since about 1990, is the explosion of survey-taking and communication technologies. The number of ways an individual can be reached has multiplied. Thirty years ago you would primarily reach a respondent through a household. These days you primarily reach a respondent through individual multiple channels. The challenge therefore, is to manage all contact attempts, across all channels, in a way that maximizes the overall survey performance.

- This may mean that you reduce the probability of success with one person or set of people in order to increase the overall probability of success, especially taking into account the need to adequately cover all subgroups.

### 2.2 Multimode Survey Management Goals

A multimode management system must be able to manage conflicting survey management goals.

**Goal 1:** The surveying institute must stay within costs, within deadlines, and achieve an acceptable response. The institute must be able to change its survey-taking approach during the survey period through Responsive and Adaptive Survey Design.

**Goal 2:** The respondent must be able to distinguish legitimate survey contacts from other contacts, for example sales contacts. The Multimode Management System should estimate perceived burden on the respondent and pace contact attempts to diminish the feeling of burden over time.

**Goal 3:** Recognize when a respondent engages in a survey. This may be a completion, a partial completion, or some kind of refusal.

**Goal 4:** Coordinate contact attempts through mail, email, text messages, and phone attempts.

**Goal 5:** Manage the tension between optimal use of resources versus the staffing demands of interviewers and supervisors.

**Goal 6:** Achieve precision levels for subgroups such as sampling strata, or hard-to-interview populations.

## 2.3 Fielding Strategies

This report recognizes three kinds of fielding strategies.

**Sequential:** One mode is fielded first then another mode (or more) is fielded later.

**Concurrent:** All modes are open at once and the respondent can choose a mode.

**Blended:** This is a combination of the sequential and concurrent methods. For example, CASI and CATI may be available at the same time, but CATI only accepts call-ins at first.

**BCLUB members** use all three approaches. An institute might have a default approach, but most institutes need the ability to adopt any strategy depending on the survey. The following mode combinations were mentioned by the group.

- CASI/CATI
- CATI/CAPI
- CASI/Paper
- CASI/CAPI
- CASI/CATI/CAPI
- Paper/CASI/CATI
- Paper/CASI/CATI/CAPI

No matter which strategy is adopted, it is necessary to coordinate what happens to the case in each mode. For example, if a case is completed in one mode, it should be turned off in other modes. If a case is partially completed in CASI (for example) the institute should be able to complete it in CATI or CAPI. A refusal in any mode should stop all out-bound contact attempts, however, but leave open the possibility that the respondent self-completes (CASI or paper) later.

## 3. Contact Information

The institute SMS holds all contact information including PII. The Survey Handling Database (SHD) holds a representation of the SMS and links to it through a unique Case ID. The SHD will hold many indicator items and many counts.

### 3.1 Kinds of Contact Information in the SMS

Possible contact items include:

- Phone numbers with information about properties such as landline/cell.
- Street and/or mailing address
- Email addresses, and information about properties such as personal/business.
- Social media addresses

## 4 The Notion of Burden

If a respondent has multiple valid channels of communication, the individual can feel overwhelmed by receiving multiple messages from different sources. The multimode management system should include a burden indicator for contacting attempts. An extensive search of the survey management literature reveals few descriptions of how to measure burden. It is possible to define a burden indicator to each happening. Perhaps a time-lapse multiplier can reduce the value of the indicator over time so that rest periods can reduce the perceived burden.

## 5 Scope: Channels of Communication Considered

Five channels of communication are considered in this report. They are:

- Phone
- Personal visit
- Email
- Text
- Social media

Phone and visit are so tied to CATI and CAPI respectively, they are considered as part of the modes section below.

## 6 Scope: Modes Considered

The BCLUB multimode management group decided to focus on the following modes:

- CAPI,
- CATI,
- Web,
- Paper, and
- Device modes; this refers to the use of a smart phone or tablet in a stand-alone capacity.

## 7 Scope: The Line between Blaise 5 Functionality and Institute Capability

Blaise 5 cannot provide all survey management capability. It can provide a core functionality that is easy to modify to suit different kinds of surveys and different kinds of institutes. It is necessary to differentiate between an institute's Sample Management System (SMS) and a Blaise 5 provided Survey Handling Database (SHD).

The SMS is an institute-provided capability. It can be a database system or a spreadsheet. The SMS provides sample data to the SHD. The SHD handles the coordination between the channels of communication and data collection modes. Blaise 5 only needs to define and operate the SHD.

## 8 Scope: Kinds of Surveys – Record Types

It is striking how many kinds of surveys are handled in the Blaise community. Taken as a whole, there are survey management needs that cannot be met completely. Thus it is necessary to define the kinds of surveys that Blaise 5 can handle out-of-the-box.

The Blaise 5 multimode survey management capability should focus first on the **person-level** survey. The SHD should allow linking multiple respondents within a household.

Longitudinal surveys are commonly conducted by Blaise institutes. In this kind of survey, the institute keeps track of household members, valid contact items, sample attrition, and related data. The Blaise 5 SHD can handle a round of a longitudinal survey as if it is a single-round survey. It is up to the SMS to provide the latest valid information to the SHD.

## 9. Coding Schemes

An important extension of Blaise 5 survey management over Blaise 4 survey management is to provide coding schemes for **happenings, operational status, actions**, and a mapping to a **set of final dispositions** such as the AAPOR Final Disposition codes (AAPOR 20016). Further, it should allow an

institute to plug in its own schemes. Even further, the institute should be able to modify its schemes between surveys, or on the fly.

## 10. Responsive and Adaptive Design using Survey Management Paradata

For this part of the report, only paradata that relate to attempted contacts or inbound contacts are important. Other kinds of paradata, such as instrument audit trails, are not treated here.

Every contact attempt or inbound contact should be logged. For each contact attempt or respondent approach, a **contact record** should be recorded.

## 11. Survey Management Reports

Blaise should produce basic reports and also provide survey management data so that an institute can generate its own advance reports. To allow the institute to generate its own reports, the Survey Handling System (SHS) should provide the following information.

- Sample frame data that are useful for reports, e.g.,
  - Stratum
  - Location
  - Respondent demographic data

These are the sample domains important for Responsive and Adaptive Design.

- Values of key design variables
- Log of contact records
- Operational status codes
- Indications of group membership for each case, e.g.,
  - Groups of interviewers
  - Survey Language
  - Survey wave

## 13. References

The American Association for Public Opinion Research. (2016). *Standard Definitions: Final Dispositions of Case Codes and Outcome Rates for Surveys. 9th edition*. AAPOR. Found at: [http://www.aapor.org/Standards-Ethics/Standard-Definitions-\(1\).aspx](http://www.aapor.org/Standards-Ethics/Standard-Definitions-(1).aspx)

De Leeuw, E. D. (2005). To Mix or Not to Mix Data Collection Modes in Surveys. *Journal of Official Statistics*, 21, 2, 233-255.

Groves, R. M. and Heeringa, S. G. (2006). Responsive Design for Household Surveys: Tools for Actively Controlling Survey Errors and Costs. (c) Royal Statistical Society, *J. R. Statist. Soc. A* (2006) 169, Part 3, pp. 439-457

“Adaptive Survey Design”, by Schouten, B., Peytchev, A., and Wagner, J., © 2017 Chapman & Hall.

About 50 IBUC papers since the 1990s.

## Some Uses of Roles in Blaise 5

*G J Boris Allan, Westat*

A field can be presented in many different ways, and in Blaise 4.8 any field might have different standard ways in which it might be presented. That is, a field always had a field name, it might have a tag, it might have question text in one or more languages, and it might have a label or description. Datamodels would have titles, categories usually have labels, and the list continues.

In Blaise 5, there are six predefined text roles, and the user can extend this list to include as many as needed for the instrument being designed. Here are some text roles defined for one survey (and we might still extend the list):

```
ROLES =  
  HelpInt, HelpResp,  
  Watermark, Units, Template, ToolTip, EditMask, Width, QText,  
  PreInst, SAQ, QNum, InstInt, InstResp,  
  NaText, SaVisible, InitYear, InitMonth, InitDay
```

In the following we discuss three distinct examples:

- a simple extension of texts to present different information to users of disconnected (interviewer laptop) interviews and users of web browsers (self-administered questionnaires);
- extending the friendliness of numerical text boxes;
- using role texts to give an easily accessible way of collecting different types of information for comment review.

### 1. Interviews and Self-administered Questionnaires

We have a study that wants to start interviewing with an interviewer (an interview), and then possibly continue data collection with the respondent connecting to the survey by means of a web browser or a mobile device (an SAQ – self-administered questionnaire). The text for the questionnaire will, for many questions, be the same for the interview and for the SAQ, but there will be differences for some questions. It was decided that, for a question, there were certain functional areas:

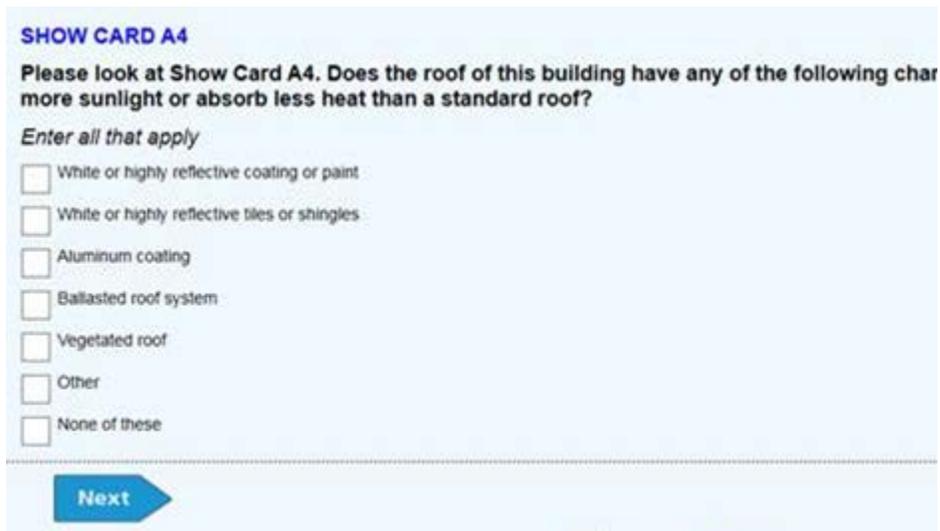
- Interviewer instructions prior to the respondent question (PreInst) not appropriate for an SAQ.
- A question asked of the respondent in an interview or an SAQ (the default).
- A question only asked in the case of an SAQ, if the default text did not make sense for a web question (SAQ).
- Interviewer instructions to the respondent on how to complete the answer (InstInt).
- SAQ instructions to the respondent on how to complete the answer (InstResp).

One other role (**QNum**) gives the question number displayed in case of problems – the tag is used to hold the SAS variable label, if applicable.

Here is the Blaise 5.4 code:

```
CoolRoof
  "Please look at Show Card A4. Does the roof of this building have any of the following
characteristics that
  allow it to reflect more sunlight or absorb less heat than a standard roof?"
  PreInst "SHOW CARD A4"
  InstInt "Enter all that apply"
  SAQ "Does the roof of this building have any of the following characteristics
that allow it to reflect more sunlight or absorb less heat than a standard roof?"
  InstResp "Please select all that apply"
  QNum "A11"
/ "Cool roof materials"
: SET OF
  (CoatingPaint "White or highly reflective coating or paint",
  TilesShingles "White or highly reflective tiles or shingles",
  Aluminum "Aluminum coating",
  Ballasted "Ballasted roof system",
  Vegetative "Vegetated roof",
  Other "Other",
  None "None of these")
```

Here is the screen produced for the interview question:



**SHOW CARD A4**

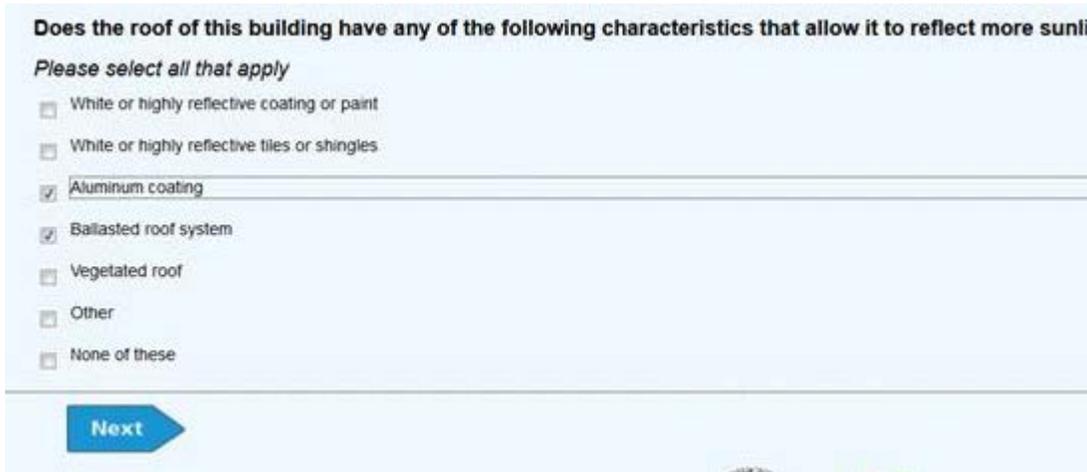
**Please look at Show Card A4. Does the roof of this building have any of the following characteristics that allow it to reflect more sunlight or absorb less heat than a standard roof?**

*Enter all that apply*

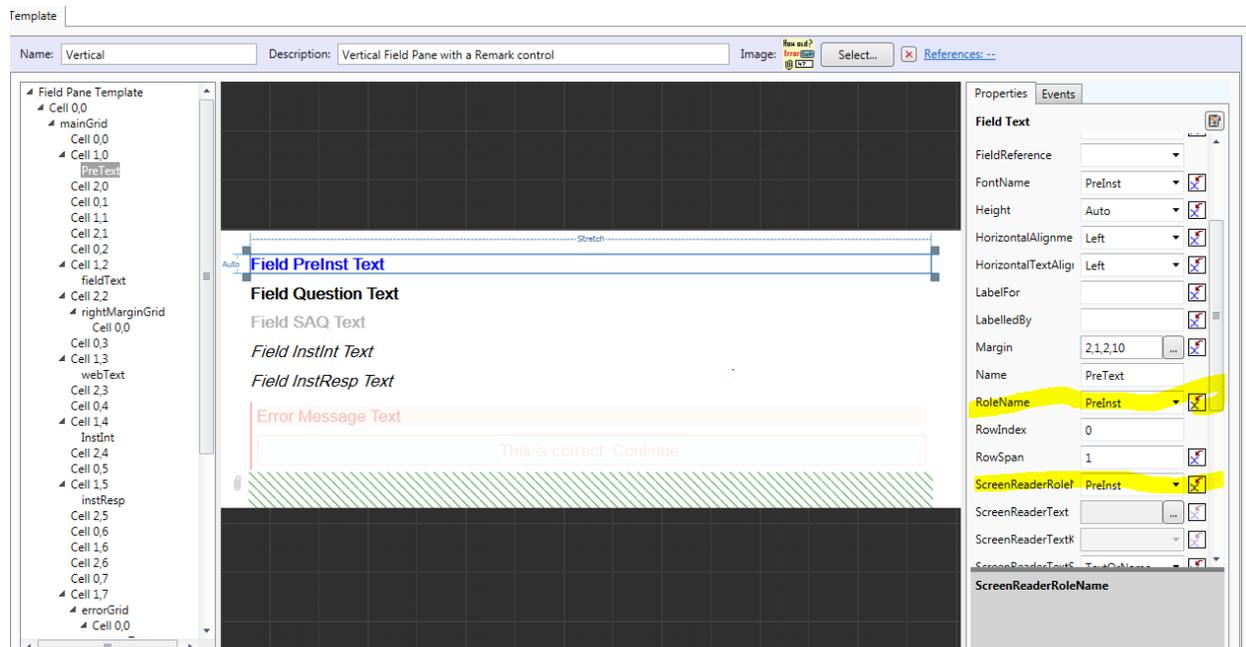
- White or highly reflective coating or paint
- White or highly reflective tiles or shingles
- Aluminum coating
- Ballasted roof system
- Vegetated roof
- Other
- None of these

**Next**

The same question in SAQ form is:



Let us take a look at the PreText control in Vertical Field Pane template in the resource database (the blrd file):



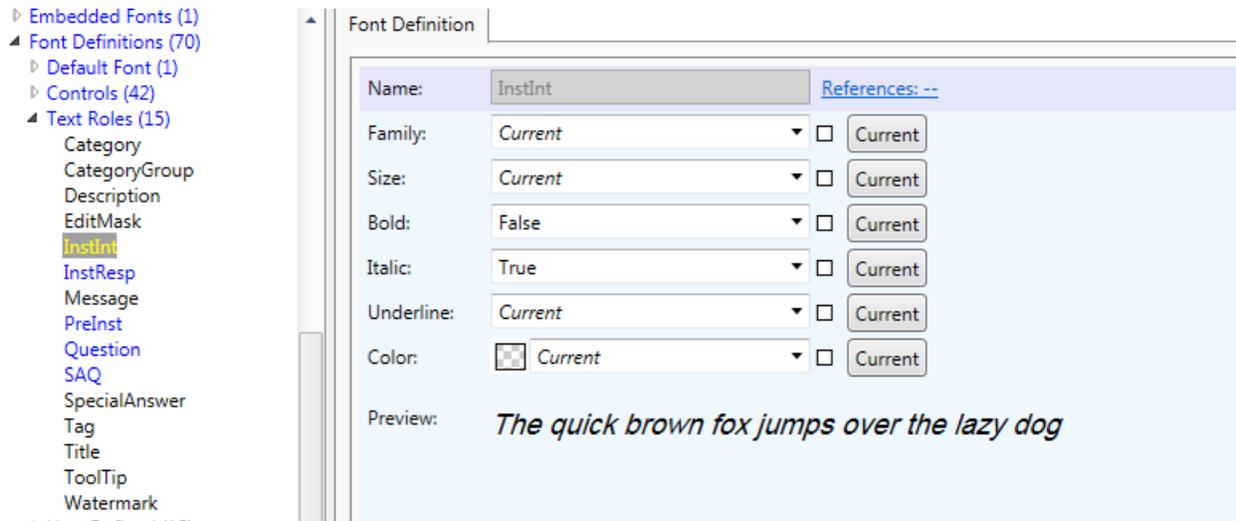
We can see that for the pretext control both the RoleName and the ScreenReaderRoleName properties are equal to the PreInst role, What controls whether we see the pretext control is its Visibility as defined by an expression:

```
IF LEN(Field.GetRoleText('PreInst')) = 0 OR State.IsCawi THEN
  'Collapsed'
ELSE
  'Visible'
ENDIF
```

That is, IF there is no role text for PreInst (the length of the text is zero) OR the the questionnaire is being viewed in a browser (IsCawi) THEN the control is collapsed, ELSE the role text is visible. All the other

field texts have similar expressions to control visibility. Note that the font format for PreInst has a different colour text by default, partly so that interviewer instructions are clearer to the interviewer.

Looking at the two controls InstInt and InstResp we can see that they are the reverse of each other (that is, both cannot be visible at the same time), and both have a different font format from standard question text – the font format is (like PreInst) defined in the resource database:

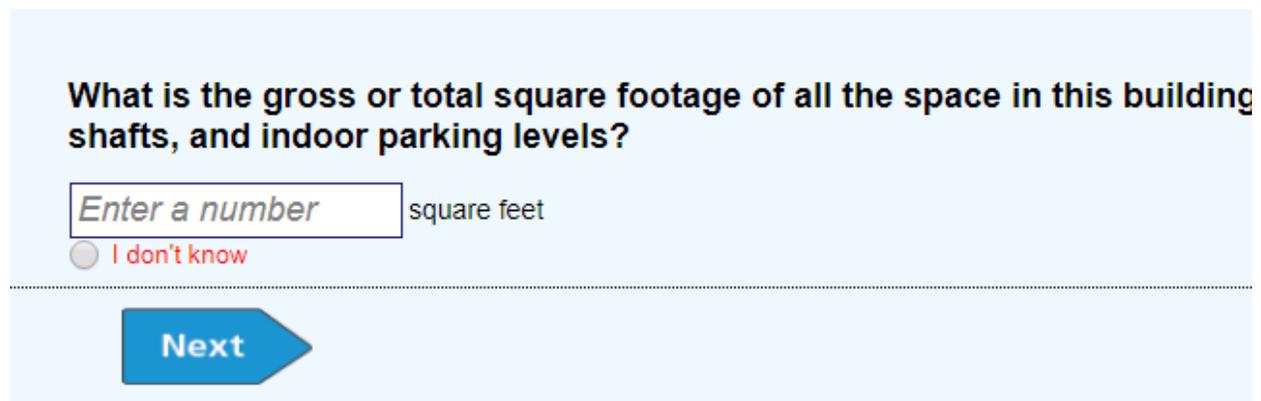


The expression for the visibility of InstResp is:

```
IF LEN(Field.GetRoleText('InstResp')) = 0 OR (NOT State.IsCawi) THEN
  'Collapsed'
ELSE
  'Visible'
ENDIF
```

## 2. Enhancing Data-collection Controls

Take a look at this question:

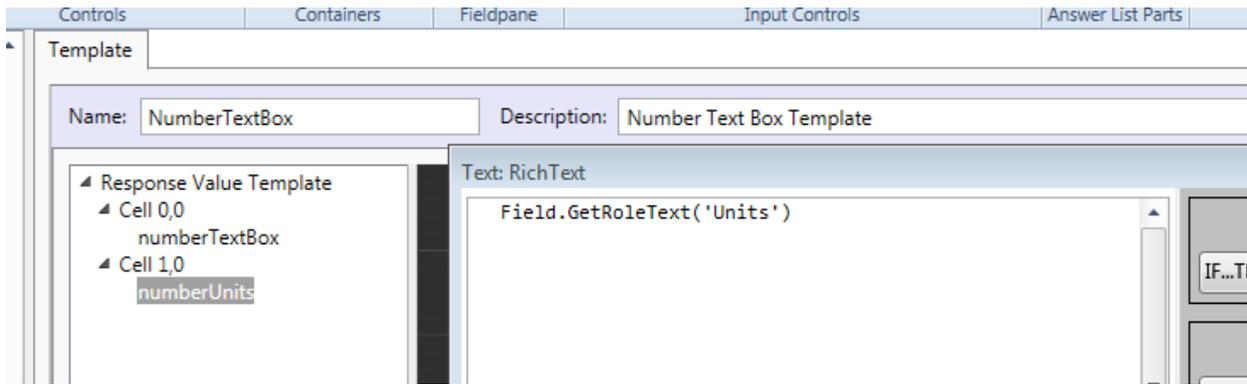


There are three points to note: there is a watermark (a standard Blaise 5 feature); there is a description of the units for the number following the entry box; and there is an explicit DK with the entry box (this question usually hides DK/RF when a question is first asked).

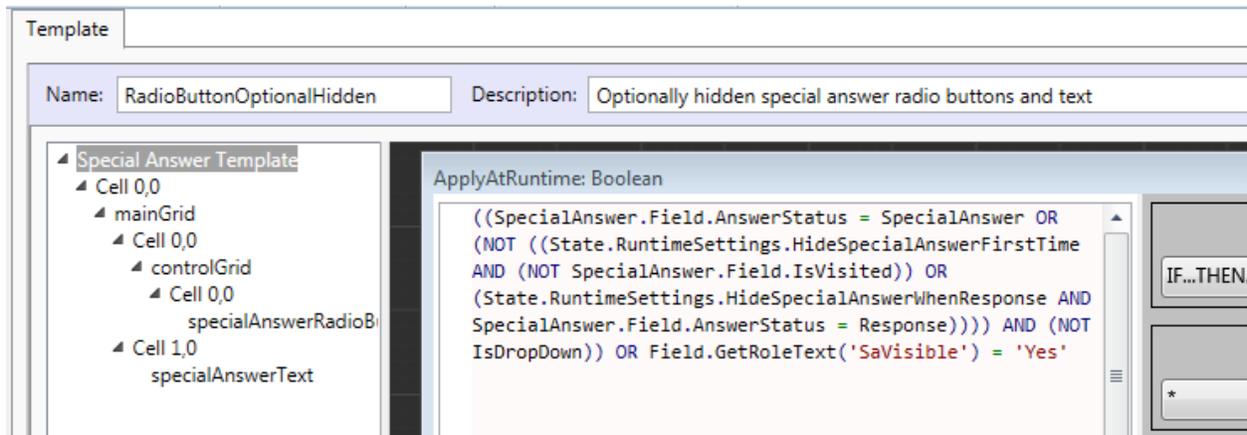
Here is the question code (note that fields in this instrument have DK and RF by default, which is why this field has NORF to switch off the refusal option):

```
SqFt (A6) "What is the gross or total square footage of all the space in this building
both finished and unfinished, including basements, hallways, lobbies,
stairways, elevator shafts, and indoor parking levels?"
Watermark "Enter a number"
Units "square feet"
SaVisible "Yes"
/ "Square feet"
: 1..99999999, NORF
```

A label (numberUnits) has been added to the template for a NumberTextBox, and the label text is taken from the Units text role – if the Units role does not exist for a question, the label is empty:



The appearance of the DK is controlled by the SaVisible text role, which is applied at run time to the RadioButtonOptionalHidden template (remember NORF):



### 3. Annotating interviews in Blaise 5

Take this page with two questions:

**Please enter the date this occurred, using the date-picker calendar.**

11/29/2017 

**Please enter the time this occurred.**

1:00 PM

We press F9 (the default for remarks is Ctrl-M, but we added F9):<sup>1</sup>

**Please enter the date this occurred, using the date-picker calendar.**

11/29/2017 

**Please enter the time this occurred.**

1:00 PM

**Information\_Source**

Observation  Respondent  OtherFamily

**Information**

That is, before we enter some information, we are asked to provide the source of the information.<sup>2</sup>

<sup>1</sup> Thanks to Ralph Dohlmans of Statistics Netherlands for his creativity in helping me explore this topic.

<sup>2</sup> Thanks to Peter Stegehuis of Westat for his work on sophisticated remarks in Blaise 4.8.

Later on, we have:

**Which of the following is true for you?**

Select one option only.

- One
- Two
- Three
- Four
- Five
- Six
- Seven
- Eight
- Nine

After F9:

**Which of the following is true for you?**

Select one option only.

- One
- Two
- Three
- Four
- Five
- Six
- Seven
- Eight
- Nine

**Comment\_Importance**

Low  Medium  High

**Comment**

---

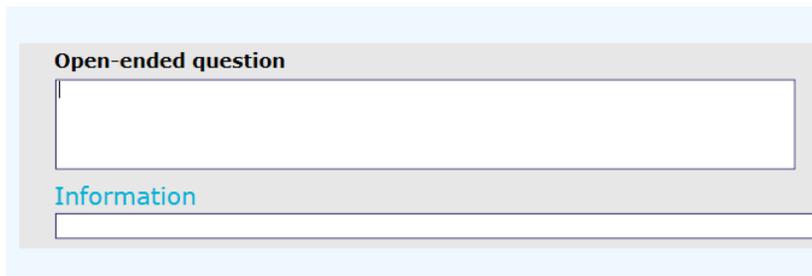
Before we comment we are asked how important is that comment.

A final example:

**Open-ended question**

---

After F9:



The image shows a screenshot of a survey form. At the top, there is a section titled "Open-ended question" with a large empty text box. Below this, there is a section titled "Information" with a smaller empty text box. The form is set against a light blue background.

This time we are simply asked for the Information, but not asked about the source.

For many other fields F9 has no effect.

An important introduction in Blaise 5 is the notion of field properties, and the beginning of this questionnaire code includes two significant sections – ROLES and FIELDPROPERTIES:

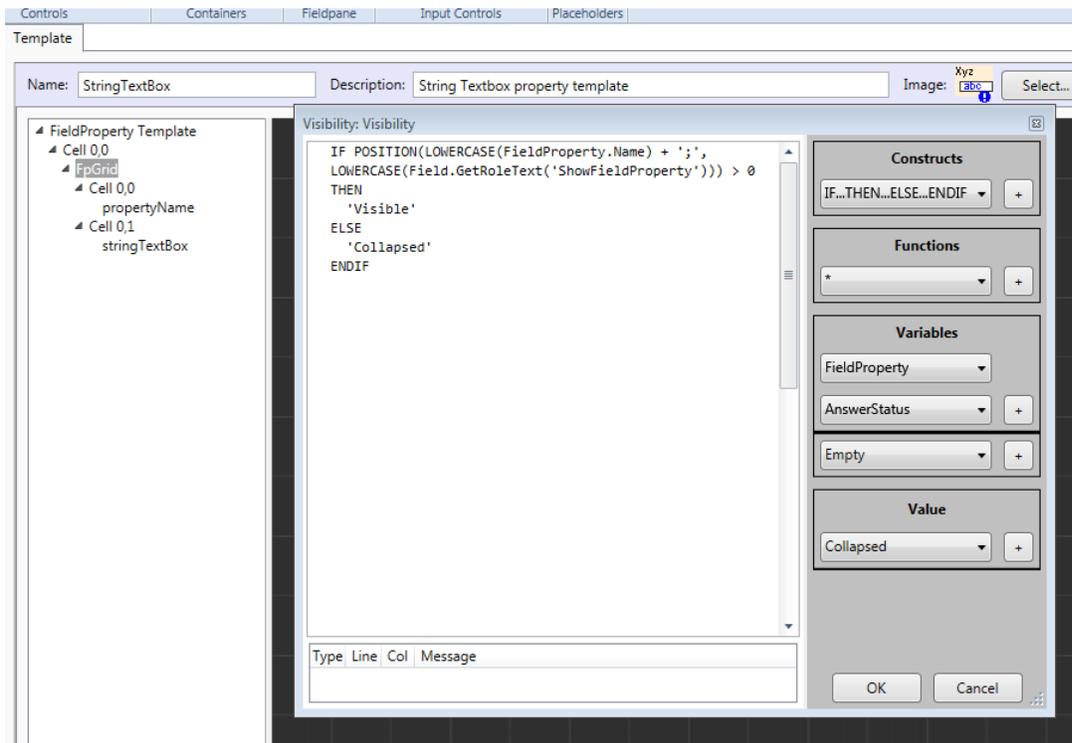
```
...
ROLES = HELP, ShowFieldProperty
...
FIELDPROPERTIES
  Comment_Importance: (Low, Medium, High)
  Comment: STRING[1023]
  Information_Source: (Observation, Respondent, OtherFamily)
  Information: STRING[1023]
...
```

In the FIELDPROPERTIES section there are definitions of the four types of remark we have encountered above and, to define which combination of remarks we show, we use the ShowFieldProperty role. For example, here is the definition of the time question:

```
TimeField ENG "Please enter the time this occurred."
  ESP "Introduce la fecha esto seleccionando la hora."
  ShowFieldProperty "Information_Source;Information;"
  / "Example time field"
  : TIMETYPE, DK
```

The ShowFieldProperty role specifies "Information\_Source;Information;", that is, show the information entry field and ask for the source of the information. If there is no ShowFieldProperty then F9 will not

have any effect on the display. So how do the field properties get shown? We go to the Field Property templates, and modify visibility for a field property listed in the ShowFieldProperty role text:



If a field property is not listed it is not shown.

It is of note that there is more to field properties than mere remarks. For example, we can filter on the information source:

```
IF (TimeField.Information_Source = Observation) THEN
  aString := TimeField.Information
ENDIF
```

And so forth.



# Web Screen Presentation Using Blaise 5

*Michelle Amsbary, Rick Dulaney, Justin Kamens, Westat  
Joelle Michaels, EIA*

## 1 Introduction: Challenges Moving CBECS to Web

This paper describes the challenges faced while adopting Blaise 5 for use on a web survey for the Commercial Buildings Energy Consumption Survey (CBECS). The CBECS periodically collects energy consumption and expenditure data from U.S. commercial buildings. This requires an extensive listing process to develop the sample frame, followed by about six months of field data collection. The survey is relatively long and complex, covering detailed aspects of building size and use, and energy consumption from several energy sources. There are additional complications when we encounter strip shopping centers or other variations on commercial buildings.

For several data collection cycles, dating back to 1999, CBECS has used Blaise for CATI and CAPI data collection. Beginning with the 2018 CBECS, the goal is to encourage respondents to complete their questionnaires on the web, and it is expected that approximately 40% of cases will be collected via web, with the remainder collected via CAPI. This transition to web data collection naturally suggests a move to Blaise 5. However, during the course of gathering requirements, the U.S. Energy Information Administration (EIA) and Westat identified several challenges around the conversion of a CAPI instrument to web. These challenges fall into two categories.

- First is the general look and feel of the instrument on the web: how to define elements of basic screen presentation, such as layout, font size, emphasis, branding, and color palette? How to handle graphics in question text or responses? How to position navigation buttons, and how to accommodate missing responses?
- The second category includes items more specific to fielding CBECS. For example, how should help screens be presented? How will we handle show cards, which interviewers usually use with long response descriptions? In CAPI, the screens often had optional text for interviewers to read under certain circumstances – how should we handle those on the web?

## 2 Process: Best Practices Working Group

When CBECS 2018 adopted Blaise 5, EIA and Westat had a great deal of experience with in-person data collection, but little experience with this instrument on the web. EIA and Westat formed a team to examine best practices for web screen presentation on CBECS. The purpose of the best practices team was to identify the key questions raised by web data collection, examine different solutions, and propose recommendations. The team used the following methods to accomplish the goal:

- The team immediately embarked on a set of discussions with expert survey methodologists at Westat and EIA. This allowed us to take a detailed look at all the elements of the design, think critically about what was on each screen, and consider carefully the effects of collection mode on data quality.
- At the same time, we conducted a literature review looking for best practices regarding web screen layout. Much of the literature consisted of suggestions or general ideas rather than concrete templates. We assembled these into a set of discussion points with our expert methodologists.

- We surveyed screen designs currently available on the web, generally from commercial services primarily engaged in providing survey capability.
- We developed an initial “style sheet,” synthesizing our findings from the literature and from commonly used screen designs into an initial specification that could be applied to a web survey.
- We compiled a series of example questions that covered the range of question types present in the CBECS questionnaire. This included the basic question types: categorical, continuous, string, date, and dollar amounts, as well as more sophisticated or combined question types such as full addresses, other-specify, select all that apply, lookup tables, grids, and calendar date pickers. We identified the resulting 44 items as our evaluation questionnaire, and we used this evaluation instrument to develop, test and refine our Blaise 5 presentation style.

### 3 Challenge: Survey Look and Feel

As mentioned above, we began by looking for examples of web surveys in the literature and on the web, and found a wide variety of practice. For example, Figures 1 and 2 show a typical categorical question for two of the most popular commercial web survey tools, Survey Monkey and YouGov.

Figure 1. Categorical Question in Survey Monkey (desktop)

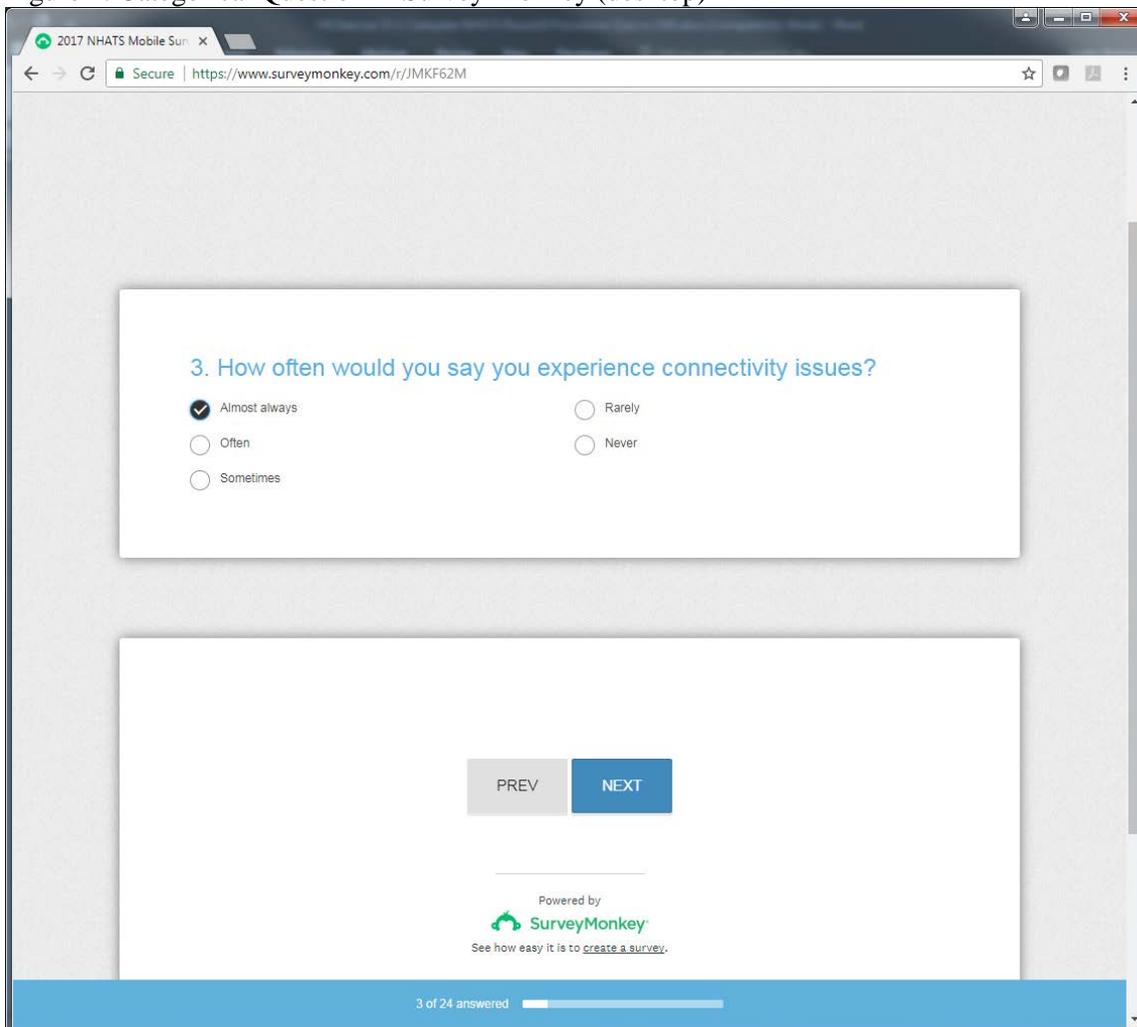
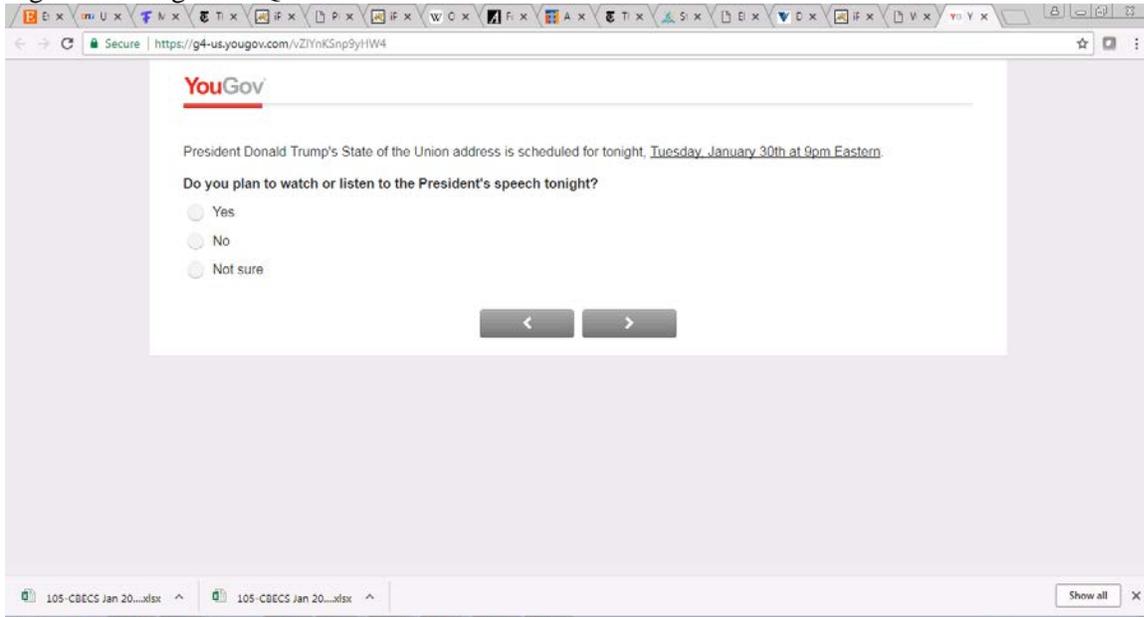


Figure 2. Categorical Question in YouGov



We also reviewed other screens built for data collection with web technology at Westat. Figure 3 shows an example of a screen using Blaise 4 with Westat Visual Survey, a proprietary extension of the Blaise rules engine that runs using web technology. Figure 4 shows an early version of a Blaise 5.2 screen. Figure 5 shows a sample screen using SurveyBuilder, another Westat system for web surveys.

Figure 3. Blaise 4 with Westat Visual Survey extension

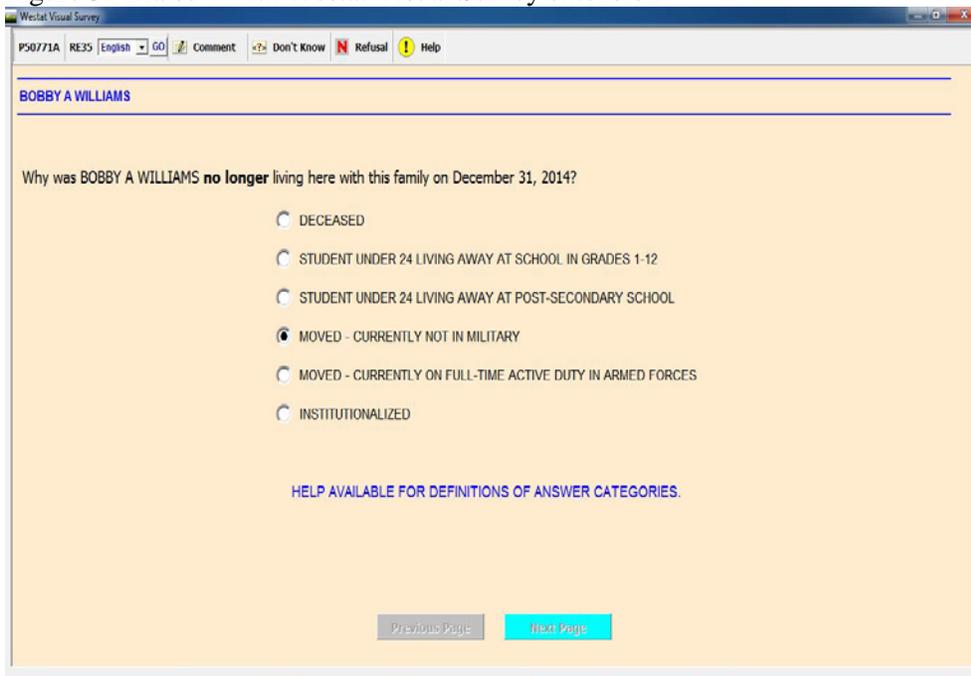


Figure 4. Early web screen using Blaise 5.2

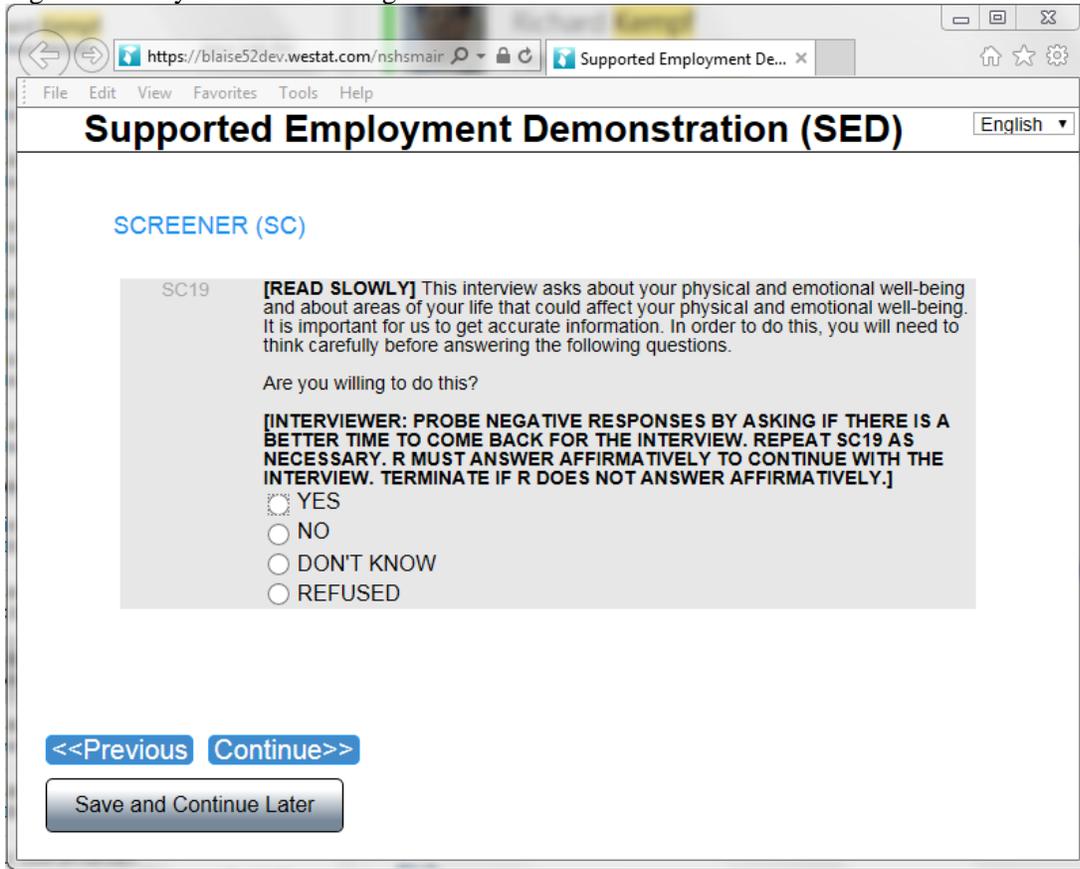
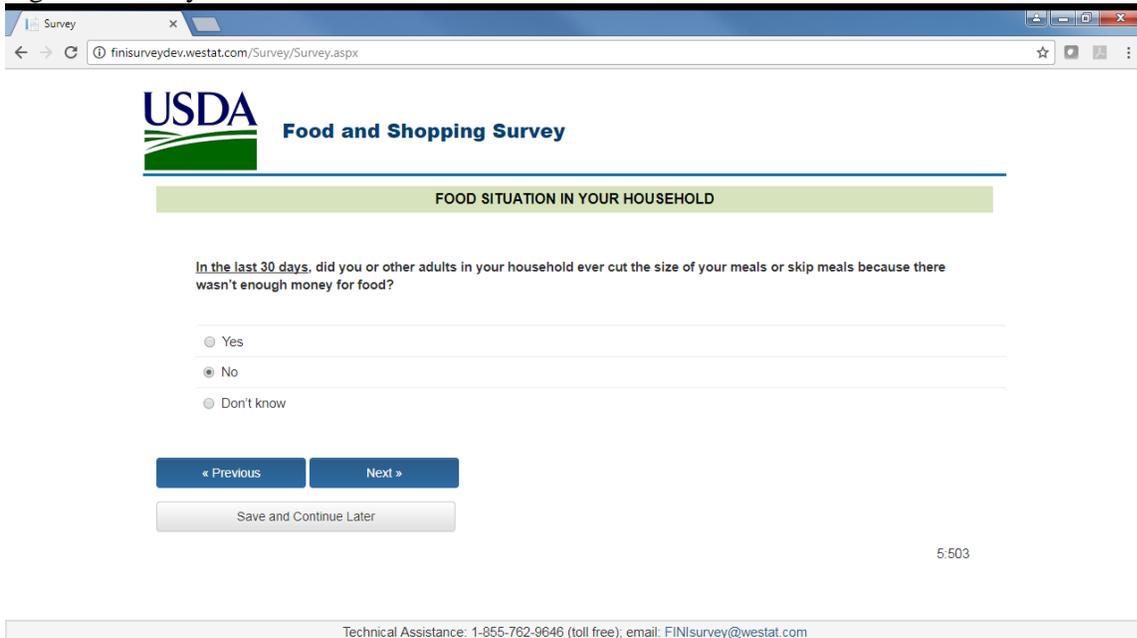


Figure 5. Survey Builder



As you can see, the approaches to web survey screen design are many and varied. We worked through the literature and discussed different approaches internally and with our expert methodologists. We also programmed several different approaches, and conducted iterative reviews with the group and senior project staff. By the fifth review cycle, we had a basic screen design that we felt comfortable taking into usability testing; Figures 6 and 7 show examples for categorical and continuous screens.

Figure 6. Sample CBECS screen for categorical item

Commercial Buildings Energy Co. x +

← → ↻ https://cbeecs2018demo.wesdemo.com/evaluation5/ ☆

**CBECS Commercial Buildings Energy Consumption Survey - Evaluation5** A9 :: Wall construction material

Which one of these categories best describes the predominant exterior wall construction material used on this building?

- Brick, stone, or stucco
- Pre-cast concrete panels
- Concrete block or poured concrete (above grade)
- Aluminum, asbestos, plastic, or wood materials (siding, shingles, tiles, or shakes)
- Sheet metal panels
- Window or vision glass (glass that can be seen through)
- Decorative or construction glass (glass that cannot be seen through)
- Other

Next

Back

Save and Continue Later

Figure 7. Sample CBECS screen for continuous item

Commercial Buildings Energy Co. x Commercial Buildings Energy Co. x +

← → ↻ https://cbeecs2018demo.wesdemo.com/evaluation5/ ☆

**CBECS Commercial Buildings Energy Consumption Survey - Evaluation5** A16 :: Number of floors

How many floors are in the tallest section of the building, including basements, parking levels, or any other floors below ground level, but excluding attics, half-floors, mezzanines, balconies, and lofts? If you're not sure, please provide your best estimate.

Enter a number floors

Next

Back

Save and Continue Later

Every part of these screens have been reviewed in some depth.

- The general look was suggested by the literature: a light background with dark text was easiest to read. Key elements of the basic look-and-feel include
  - Question text and response options are left justified and allowed to wrap
  - The text area is horizontally centered on the screen
  - Fonts in priority are Helvetica, Arial, and Sans Serif (different browsers and operating systems have access to different font families)
  - Question text is size 18 and bold, instructional text is size 18 and normal, response options are 16 and normal
- CBECS, like many projects, already had a graphical presence used for brochures and other materials, so we adapted that for the banner across the top of every page. The banner includes the project logo, and the white text on dark background contrasts with the area containing the question text and responses.
- We felt that including the official logos of the government offices supporting the survey gave the screen credibility.
- The literature and the opinion of our experts also held that we should avoid extensive use of color so it is sparing, muted, and consistent. The most prominent part of the question should be the question text – thus it is bold and somewhat larger than the response options. Question text is size 18 and bold, instructional text is size 18 and italicized, response options are size 16 and normal, and explanatory text is size 14 and italicized.
- The response options should be easy to click – therefore the radio buttons and check boxes are slightly oversized, and the entire text area is clickable. We experimented quite a bit with larger and smaller radio buttons and determined this size – slightly larger than the height of the text but with definable space between each button – worked best.
- For continuous questions, we took advantage of the watermark feature as shown in Figure 7.
- As you can see, the question is separated from the navigation buttons by a thin horizontal line.
- In our early versions, we always had the navigation area “pinned” to the bottom of the screen. However, with certain questions and browser settings, the response options may not all be visible, and the respondent may not realize that there are additional possible responses. So we let that area “float” with the question text and responses so that the respondent must scroll to the bottom, and therefore see all possible responses, before moving to the next screen.
- The Next and Back buttons went through several iterations. The goal was to provide a backup capability but draw the user to the Next button as a default. One recommendation in the literature was to align frequently-used buttons on the left side of the screen, within the respondent’s line of sight while reading the response options. When we placed the Previous button on the left and the Next button on the right, consistent with the way we read text, we found that the eye was drawn to the Back button and we had to work to locate the Next button. After some trial-and-error, we preferred the Next button in the most prominent position, and made it more prominent with a lighter colored button. We decreased the prominence of the Back button by moving it under the Next button—consistent with some recommendations that this can discourage respondents from unintentionally clicking Back—but kept it slightly to the left of the Next button. This also allowed us to keep the more frequently-used navigation functions together, and to provide substantial distance from the “Save and Continue Later” button which is much more rarely used. We also experimented with text-only links (i.e., no directional arrows) and with text-free arrows, and we preferred the combined approach shown here with text included on the arrow shape. The font size is 14, consistent with the smallest of the other screen elements.

## 4 Challenge: CBECS Design Elements

The second major challenge in moving CBECS to the web was to accommodate design elements that had been present in the CAPI screens. One very important consideration was that the 2018 CBECS was going to be conducted via CAPI and web, and the design needed to accommodate both modes while minimizing mode effects. The goal of the best practices group became to design an approach that could be used on both modes, and in fact, we decided to use a single application in both CAPI and web modes.

### 4.1 Show Cards

In CAPI data collection it can be quite common to use show cards, that is, hard copy display materials that contain more information, such as extra descriptions or definitions that may not fit on a screen or may not be easily understood orally if read aloud by an interviewer. CBECS show cards may contain lengthy categorical responses, detailed explanatory text, or pictures that best describe the desired response. For example, we used to ask about the pitch of the roof, providing text responses in CAPI and example images on a show card (Figure 8). We did not want to include show card content only in CAPI mode, as we felt this could introduce an unacceptable mode effect. We briefly considered making a show card document available electronically to web respondents, but dismissed this as too unwieldy and confusing. Ultimately, after reviewing all show card text to shorten it as much as possible, we adjusted the text on the screens in both modes to include the show card text, and we continue to provide show cards to field interviewers for use in CAPI interviews. Figure 4.2 shows the revised version of the roof tilt example described above, with images now included on both the CAPI and web screens.

Figure 4.1 Example of show card used for 2012 CAPI interviews

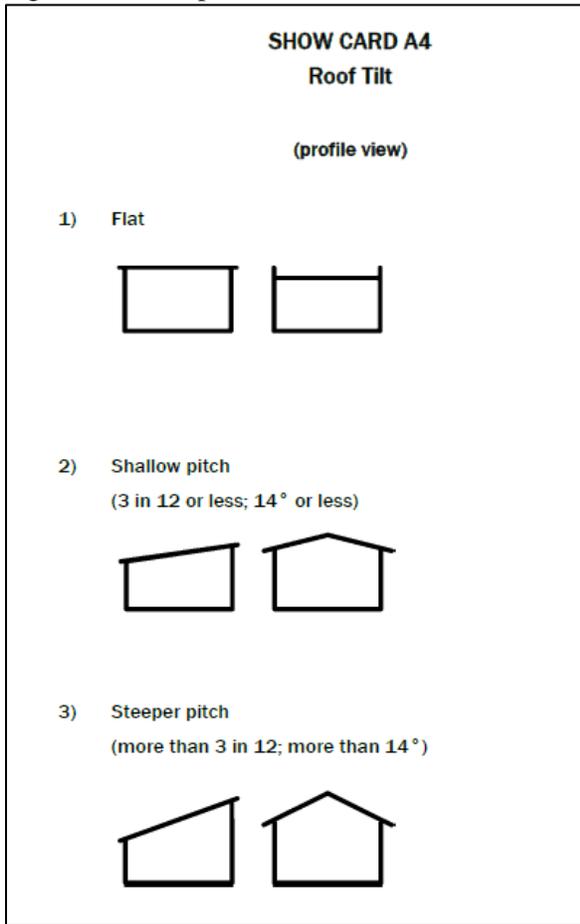
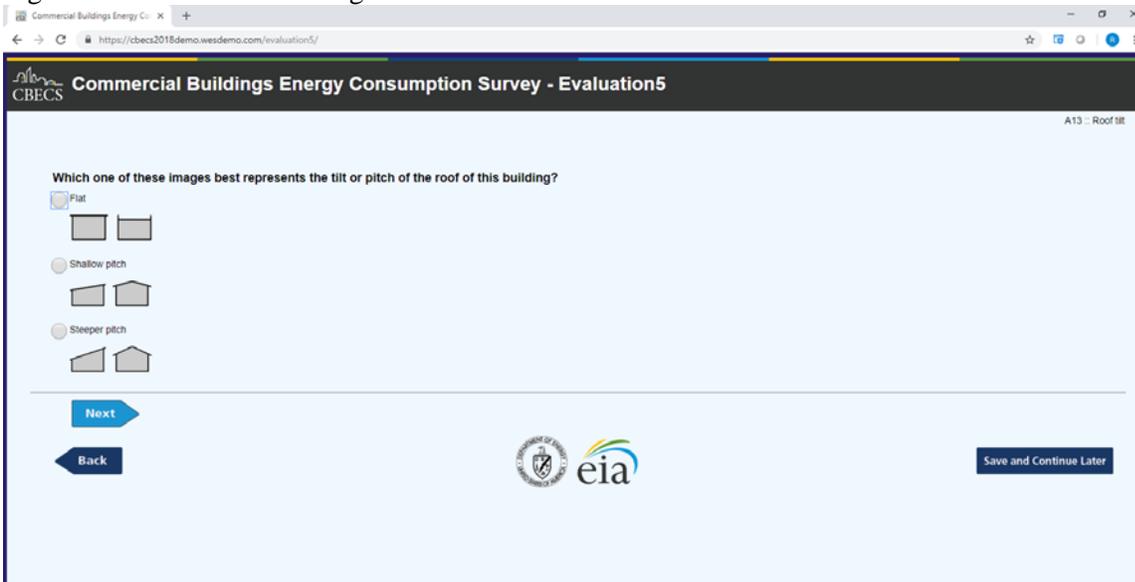


Figure 9. Show card text integrated onto screen



## 4.2 Question-specific help

Question-specific help was historically available throughout the CBECS questionnaire, and this information is particularly important as the instrument asks for fairly detailed technical responses. In previous CAPI data collection, we used the F1 key to allow interviewers to bring up more complete explanations, sometimes referred to as Q-by-Qs. As we moved to the web, we initially programmed messages that appeared when the cursor hovered over particular terms. These “tooltips” worked very well with mouse-based browsers, but were not available if the respondent chose to complete the questionnaire using a mobile device. After discussion and refinement through trial-and-error, we eventually settled on a blue question mark for both web and CAPI modes. Figure 4.10 shows the additional text when the button next to “Public assembly” is pressed. This functionality is used sparingly, however; reviewing the 2012 data on the incidence of help screen use showed that most were not used much, so most of them were removed and any content from help screens that we felt was particularly important was moved into question text.

Figure 10. Question-specific help text

Commercial Buildings Energy Consumption Survey - Evaluation 5

B1 - Primary activity

Which one of these activities accounts for the majority of the floorspace in this building?  
*If there are equal shares of floorspace for more than one activity, select any of those activities here.*

- Office/Professional
- Data center
- Warehouse/Storage
- Food sales or service
- Enclosed mail
- Retail (other than mail)
- Education
- Religious worship
- Public assembly
- Health care
- Service
- Lodging
- Public order and safety
- Residential
- Industrial
- Agricultural
- Vacant
- Other

Buildings where people gather for entertainment, cultural, recreational, or social activities. Examples: art gallery, exhibit hall, library, museum, casino, concert hall, night club, observatory/planetarium, stadium/arena/coliseum, theater/cinema, amusement arcade, bowling alley, community center, gymnasium/health club, indoor racquet sports, recreation center, indoor swimming pool, skating rink, assembly hall/auditorium, convention center, funeral home, lecture hall, lodge hall, meeting hall, student union, senior center, town hall, armory, airport terminal, train/bus station.

Next

Back

Save and Continue Later

eia

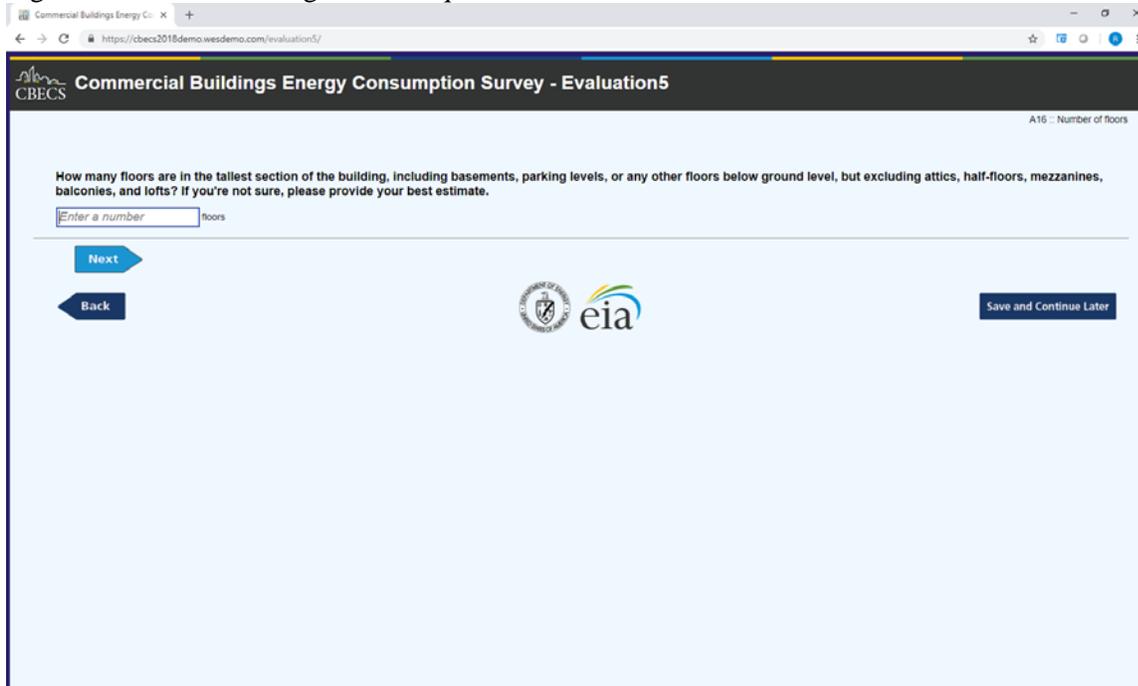
## 4.3 Optional explanations and definitions

Previous versions of the CBECS sometimes included interviewer instructions, that is, additional text to guide the interviewer that was not read to the respondents. In some cases these were additional definitions, and in other cases instructions on how to handle specific situations. Clearly these would not function properly in web mode, so we either integrated them with the question text, converted them to question-specific help, or eliminated the text altogether.

## 4.4 Probing

Another staple of CAPI data collection is the use of probes, generally in the form of interviewer instructions providing optional text if necessary to clarify the question for the respondent. We needed a different approach for web respondents, and in general we adjusted the question text to include the probe. Figure 11 shows a typical example. In previous CAPI versions, we did not include the final sentence with language about estimating; rather, we included optional interviewer text instructing to probe for an estimate if the respondent was unsure.

Figure 11 Probe text integrated into question



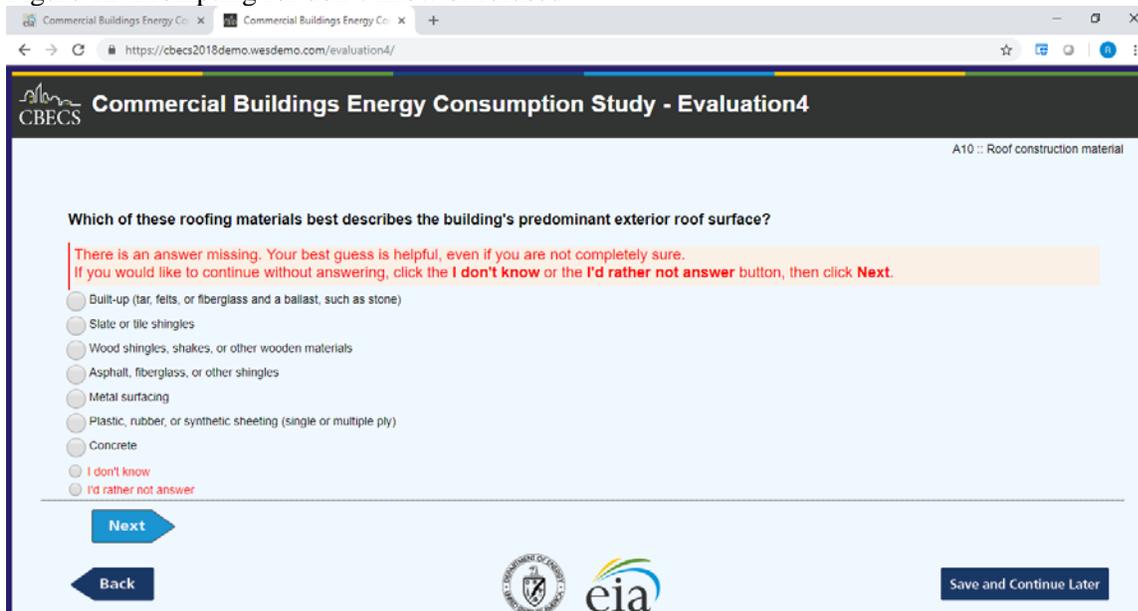
The screenshot shows a web browser window displaying a survey question. The browser's address bar shows the URL <https://cbecs2018demo.wesdemo.com/evaluation5/>. The survey title is "Commercial Buildings Energy Consumption Survey - Evaluation5". The question is: "How many floors are in the tallest section of the building, including basements, parking levels, or any other floors below ground level, but excluding attics, half-floors, mezzanines, balconies, and lofts? If you're not sure, please provide your best estimate." Below the question is a text input field with the placeholder "Enter a number" and a "floors" label. Navigation buttons for "Next" and "Back" are visible, along with a "Save and Continue Later" button. Logos for the U.S. Department of Energy and EIA are also present.

## 4.5 Item nonresponse: don't know and refused

One of the most challenging issues of moving the CBECS to the web was how to handle don't know and refused responses. In CAPI, the interviewers have special function keys to accommodate these responses, but the literature and expert review all agreed that we should not routinely supply "Don't know" and "Prefer not to answer" on each screen.

Our first solution was to assume that a respondent who did not know or did not care to answer would simply attempt to move past the item without answering. So we trapped those empty fields and prompted the respondent to answer using the categories or to mark don't know or refused if he or she clicked next without responding, as shown in figure 12. This approach covers respondents who inadvertently moved past the question as well as those who don't know or prefer not to answer.

Figure 12 Prompting for don't know or refused



However, a complex questionnaire often has more complex needs for missing data. For some items, we decided the best approach was to reverse course and include don't know as an explicit option when the respondent first arrives on the screen. We did this where we specifically wanted to measure respondent knowledge of a concept, or where we would rather follow up with a categorical item than accept an estimate. For instance, in Figure 13, we would rather not have a guess; a don't know response will skip to a categorical question more in line with the analytic needs of CBECS. We developed a similar approach for explicit refusals. This is the only time we were really interested in the distinction between a don't know and a refusal, so if the respondent left blank an item that did not have an explicit don't know or refusal, we just wanted to make sure the question was not missed in error, rather than press the respondent to report don't know or refusal. The revised approach is shown in Figure 14. A fourth nonresponse treatment is to allow the respondent to leave a response blank without displaying a prompt message. This will be used in limited circumstances, for example in the case of some "other/specify" responses.

Our approach to missing data caused us to add a new missing value – SK for "skipped" – in addition to the DK and RF values. If a respondent leaves an item blank, receives the missing value prompt as shown in Figure 14, and presses Next with no response, the item gets the SK attribute. The same is true if the respondent leaves blank an item with an explicit don't know or refusal. This allows us to know that the question was shown to the respondent and that he or she declined to respond.

Figure 13, Explicit don't know

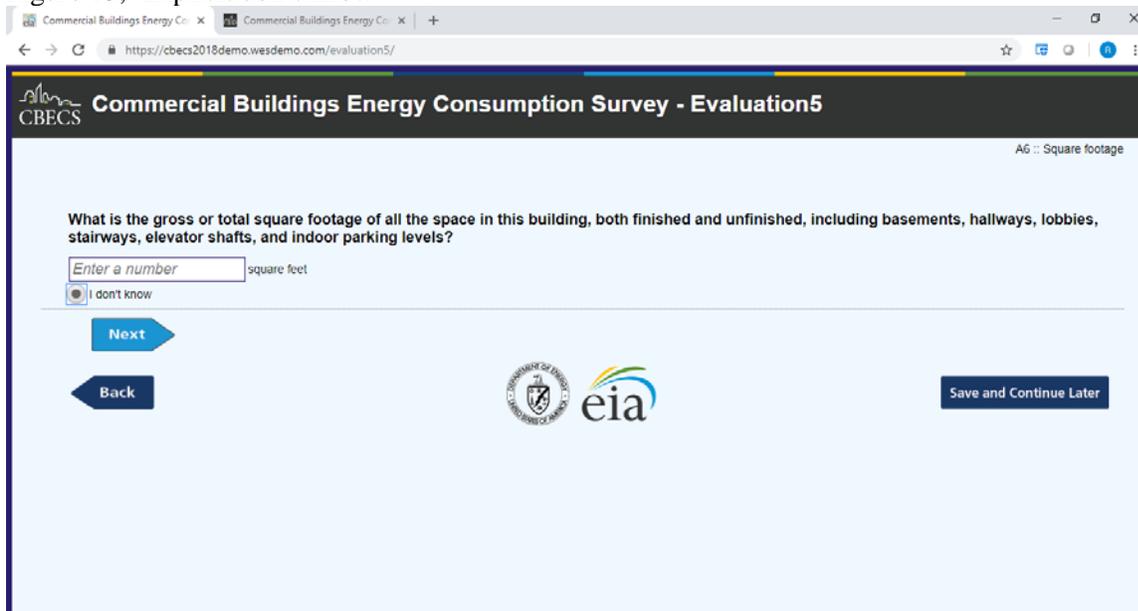
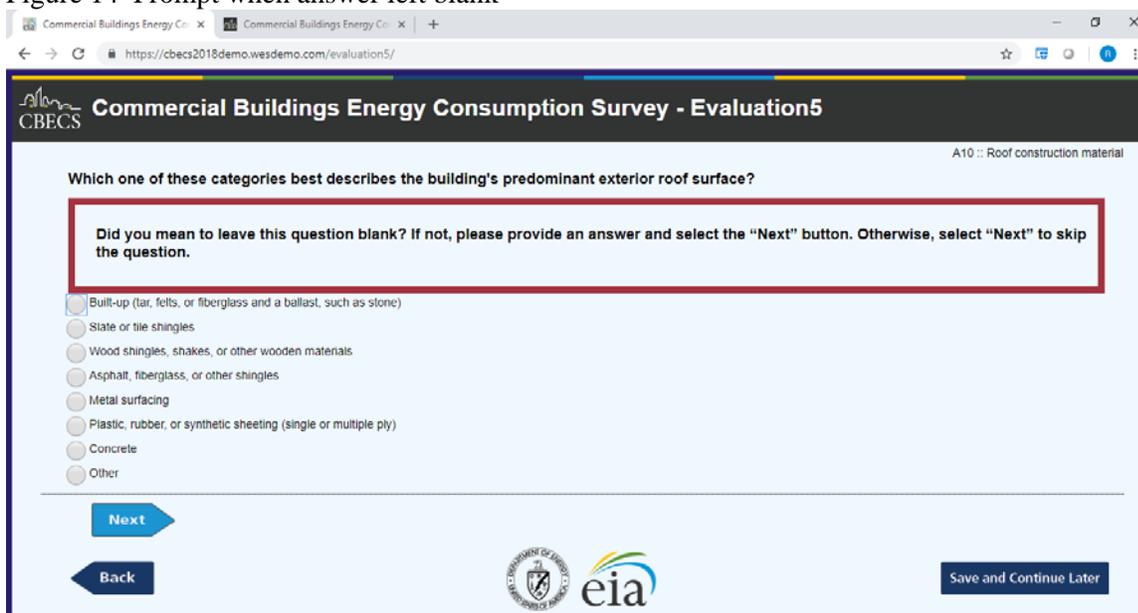


Figure 14 Prompt when answer left blank



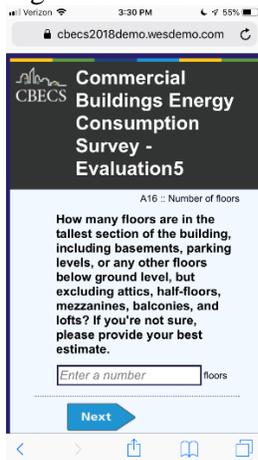
## 5 Additional Lessons Learned

In addition to the detailed learning on web screen presentation, this process taught us lessons in two additional areas.

## 5.1 Mobile

We understand that some respondents will only complete a web interview using a mobile device. This may be a personal preference, or the respondent may not have ready access to an Internet-connected desktop computer. Our preference is that the interview be completed using a desktop computer because of the complex and detailed nature of the questionnaires, but we recognize that this may not always be possible. Accordingly, the web screens may be optimized for the desktop but must also function properly on a mobile device with a much smaller form factor. Fortunately, Blaise 5 adapts very nicely to smaller form factors, and we were able to implement a working version of the web questionnaire for mobile devices very quickly. Figure 5.1 shows an example of the CBECS presentation on the mobile form factor.

Figure 15. Mobile web screen



## 5.2 Changes to specifications

A related lesson is that we needed to adjust our specifications to accommodate new requirements imposed by web data collection and new capabilities afforded by Blaise 5. In some cases, the text needed to change between web and CAPI modes. We needed to add specifications to contain information text behind the blue question marks, and we needed to specify the watermark text. We also needed to add the correct treatment of missing data to each item.

## 6 Conclusion and next steps

In general, the process we developed to analyze, design, and adapt screens from CAPI to web administration using Blaise 5 worked quite well. We were able to take advantage of deep experience on the project, we gave ourselves enough time to review and refine several iterations, and we addressed all of the major issues we could identify. Our next steps are to perform some formal usability testing and continue to improve the application incrementally based on conclusions from that testing. We feel confident that these screens will allow us to collect data that is of high quality when the study launches in early 2019.



# Developing organization level screen layout and design guidelines for Blaise 5

*Karl Dinkelmann, Shane Empie, Rebecca Gatward, Lisa Holland, Andrew Hupp,*

*Survey Research Center, University of Michigan*

## 1. Background

The Survey Research Center (SRC), at the University of Michigan, originally developed screen design guidelines for Blaise survey instruments in 2000 - these guidelines were then reviewed and updated in 2002 and 2008. The benefits of establishing organization level screen design guidelines have been well documented in earlier papers (Couper (2000), Gatward (2003), Hansen (2003), Kuusela (2003), Wensing (2003)). In addition a series of papers presented by the Survey Research Center outline the process and design principles that were followed when the Screen Design Guidelines, Specification and Programming standards were initially developed (Hagerman 2009, Hansen 2003, Hansen 2004).

Over the past five years SRC has been testing new releases of Blaise 5. As the method to design screen layouts is completely different in Blaise 5, the necessary programming or settings cannot be rolled forward from Blaise 4.8 into Blaise 5. We had to start from scratch and develop screen design templates in Blaise 5. For this reason, it was an ideal time to review and update the screen design and layout guidelines. In addition, we wanted to take advantage of the increased flexibility for screen design that Blaise 5 provides.

The focus of this paper is to detail the steps we followed to review and update SRC's long-standing guidelines for Blaise 5, including decision making from both a technical and as well as a design perspective and lessons learned. Although the scope of the project included updating guidelines for web surveys, this paper focuses just on the work necessary to develop the screen templates for interviewer administered modes.

For context, the templates were developed as we transitioned the Health and Retirement Study to mixed mode and Blaise 5. This work began in 2012 and mixed mode data collection launched in April 2018 using Blaise 5.3.1501.

## 2. Scope and goal

The full scope of the work undertaken is broader than developing screen design and layout guidelines for Blaise 5 in interviewer administered modes. Those relevant to Interviewer administered modes are as follows;

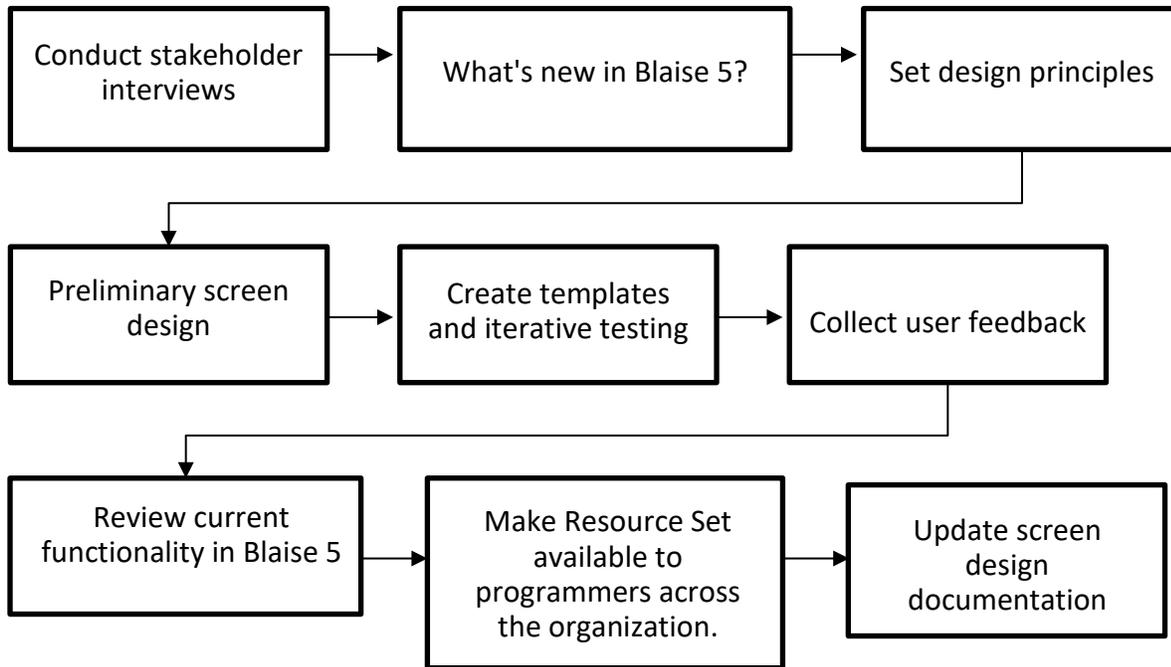
- Update the Blaise screen design guidelines for Blaise 5.
- Create a library (or Resource Set) of Blaise 5 screen templates (interviewer administered, self and mobile) to be used across SRC projects. The creation of the templates includes creating an organization level Resource Editor Database.

## 3. Review and design process

The process we followed to review and update the Blaise screen design guidelines is described below, and summarized in the flowchart (Figure 1). It is a standard process, there is nothing particularly innovative

about it, although some additional steps were necessary because we are adapting to and creating guidelines in Blaise 5.

Figure 1 - Summary of the process followed to review and update the screen design guidelines.



### 3.1 Conduct stakeholder interviews

The first step was to conduct interviews with stakeholders with the aim of collecting ideas about potential improvements to CAI screen designs. Individuals selected (n=5) to be interviewed were identified with the intention of representing various perspectives and projects with respect to Blaise and CAI instrument development. They included faculty, interviewer managers, and other project staff with considerable experience in specifying and testing Blaise questionnaires.

### 3.2 What's new in Blaise 5 – or what can we now do using Blaise 5?

Blaise 5 allows more flexibility around controlling the look of a screen and thus more complex design is possible. This step involved gaining familiarity with the screen design process in this new version of Blaise and the additional functionality available. We also needed to identify layouts that we were not able to achieve using previous Blaise versions (or required workarounds) and then assess if these were now possible using Blaise 5.

### 3.3 Set design principles

At this stage we set the parameters we would work within as we updated the guidelines. These design principles were established as we reviewed the feedback provided during the stakeholder interviews (they are described in sections 4 and 5 below).

### 3.4 Preliminary screen design

Using the design principles we created a prototype screen. This was simply an initial screen for the team to react to.

### **3.5 Review requirements across current surveys**

We reviewed surveys conducted by SRC to establish a comprehensive list of question types and screen components that are used across surveys. This served as a check list to ensure new templates were created to meet all project needs.

### **3.5 Creation of templates and iterative testing**

Once we had agreed on a basic design and developed a prototype using Blaise 5, we began developing templates for all the basic question types.

### **3.6 Collect user feedback**

We were now ready to begin ‘in house’ testing and collect feedback from various Blaise users - including, Field Managers, Interviewers, Blaise programmers, and Faculty/experts in screen design. Feedback was collated and reviewed and the programmer implemented necessary changes to the templates.

### **3.7 Review current functionality in Blaise 5**

Development of Blaise 5 continued as we were working through the development process which meant that it was necessary to review more recent versions of Blaise 5 as they were released, for additional functionality relevant to this work.

### **3.8 Make templates available to programmers and other users across the organization**

A primary goal of this project was to achieve consistency for the user. To encourage the use of the guidelines across all SRC projects we developed a library of templates that can be applied to Blaise questionnaires by programmers.

### **3.9 Update the screen design guidelines**

The final step is to update the current documentation which details the screen design and layout guidelines.

## **4. Outcome of the stakeholder interviews and resulting design decisions**

### **4.1 Common interview themes**

There were several general themes that were raised during all or most of the stakeholder interviews. These are summarized below. For reference, an example question in the current Blaise 4.8 design guidelines is included below (Figure 2).

- Current CAI (4.8) screen designs are based on sound principles, and people generally felt that the current screen designs were effective as is.
- Users recognized several areas in which Blaise 5.0 functionality is not yet to the level of Blaise 4.8. Users assumed that any previous capabilities would eventually be available, but their absence was a dominating topic in the present interviews. (*The Stakeholder interviews were conducted in May 2016*)

- As an organization, we need to improve our consistency in screen presentation and functionality both within and across interviewer- administered data collection instruments. Issues of consistency were primarily related to design and implementation (rather than Blaise functionality).

## **4.2 Initial Design recommendations**

Each of the following items was discussed in the majority of the interviews. Most of these items speak to the need for improved consistency in design. All functionality is already present in Blaise 4.8, and is expected to be available in 5.0 if it is not already.

### **Screen panes**

There was overall agreement that the three distinct panes for the question, response categories, and entry fields should be retained. Most agreed that the size of the panes should remain flexible to best accommodate the question content.

### **Entry pane**

The entry pane should remain gray and continue to display relevant items adjacent to the present questionnaire item. Users agreed that the presence of additional items provided context for the interviewer regarding skips and the relationship between items, as well as confirmation that responses were entered correctly.

### **Entry pane labels**

Users agreed that the labels on the left of the entry pane should be descriptive, intuitive representations of the survey questions, rather than cryptic labels or variable names.

### **Entry field labels**

Users also agreed that it is most beneficial to the interviewer when the response category text is displayed to the right of the entry field after the interviewer keys the response.

### **Question pane**

A new question should always begin in the same place on the screen so that the interviewer does not need to scan for the question text. This may vary, however, if an interviewer instruction is required before the interviewer begins reading the question.

### **Definition (QXQ) placement**

Ideally, important definitions are incorporated into question text. Any additional definitions that are displayed on the screen should be displayed in a consistent location. Any definitions that are not displayed on the screen should be displayed in a consistent location (triggered by a short-cut key) within the question pane.

### **Question batteries/series**

When a question series includes a common stem with a single construct changing within the question, the construct unique to the question should be highlighted (or displayed in another color) so that the interviewer can easily identify the unique part of the question.

### **Consistent navigation**

Users were in agreement that specifications needed to be developed for consistent keyboard navigation across projects. Specific items included the use of hot keys, DK/Ref conventions, arrow keys, page up/down (or other forward and back navigation), saving open-ended responses, and functionality for pick lists (and plan for drop-down equivalents in web).

Figure 2. Basic screen formatted with existing guidelines

1 SRO Standard Blaise Project Template

2 Forms Answer Options Help

3 Do you currently use a computer, either at work, at home, or at school?

4  1. Yes  
 5. No

5 Currently Use a Computer  No  
Used the Internet  
Ever Used a Computer   
Vacation in Last Year   
Regions Visited  
Regions Specified  
Value IRAs   
Own or Rent Home

6 B1 1001 09/12/2007 12:16:57 PM Version Date: 09/05/2007 Version Time: 2:10PM READBACK MODE

- 1. Current project
- 2. Drop down for user tasks
- 3. Question pane
- 4. Response categories
- 5. Data entry pane
- 6. Status bar

#### 4.3 New design suggestions

The following items were discussed in all the stakeholder interviews, and there was no disagreement (although some ambivalence) about these suggestions.

- **Paired items/two-part questions**  
Users unanimously agreed that it would be preferable for two-part questions to have both parts of

the question displayed on the same screen, with both entry fields visible at the same time. Figure 3 provides an example of a commonly used format.

Figure 3 – example two part question

On average, how often you do participate in vigorous exercise?

\_\_\_\_\_ Times

Per

- 1 Day
- 2 Week
- 3 Month

- **Related questions**

Similarly, there are question pairs or series where the same group of questions is consistently asked together. Presenting these questions together would provide context for the interviewer in asking the question, recording, and providing feedback. Figure 4 shows a closed question with an open question follow-up that could be displayed on the same screen.

Figure 4 – example related question (closed with open question follow-up)

Do you think that prices will continue to rise over the next 5 years?

- 1 Yes
- 5 No

Why do you think so? (Open-ended)

The users agreed that not all questions which could potentially be formatted as a grid should be presented together, but that there are examples of scales or series where this presentation makes sense, as in the interference scale in figure 5. Longer “grids” however, are generally best presented as single items with the common text in parentheses, assuming that there is no increase in performance time that results from many separate items.

Figure 5 – example of a grid with all items presented together

Using a scale from 0 to 10 where 0 means no interference and 10 means very severe interference, what number describes how much your fear of social situations interfered with each of the following activities over the past month?

	Number (0-10)
a. ...your home management like cleaning, shopping, and working around the house or yard?	_____
b. ...your ability to work?	_____
c. ...your ability to form and maintain close relationships with other people?	_____
d. ...your social life?	_____

- **Additional suggestions**

The following items were mentioned in only one or two of the interviews.

**Highlighted text** - two users commented on the importance of highlighting the current item for the interviewer if multiple items are presented on a single screen.

**Navigation buttons** - one user suggested removal of the buttons to minimize, maximize, and close the application. They suggested including only a “Quit” button.

## 5. Design principles and implementation

### 5.1 Design principles

Using the stakeholder feedback we established the following key design principles to guide our decision making as we designed the new screen layouts:

- Maintain core design elements from the existing Blaise 4.8 screen guidelines
- Enforce consistency across study instruments. Facilitate improved adherence to the guidelines through the design of the templates and their implementation - i.e. create guidelines that are well designed, comprehensive and easy to implement.
- Take advantage of the flexibility around screen design that is provided in Blaise 5 to eliminate need for work arounds necessary in Blaise 4.8.

## 5.2 Implementation

With these principles in mind, our initial direction was to recreate the existing design in Blaise 5. It became clear, however, that this was not the right direction. By doing this, we were not using the functionality available in Blaise 5 in the most efficient way and we were trying to ‘contort’ Blaise 5 into behaving in the same way as previous versions. We were able to create a basic design for simple questions but we were not able to find a solution for all question types or one that could efficiently be applied to multiple surveys without much additional programming. At this point we paused to assess the issues and decide how to move forward.

After some discussion we agreed to begin working on a new basic template that was simple to create using Blaise 5 but followed the key design principles we had set.

### 5.2.1 Prototype screen design

The screen design we agreed on and we began developing for all question types is shown in figure 6 below.

Figure 6 - Question formatted according to the update screen design guidelines

The screenshot shows a survey question interface. At the top, a dark blue header contains the text 'GIT 2018 - The American Study of Interesting People' and the logo of the Institute for Social Research, Survey Research Center, University of Michigan. Below the header is a task bar with buttons for 'Suspend', 'DK', and 'RF'. The main content area is a yellow box containing the question: '(In the last month, how much did you receive in...)' followed by 'Any dividend or interest income, investment income, or royalties?' and a prompt '• ENTER dollar amount' above a text input field. Below the yellow box are three inactive question & response boxes: 'Last Month Income Other', 'Individual Income', and 'Household Income 11'. At the bottom, a status bar displays technical information: 'Sample ID: 2398011 | Version Date: 21-3-2018 | Version Time: 13:00:00 | Current Date: 31-8-2018 | Current Time: 11:25:59 | V.1.6 | Section\_C.C4j'. Five numbered callouts point to these elements: 1. Current project and organization/project logo; 2. Task bar; 3. Active question & response; 4. Inactive question & response; 5. Status bar.

1 GIT 2018 - The American Study of Interesting People

2 Suspend DK RF

3 (In the last month, how much did you receive in...)  
Any dividend or interest income, investment income, or royalties?  
• ENTER dollar amount

4 Last Month Income Other  
Individual Income  
Household Income 11

5 Sample ID: 2398011 | Version Date: 21-3-2018 | Version Time: 13:00:00 | Current Date: 31-8-2018 | Current Time: 11:25:59 | V.1.6 | Section\_C.C4j

1. Current project and organization/project logo
2. Task bar
3. Active question & response
4. Inactive question & response
5. Status bar

The prototype screen design retains the following basic elements from the Blaise 4.8 design -

- Light background color and dark text to provide sufficient contrast and allow the question to be the dominant element on the screen.
- Interviewer instructions are still presented in blue text and question (or 'read aloud') text in Black. Although we have increased the text size and change the fonts.
- On screen text is still displayed to mirror the required interviewer tasks.
- Task related action verbs are displayed in bold blue text.

The following global changes were introduced -

- The screen is no longer split into sections – questions, instructions and the corresponding response categories are all displayed in a single 'active question' pane. Once a question has been answered this question pane collapses – a short variable description and response is still visible.
- The data entry field is positioned to the left of the response categories.
- Responses are no longer displayed in the lower half of the screen. They are still available for the interviewer to refer to within the box that displays the 'inactive' question.
- Interviewers are provided a mouse-driven alternative for all actions that were previously just controlled by short cut keys. These include suspending an interview, referencing additional question level guidance or entering a 'don't know' or 'refusal' in the data entry field (Figure 6).
- The study title is clearly shown in the top right hand side of the header banner. Any additional organization level logos can be shown on the left hand side (Figure 6).
- The additional question level help is displayed below the question and response categories but within the question pane – rather than in a pop-up text box.
- Bold is now used to indicate if text should be emphasized when read (underline was used previously).
- Checks and signals appear on screen in a pop-up box, the box for a check is shaded in red and signals shaded blue (Figure 7).

Figure.7 Example signal at a question level

Health and Retirement Study

INSTITUTE FOR SOCIAL RESEARCH  
SURVEY RESEARCH CENTER  
UNIVERSITY OF MICHIGAN

Health and Retirement Study

Suspend DK RF English

• Is R living in a nursing home or other health care facility?  
(Are you living in a nursing home or other long-term health care facility?)

DEF: By "nursing home or other health facility" we mean a facility that provides all of the following services for its residents: 24-hour nursing assistance and supervision, dispensing of medication, personal assistance, and room & meals.

1  1. Yes  
 5. No

**Active Signal**  
You have reported that this R (is/was) living in a nursing home (or hospice).  
If this is NOT correct, suppress this message and select "No".  
If R (is/was) living in a nursing home (or hospice), suppress this message and proceed with the interview.

Suppress Close

TYPE OF HEALTH CARE FACILITY

The new design allows easy implementation of each of the additional suggestions that were raised during the stakeholder interviews. Specifically, it is now possible to display all questions in a two (or more) part question or related questions on one screen (figure 9).

Figure 8 – example multi-part question

SRC Screen Design Guidelines (V.1)

INSTITUTE FOR SOCIAL RESEARCH  
SURVEY RESEARCH CENTER  
UNIVERSITY OF MICHIGAN

SRC Screen Design Guidelines (V.1)

Exit DK RF

How much do you earn now from this job?

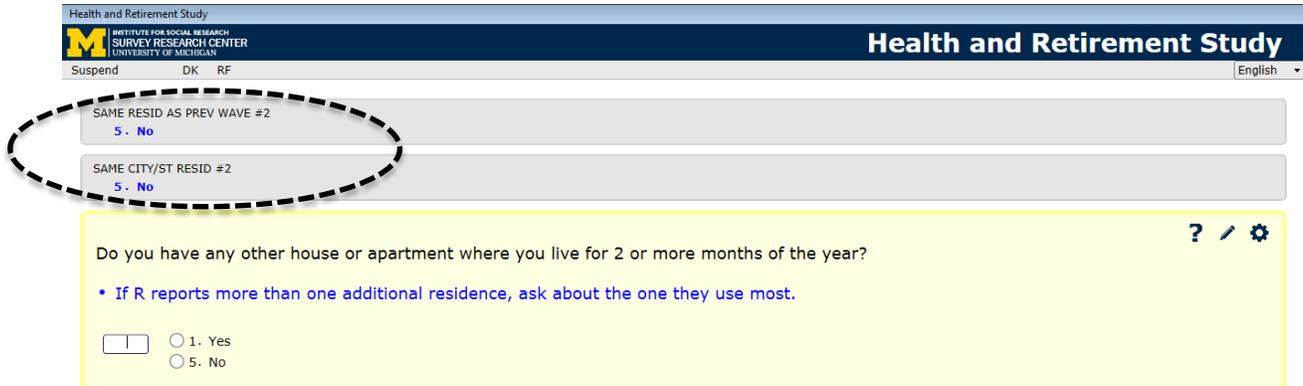
7.25

Per:

1  1. Hour  
 2. Week  
 3. Two weeks  
 4. Month  
 5. Year  
 6. Other -- specify

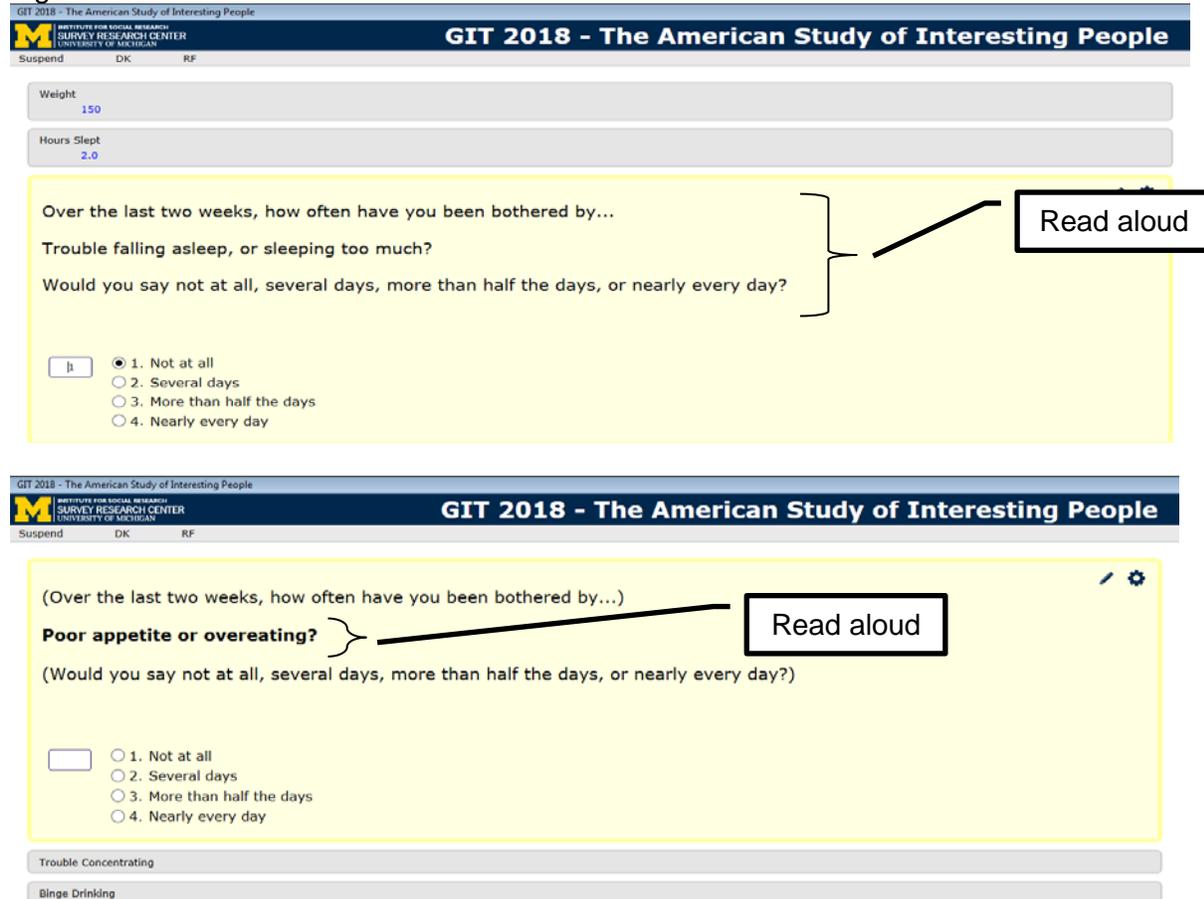
The active/inactive design naturally focuses the interviewer on the question text they should read (Figure 9) by graying out the inactive panes and displaying previously recorded responses in blue.

Figure 9 – example of an active question



In an attempt to make it easier for interviewers to identify which text they should be reading when they arrive at one of a series of questions with the same stem question, we have highlighted, in bold, the ‘read aloud’ text (Figure 10), retaining the use of parentheses for the stem question or response options, which the Respondent has already heard.

Figure 10



## 6. Implementation

The guidelines were operationalized by creating a series of templates for question types. These were created and stored as a resource set in the Blaise Resource Database (BLRD). Initially, the templates were assigned to fields using role text or applied based on question type. When the LAYOUT function was added to a newer version of Blaise 5, these were used to assign templates to questions.

Many of the challenges we faced as we created and applied the screen templates were due to project schedules; we found ourselves developing the templates in parallel with Blaise 5 development. The project schedule did often not allow us the time to wait for necessary bug fixes or additional planned functionality to arrive, so it was sometimes necessary to create 'workarounds'. Developing expertise in screen design using Blaise 5 also required dedicated time - often a luxury when a project is in the preproduction stage.

The following are some specific examples of issues we faced -

- **Look up lists vs Drop downs** - Our preference was to use look-up lists for long code frames (like, US States) for Interviewer administered modes and Drop down lists for equivalent lists presented in self-administered modes. We encountered a couple issues with the look-up lists - firstly, entering data on the screen triggered an on screen error which disappeared after a few moments, but gave the impression to the interviewer that they had made a data entry error. The second issue was found in the data - in some cases the name of the state had been saved but the corresponding code had not, and vice versa.
- **Number of questions on a page** - Interviewers faced considerable performance issues as they worked through some sections of the first production project. The slower transition between questions is a general issue but is more noticeable on questions that have more complex routing or where more items are displayed on screen. For this reason we were hesitant to add more complexity to some questions or display more questions on one screen.

## 7. Next steps

The new screen design has been well received by Interviewers. Keeping key elements of the screen unchanged has meant it was a straightforward transition for interviewers working on questionnaires programmed using Blaise 4.8 and Blaise 5. There were some items that required focus at training - for example, using a different key combination to back up through the questionnaire and placing 'dashes' between responses in multi responses question rather than using the space bar. We will continue collecting and reviewing feedback from interviewers about the new designs.

There are some question types that we have not yet tackled. These include 'Roster' or 'Grid' type questions. The first of the SRC projects to adapt to Blaise 5 redesigned these type of questions, however before this design is adopted as the preferred design and recommended in the guidelines, we need to review interviewer feedback and the data collected during the first months of data collection on this project. (This new design is described in a paper by *Ostergren, J. (2018)* to be presented at this conference)

As the new guidelines are applied to more surveys we will learn more about the process and how to make this process more efficient through instructions and enhancements to the method used.

Finally, as Blaise 5 continues to develop and new versions are released, we will need to review these new versions for additional functionality.

## 8. References

Couper, Mick P. 2000 Development and evaluation of screen design standards. Paper presented at the 6<sup>th</sup> International Blaise Users Conference, Kinsale. May, 2000

Gatward, Rebecca. 2003 Developing and updating screen layout and design standards. Paper presented at the 8<sup>th</sup> International Blaise Users Conference, Copenhagen. May, 2003

Hagerman, James, R et al. 2009 The three-legged stool is now a four-legged chair: specification guidelines for Blaise survey instruments. Paper presented at the 12th International Blaise Users Conference, June, 2009

Hansen, Sue Ellen et al. 2003 Screen Design Guidelines for Blaise Instruments . Paper presented at the 8<sup>th</sup> International Blaise Users Conference, Copenhagen. May, 2003

Hansen, Sue Ellen et al. 2004 Programming Guidelines for Good Data Documentation. Paper presented at the 9<sup>th</sup> International Blaise Users Conference Gatineau, 2004

Kuusela, Vesa. 2003 Screen Layout Standards at Statistics Finland. Paper presented at the 8<sup>th</sup> International Blaise Users Conference, Copenhagen. May, 2003

Wensing, Fred et al. 2003 Developing an optimal screen layout for CAI. Paper presented at the 8<sup>th</sup> International Blaise Users Conference, Copenhagen. May, 2003



# Using Blaise 5 in CAWI

*Rogier Hellenbrand, Statistics Netherlands*

## Introduction

The presenter, Rogier Hellenbrand works for CBS (Statistics Netherlands) in the IT Department and is currently involved in the Phoenix Program as information analyst. His team is directly involved in development and maintenance of the Computer Assisted Web Interviewing (CAWI) channel.

This paper will cover the following topics:

- First I will give you an introduction on the Phoenix+ program and why we chose to use Blaise 5 as our instrument for our main survey method CAWI
- After that I will provide you with a quick overview of our solution to integrate Blaise surveys in our business process
- Then I will elaborate on some (fairly) new features of the Blaise suite for which we built specific functionality within our CAWI channel, as well as discuss some challenges we encountered along the way.
- And finally I will give you our judgement on the Blaise product.

## Phoenix+ and CAWI

The Phoenix+ program was launched in 2015 with the expressed goal of replacing several data collection legacy systems by one, new application landscape. These legacy systems are technically fragmented, outdated, file based and often offline. In other words: they are end of life.

CBS' policy is to use CAWI as preferred interviewing method, because it is the most cost efficient. Philosophy is to start with CAWI and to use other modes only if necessary to get statistically valid results. So the CAWI channel is of the utmost importance. The architectural guidelines for our new CAWI channel were:

- Fully web based (multi platform, multi device) → for his purpose we decided to switch to Blaise 5 questionnaires
- Storage in database (SQL) instead of file based
- Generic for all types of surveys (social and economic)
- Message-based communication with other domains

Our first experience with using Blaise 5.x was in 2014. For a large production survey (80.000 companies were sent a questionnaire) we created a new CAWI channel, called PS-online. Special in this case was that we generated 12 different templates with in total 250 different routings using meta data from an old legacy system. Automated generation of templates based on meta proved to be possible, but it was quite challenging at the time.

In 2015 we started with the Phoenix+ program and reused lots of the PS-online software. But because the CAWI channel had to fit into the overall domain landscape, numerous alterations were necessary.

The first stage of Phoenix+ was completed august 2016. At present our KPI's are as follows:

- 30 of 125 surveys migrated to Blaise 5 surveys
- 600K+ records per year, response rate around 20 - 40% (for social surveys. For economical surveys where companies are obliged to respond, the response rate is much higher, up to 100%)
- Only 4 surveys need CAWI functionality which is not yet fully developed

In the future we will expand to other modes, probably using Blaise 5:

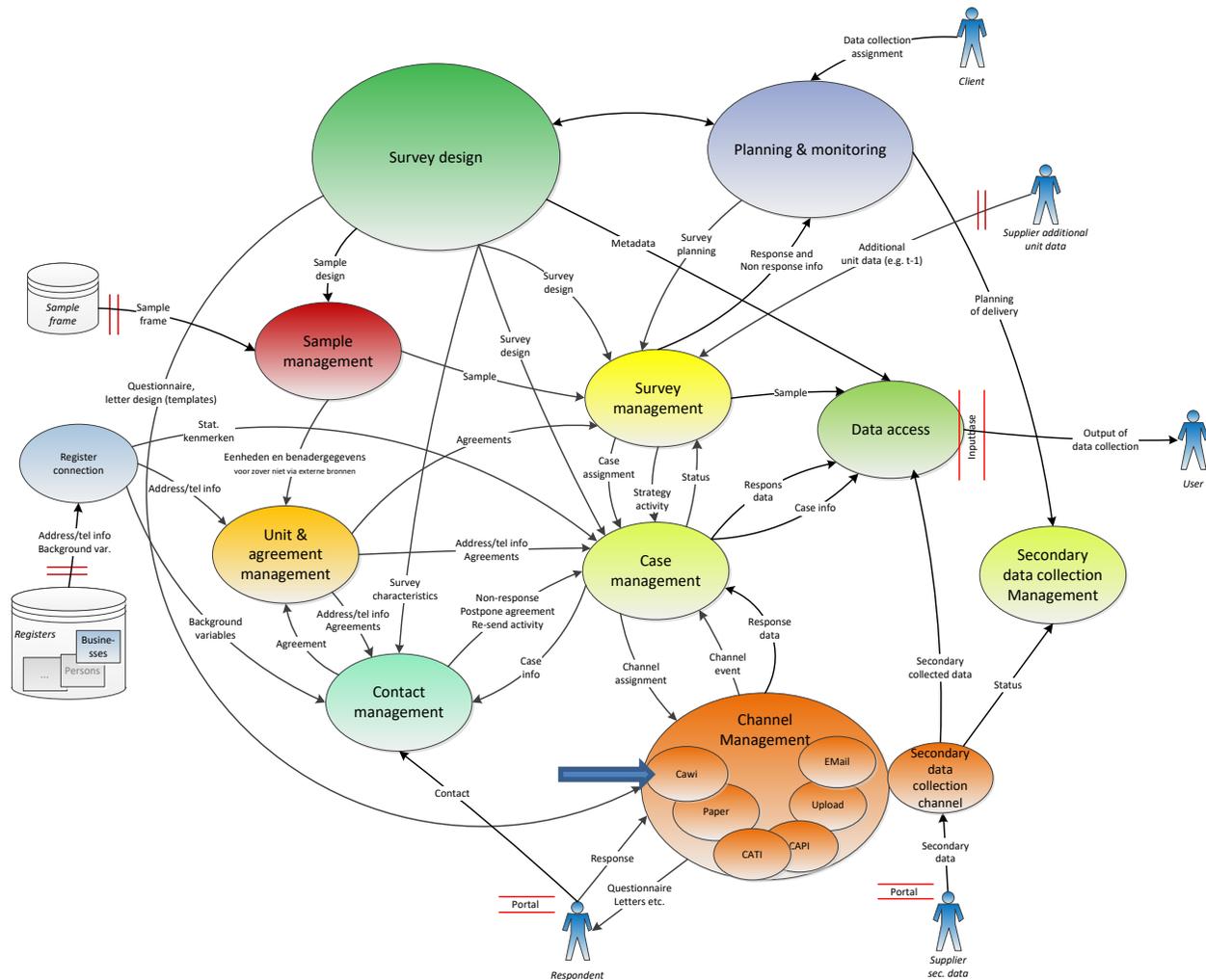
- Telephone
- Face to face
- Paper input (less and less)

And apart from this we are looking at an easier way to generate surveys (VLOP i.e. Survey template development process). Phoenix+ is scheduled to be completed during 2020.

## Blaise 5 surveys within Phoenix+

As already said, Phoenix+ is an application landscape based on different domains and consistent messaging between these domains. See figure 1 for an impression of the complexity of this application landscape.

Figure 1: Phoenix+ domain architecture



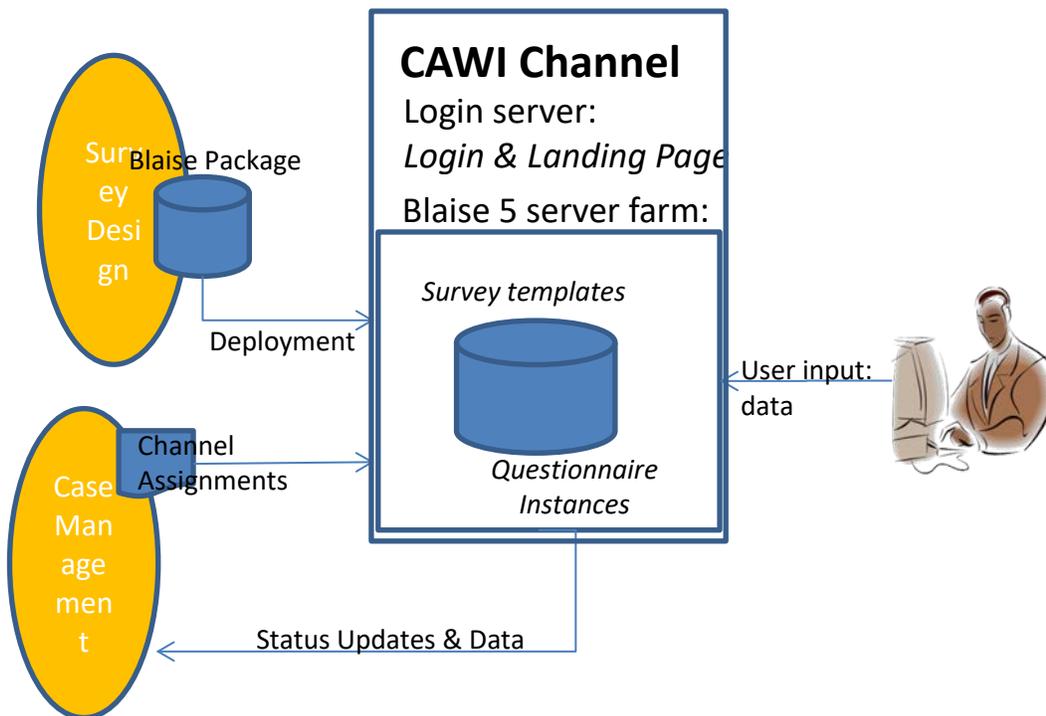
Within this picture the CAWI channel (blue arrow) seems to be only a small part, but in reality it is of extreme importance, as it is the main primary data collection channel.

In figure 2 you will find a simplified version of the interaction with and function of the CAWI channel in Phoenix+ context. The main functionalities are:

- Deployment of Blaise 5 questionnaire templates. All specific channel requirements are added to a package during deployment. This includes Master/Detail questionnaires, a switch for single or multiple reply possibility etc.

- Receive Channel assignments from other domain (Case Management). These channel assignments are transformed using the Blaise API to prefilled Blaise records
- Transforming Login credentials to a start survey command, opening the right instance of the questionnaire. The login functionality itself is generic and also used in other modes (like the upload channel or designated survey software).
- Possibility to fill in the survey safely (using standard Blaise functionality, but augmented with extra software to ensure safety)
- Re-enter survey session in a safe way after interruption/abort
- Acknowledge completed response (including possibilities to print a PDF with the response)
- Receive response and transferring this response to other domains.

figure 2: CAWI interactions with Phoenix+



In addition to these main functionalities the CAWI channel also provides:

- Heartbeat solution
- Audit trail solution
- Centralized logging

## Newly incorporated Blaise 5 features

To be as real-time as possible, we incorporated Blaise events to trigger the distribution of response from the Blaise DB to other domains in Phoenix. In particular the EndQuestionnaire event is used by a windows service to read the newly sent-in record, to convert this to the desired format and to send it as a JSON message to the next domain. This functionality is purely on the background.

Although it should not be necessary, it is a fact of life that people make mistakes which are discovered when it is too late: a misspelled sentence, a glitch in the layout. So there is a need to be able to change a questionnaire template while the survey is already in production. Blaise knows whether the data model of the questionnaire is the same by comparing the checksum of the package. We used this knowledge in our own application: if name and checksum are the same, then allow the change, if not then do not allow. See figure 3 for the screenshot of this functionality

Figure 3: Changing the template in production



In our testing environment we noticed that starting the first instance of a newly deployed package can take up to 30 seconds. We call this the 'cold start up'. One of the most time consuming actions in this cold start up is loading the Blaise Metadata file BMIX. In earlier versions of Blaise this BMIX is kept in server memory for 1 hour. If the questionnaire is not used within this hour, the BMIX is cleansed out of memory. In Blaise 5.4 the time-out of keeping the BMIX in memory is made configurable. First tests did not lead to the expected results. We are still analysing this, but as we do not get any complaints from respondents, it is not a high priority issue.

## Challenges with Blaise 5

In the earlier versions of Blaise 5 we experienced some difficulties regarding the backwards compatibility of the web layout after an upgrade. And because of the major technical changes from ASP.NET to MVC we even implemented a second CAWI channel to be able to support both Blaise 5.0.5 templates and Blaise 5.2.5 templates. But the transition from Blaise 5.2.5 to Blaise 5.3 went much more smoothly and the transition from Blaise 5.3 to Blaise 5.4 didn't even create ripples. It is safe to say that Blaise as an organisation has grown substantially in this field.

The second challenge we faced (and still face partly) concerns load testing. CBS uses virtual hardware and the results of our load tests were quite different from the results of similar tests done by the Blaise organisation on physical hardware. The point on which additional concurrent users clog the system lies substantially lower using virtual hardware. In our case we saw congestion on the webserver starting at around 80 concurrent users, whereas on physical hardware congestion didn't start until 300 concurrent users. We tried lots of different test scenarios, but in all of them we faced congestion at approximately the same number. And when the Blaise team tried their own test on our environment they faced the same. As of yet, we have not figured out why virtual hardware has such a bad influence on concurrency, but fortunately we can upscale the numbers of webserver to meet the demand (in the short run at least).

One other thing stands out in the performance area: Blaise 5.3 MVC in combination with Microsoft browsers is much slower than with other browsers. This seems to be caused by the JavaScript engine used. In one survey in particular this presented us with a big challenge, which we faced by compiling this template in Blaise 5.2.5, so the size tree functionality is not present in the build-up of the webpage. This resulted in a performance improvement of nearly 50 per cent on certain pages with a lot of elements on the page.

## **Evaluation using Blaise 5**

In general Blaise 5 ticks the following boxes:

- Receiving of questionnaire instances (channel assignments) event driven
- Distributing response event driven
- Deployment of survey templates based on interface from survey design domain
- Stability of the whole process
- Backwards compatibility

Apart from this we have seen substantial growth in the Blaise organisation over the last couple of years. The one thing still bothering us is the performance of web based surveys containing tables in the combination Blaise 5.4 and Internet Explorer browsers.



# Implementation of Blaise 5 web questionnaires into an existing Business Survey Management System

*Leif Bochis Madsen, Statistics Denmark*

## Abstract

For more than ten years Statistics Denmark has used Microsoft Infopath as tool for development of web questionnaires for our Business Survey Platform. Because this tool is going to be discontinued in a few years, we have considered alternative tools and settled for Blaise 5 as the main tool for developing web questionnaires in the coming years.

However, the entire Business Survey Platform also consists of a number of frameworks and systems in order to support automated procedures and effective work procedures in the data collection process. Implementing Blaise 5 as tool for web questionnaire development therefore implies a range of adaptations in order to support exchange of data as well as metadata between the various sub systems.

Important parts comprise exchange of data to and from our Business survey data store (XIS = Xml-based Input System), incorporation of Blaise questionnaires into our Business survey portal (VIRK) and our backend survey administration system (IBS).

The work has implied a large number of decisions with respect to details of communication between the various parts, i.e. which specific constructs should be needed to transfer data and metadata to and from the Blaise questionnaire. As a result, a basic template for questionnaires has been developed alongside a standard resource database including – beside layout standards – also constructs supporting the communication between Blaise and our backend system.

In order to make it possible to move a portfolio of approx. 65 questionnaires to Blaise in a few years, we also need to consider possible ways to auto-generate Blaise code from existing metadata in various formats.

## 1. Introduction

### 1.1 Statistics Denmark Business Survey Platform

In 2006 Statistics Denmark launched a new, centralized data collection system for all business surveys comprising data collected by paper forms, web forms, system-to-system transfers or – occasionally – telephone interviews.

The system consist of common practices and components for the entire data collection process, including a central data store and common administrative procedures like follow up on the data collection progress for each survey, providing user support, handling non-response and reminders, etc.

Furthermore, the survey portal is a part of the Danish governmental enterprise portal – the so-called VIRK portal<sup>1</sup> – including full integration with its login and authentication processes using digital signatures.

<sup>1</sup> See ref. [VIRK]

It was decided to develop the web forms using Microsoft Infopath and during the following years approximately 70 web forms has been developed. However, as Microsoft has announced the discontinuation of support for Infopath by 2022 it has become crucial to find a replacement in time.

This general system has proved a success by generalizing the procedures of administration and conduct of the wide range of business surveys into a common system. This system has become the unified entrance of surveys, questionnaires and sample data well known to all divisions working in the field of business surveys and this system was not ready for a replacement. It was therefore important to find a tool that could be integrated into this general framework.

After studying the market it turned out (in spring 2017) that there were only two realistic options available:

1. To develop our own tool for building html forms. Advantage: Easy to integrate into the existing setup. Disadvantage: A lot of work to build generators, standards, tools, and keep up with technology improvements.
2. To integrate Blaise 5. Advantage: It is a widely used system for building questionnaires for multiple platforms, it is maintained to cope with technology changes and Statistics Denmark had 25 years of experience using Blaise. Disadvantage: It seemed not easy to incorporate into our existing setup including security and authentication procedures.

After studying the two options it was decided to opt for the choice of Blaise 5 in the early spring 2018. However, because integration is easy as it resembles the integration of Infopath, it may still be possible to write hand-tailored html forms and use them alongside Blaise and Infopath forms. This option makes it possible to collect data even for surveys where Blaise 5 for some reason may not be suitable.

## **1.2 Data storage**

The Business Survey data collection system comprises a central database for storage of all collected survey data whether from web surveys, system-to-system transfers or scanned paper forms as well as internal data editing. The XIS database has been in use for the last 12 years and form the backbone for all Business survey data. I.e., all forms – whether from Infopath or Blaise questionnaires – should be stored in XIS. The entire Business Survey system operates on this database with standardized procedures for upload of sample and background data (aka. “prefill”), dissemination to data editing and analysis, sample management, etc. The questionnaire engines draw updated information from this database upon request from a respondent to fill a form. This comprises sample data and background information as well as partly filled form data from e.g. previously interrupted attempts to fill the form.

The data in XIS are stored in a relational database in a structure similar to the Blaise “In depth” data partitioning option, i.e. the data from the collected forms are all stored as field-value pairs. However, there is a logical layer – the Data Report Definition – that may describe these data in a simple, hierarchical structure.

The data report definition consists of a collection of fields, each of which may be of type integer, real, string or “group”. The latter type consists of another collection of fields and may be repeated. From this simple model data may be transferred in various formats: as objects, as xml documents or in a relational format where each defined group is represented as a table.

## 2. Implementation of Blaise 5

### 2.1 Architecture of the Business Survey Portal

The business survey setups are based on pairs of frontend web servers and backend data control servers as shown in fig. 1 below.

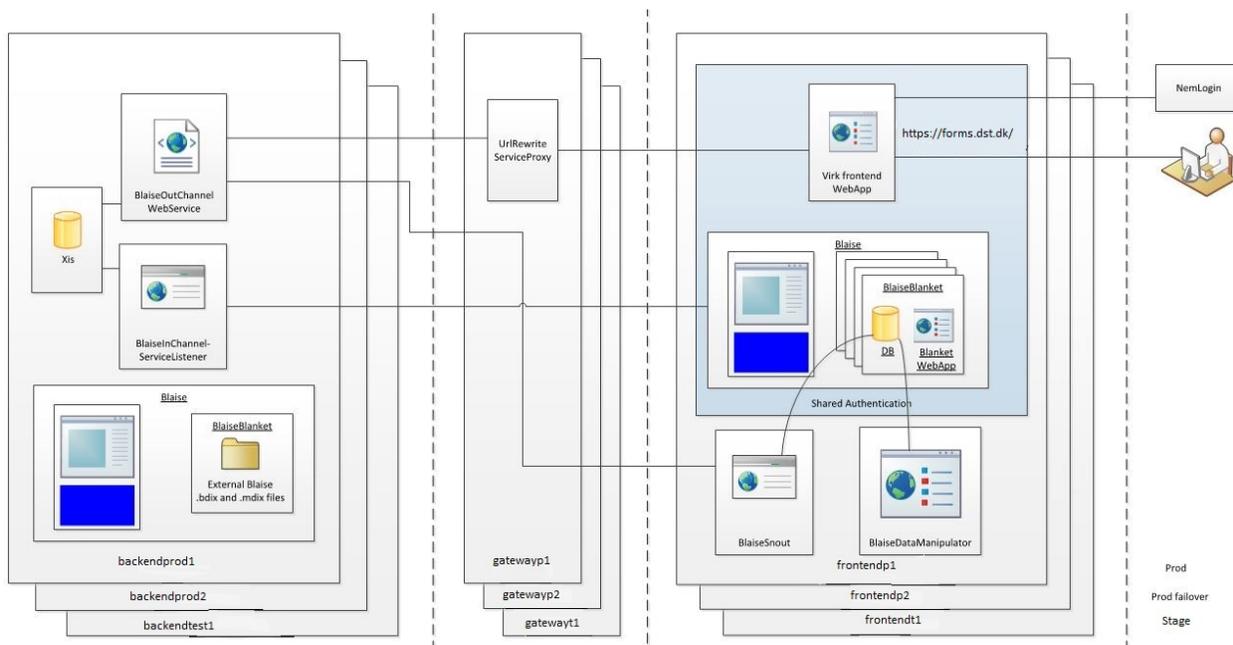


Fig. 1. Architecture of Business Survey Portal

The servers described on the figure comprise three frontend web servers, briefly named *prod*, *prod failover* and *stage*. The latter is a test server configured as an exact copy of the production servers and used for acceptance test. Each of the frontend servers are paired to an internal server used for controlling data retrieval and store.

The respondents are authenticated via the login procedure (“NemLogin”<sup>2</sup>). The authentication is carried out by an external identity provider which is part of the Danish Public Key Infrastructure. Via the authentication process the user passes an authentication token according to the SAML 2.0 standard.

All data transfers (via `BlaiseInChannel` and `BlaiseOutChannel` services) are controlled by the internal servers and operate through a relative Blaise data interface to the database on the frontend web server.

Beside these three server setups, there is also a parallel internal test setup, configured in the same way, except that the frontend server is not accessible from outside.

<sup>2</sup> See ref. [NemID]

## 2.2 Blaise as a form engine

Blaise is used in this system as a mere form engine. Originally, the requirement was that data should not be stored on the web server, but only exist as internal data in the web application. Due to the main architecture of Blaise we could not achieve that, but the database is filled with the necessary data upon start of the session and is removed again after completion, quit or timeout. The Blaise database thus works as a temporary database<sup>3</sup>.

Data are transferred between XIS and Blaise questionnaires through the standardized procedures BlaiseOutChannel (from XIS to Blaise) and BlaiseInChannel (from Blaise to XIS) as showed in fig. 2 below.

These procedures are C# programs using the Blaise API for accessing Blaise data while they share the solutions for accessing XIS with similar procedures handling MS Infopath forms.

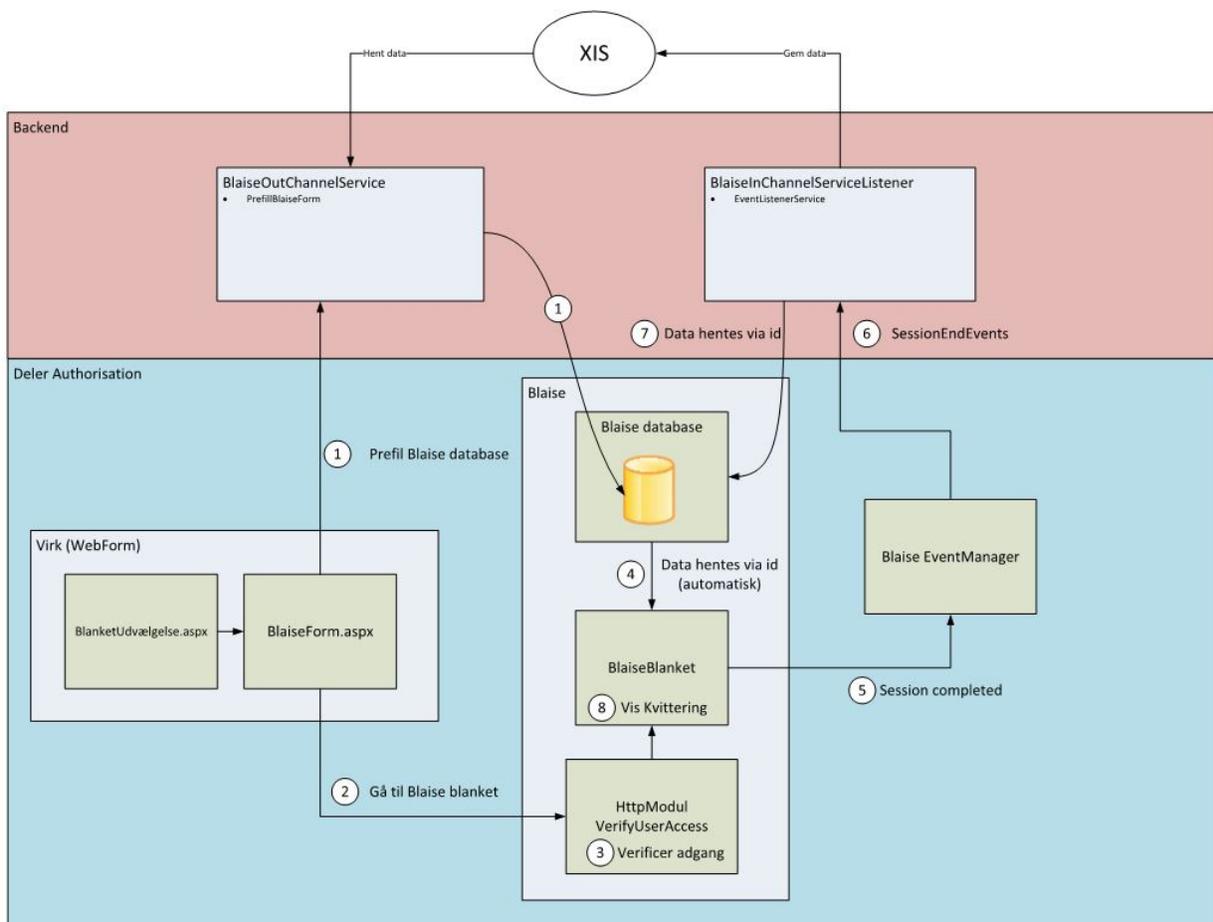


Fig. 2. Access to a Blaise 5 form

When the authenticated respondent requests a web form from the VIRK form overview (left in figure, "BlanketUdvælgelse.aspx") the web application transfers control to the BlaiseForm controller (similar procedures are available for forms developed in MS Infopath). The Blaise form controller asks the

<sup>3</sup> We are considering the option of storing Blaise data in an external database.

BlaiseOutChannel service to fetch sample data and possibly partly filled form data from the XIS data store and put it into the Blaise database identified by an ID (primary key) generated as a hash from the respondent ID and the authenticated name (steps 1). When this is done control – including verification of the provided ID – is transferred to the Blaise questionnaire (“BlaiseBlanket”, steps 2 to 4). When the form has been completed the receipt page is displayed (“Kvittering”, step 8) and meanwhile, the session is closed and within three seconds<sup>4</sup> an event is raised through the EventManager (step 5). This event is caught by the BlaiseInChannelServiceListener and the filled form data are fetched and deleted from the Blaise database and stored in the XIS data store (steps 6 and 7). Also, an email receipt is submitted to the respondent.

At last, control is returned to the form selection pane and the respondent may possibly fetch another form to fill.

A similar procedure is carried out when the respondent quits, though the status of the form will be set to “Draft” in the Business Survey Portal. Closing the browser will trigger a timeout in due time and meanwhile the form will be closed to other users, but may be reopened by the same user.

A pdf copy of the filled form may be generated from the receipt page, which is a standard feature in Blaise. Also, we have implemented a way to generate a copy of an already completed form from the form selection pane by launching the same process, though with an extra parameter to inform the Blaise questionnaire to jump to a separate page in the end of the questionnaire.

### 2.3 Necessary constraints to developed datamodels

In order to adapt Blaise data models to fit into the existing data collection scheme some limitations are implied.

1. An overall structure of the datamodel through a general template for the .blax file
2. Certain constraints to the definition of the questionnaire that allows a generic exchange of data between Blaise and XIS, i.e. storing Blaise Datafields in XIS
3. A general resource database to support the functionality required for proper communication between the Blaise Questionnaire and the Business Survey Platform

```
Datamodel example
Primary
  RespID
FIELDS
  RespID : string[100]
          // Form ID generated from respondent ID and authentication ID

  Stamdata : BStamdata
          // a block of data with info from the Business Survey Portal

  Metadata : BMetadata
          // a block of data about the respondent and the form, period etc.

  Indberetningsdefinition : BIndberetningsdefinition
          // a block of data defining the questionnaire
```

Fig. 3. Datamodel template

<sup>4</sup> The default of 10 seconds was considered too long time to wait for updates to take place.

The BStamdata block defines fields necessary for the handling of the form by the BlaiseInChannel service. The BlaiseInChannel service takes care of removing the data from the Blaise database and store the contents in XIS, and in order to do this basic information like Survey ID, Report Form ID, Report period, Entity Key (unique key per respondent per report period), version number of form, data collection period, etc. is needed.

The BMetadata block defines information needed for – mostly – personalization of the form. This includes information to publish on the receipt page, a series of links that should be used upon returning from the form to the Business Survey Portal, a help link specific for the survey, plus a couple of fields containing the state of the form – briefly: prefilled form, partly filled form, completed form. The form status information is also passed on to the Business Survey Portal when the form is closed.

The BIndberetningsdefinition block merely contains the definition of the survey questionnaire.

This template is accompanied by a template layout file, which define field and type mappings, selection of master page template and a couple of new page instructions<sup>5</sup>.

## 2.4 Questionnaire definition constraints

As described earlier, data should be stored in XIS using a simple, hierarchical structure. Compared to the constructs available in Blaise the Form Report Definition for XIS is rather primitive, so in order to support a generic exchange of data a few constraints and decisions has been made.

Also, because the coming Blaise questionnaires will be replacing MS Infopath forms with already existing counterparts in XIS Form Report Definitions it might be necessary to allow for several conversion schemes. First priority, however, is that we can avoid mapping lists for specific survey and strictly keep to a generic conversion.

Basically, we took the Blaise metadata definitions as starting point, as these are by far the mostly detailed of the two metadata systems. As we while moving data back and forth are always going to have access to both metadata definitions, we may simply check whether it is possible to store the contents of the Blaise field into the type of the Form Report field. The basic mapping should map a Blaise field into a field in the Form Report Definition based on equal names (non-case-sensitive in both languages). For the simple fields we could set up a mapping scheme as follows:

Table 1. Mapping of field types

Blaise type	Form Report Type
Enumeration	String or integer: always store the code
Set	String: comma-separated list of codes
Datetype	String: YYYY-MM-DD format
Timetype	String: HH:MM:SS format
Real, Real ranges	Float
Integer, Integer ranges	Integer
String, Open	String
Classification	String
Block	Group
Arraytype	Group, repeated

<sup>5</sup> We have not yet started to use the layout language available from Blaise 5.3.0.

Integers and reals have been implemented in the same way as in Blaise, but strings are restricted to a maximum length of 2000 characters, due to storage in an Oracle database

A Blaise BLOCK is equivalent to an XIS Group, however, an XIS Group is the only XIS type that may be repeated. Therefore, it is necessary to define Blaise data fields comprising an array of e.g. strings in a surrounding block type, e.g.:

```
BLOCK BStringInBlockToSupportArray
FIELDS
    StringToBeRepeated : string
ENDBLOCK
FIELDS
    MyStringArray : array[1..] of BStringInBlockToSupportArray
```

These constraints may of course lead to avoidance of certain constructs that are “natural” to Blaise. To overcome this limitation we have defined a naming convention, so that we can define a data field with a postfix of “NONXIS”, that will not be stored (alternatively, using auxiliary fields may also solve this problem). Then the rules of the datamodel must take care of the proper assignments of field values, e.g.:

```
Rules
    DatafieldInXIS.keep
    // move value(s) from DatafieldInXIS to Datafield_NONXIS
    // (once, at form initialization)
    Datafield_NONXIS.ask
    // move value(s) from Data_NONXIS to DatafieldInXIS
```

## 2.5 Standard resource database templates

The resource database should of course define the standard layout settings as decided by the design team. More important, however, is that the templates support the communication between the Blaise form and the general Business Survey web application.

First, we have defined a set of field references in order to pass background information about the respondent or survey to the form. For example, URL info that is used to direct the user back to the portal page, a help URL concerning specific help info, information needed for personalizing the receipt page and form status info that can be passed back.

Second, page templates (Master page, Receipt page, Abort page) are adapted to support the design chosen for our business surveys. On these pages are also defined a set of buttons needed to carry out the actions decided by the user. The user has the options of saving a draft or (if the form has been completed) to submit the form. When choosing the latter, the field containing the status info must be updated before continuing to the receipt page. This is defined as part of the events attached to the submit button.

As mentioned earlier, it is also possible to start a session only for printing a copy of an already submitted form. This feature is supported by checking a field reference: If the field has a value “Print form”, the usual navigation buttons are disabled and only the options of printing the form and returning to the portal are available.

## 2.6 Verification

A module has been built in order to check concordance between the Blaise metadata and the XIS structure. The aim of the program is to verify that the data defined in the Blaise questionnaire definition can also be stored in the XIS data store. The first version of the program is ready and we are planning further releases.

In the backlog are improvements of the user interface so the questionnaire designers may easily verify questionnaires during development and possibility to plug it into the deployment procedures so installation of non-compatible questionnaires may be avoided.

## **2.7 Deployment of questionnaires**

In order to ease deployment of questionnaires some scripts and programs have been developed comprising the necessary tasks for automating installation and adaptation to the Business Survey Portal scheme.

Automated installation of questionnaires on the web servers was the easy part as it just involved supplying the ServerManager.exe with the proper parameters like server name, Blaise server credentials and installation package.

Generation of a relative Blaise Data Interface was implemented by developing a C# program using the data interface API. The program is supplied with the installation package (.bpkg) which is actually a zip file. The meta (.bmix) and the manifest (xml format) files are extracted, the latter in order to extract the survey ID which is necessary in order to automate generation of the data interface.

Also, a powershell script was developed in order to include elements necessary to integrate the questionnaire with the Survey Portal Authentication setup into the survey web.config file. This script copies common elements and elements specific to the web servers (test and production servers) into the web.config file. Some of these settings might have been included in the installation package, but it was easier to manage these changes at survey installation and just generate a general installation package to use in any of the environments including development and pre-test (outside the portal environment).

## **2.8 Status, right now**

September 3<sup>rd</sup>, 2018 we have succeeded to deploy the first Blaise questionnaire into the new version of the Business Survey Platform in Test, Stage and Production environments, where we are now supporting Infopath- as well as Blaise 5-based web forms<sup>6</sup>.

We have started working on the implementation of further 3-4 Blaise questionnaires, which should all be ready for production by the end of 2018.

<sup>6</sup> Upon writing this paper we have not yet started to use this for real production, i.e. invited business respondents to fill the forms.

## **3. Future development**

### **3.1 Blaise Questionnaire Generation**

The existing Infopath questionnaires have been generated partly from metadata and descriptions stored in Excel files, one Excel project per survey. Of course, we want to use this information for generating the Blaise source code, too.

Therefore, we have started a project in order to define the requirements for developing a Blaise source code generator using this information. To make it work it will be necessary also to supplement the existing information with metadata from other sources.

The project aims at releasing the first version of a generator in the beginning of 2019.

### **3.2 Procedures for future deployment of newer versions of Blaise 5**

Blaise 5 is constantly improving with new features added to the system twice a year and these features we certainly want to apply for our coming surveys.

Unfortunately, it often implies certain incompatibilities between Blaise versions. Therefore, we will have to upgrade the Blaise installations as well as the programs for integration in order to use the proper versions of the APIs so we may exploit the new possibilities.

As we still we need to run existing surveys without the need to constantly upgrade and redeploy them, we also need to refine and automate our test and deployment operations in order to support new and coming features of Blaise 5.

### **3.3 Load balancing**

At last, we shall decide how to implement load balancing into the system. It would be possible to carry it out at login by the Virk Frontend Web app. An alternative might be to apply the Blaise built-in scheme for load balancing. We should make this decision soon in order to implement it in due time before the vast majority of the questionnaire portfolio has been moved to Blaise 5.

## **4. Conclusions**

Blaise 5 has reached a level of maturity that makes it an obvious choice for setting up stable, scalable data collection solutions for complex questionnaires.

The resource database definition has also been improved considerably, and we are now able to develop and apply a standard. The making of a stable, organization-wide resource database is one of the most important and time-consuming tasks in order to set up an effective, professional environment for managing business surveys.

Still, upgrading from one version of Blaise 5 to another has been a challenge and we shall need to test thoroughly before jumping into the exploitation of neat, newer features.

## 5. Acknowledgements

Thanks to Lon Hofman and Roger Linssen from the Blaise team for discussing the possible ways to make integration work. Special thanks to Eric Munsters, CBS for kindly sharing info and documentation describing the architecture of the CBS Phoenix project. Also thanks to my colleagues, who took part in the development team, without which this paper would never have been written. Especially, Maja Kirchhoff Krølner and Nicolai Zangenberg, who implemented the crucial parts and provided me with nice illustrations.

## 6. References

[VIRK] <https://indberet.virk.dk> – portal for reporting forms to municipalities and government agencies etc. A brief explanation in English can be found at <https://danishbusinessauthority.dk>

[NemID] NemLogin (“nem” = Danish for “easy”), <https://digst.dk/it-loesninger/nemlog-in/> - brief info in English: Agency for Digitisation, <https://en.digst.dk>

# Transitioning an established longitudinal study to Blaise 5 and to a mixed mode design

## Introduction to this session

The following is a session designed to share our practical experience of transitioning a study from Blaise 4.8 to Blaise 5 and to mixed mode. The session uses Health and Retirement Study as a case study to describe the processes involved in transitioning an established and complex survey to Blaise 5 and adding a self-administered mode. We will describe how we adapted the tools and systems involved in the major phases of the survey process to Blaise 5, for example, data collection, sample management, data processing and delivery. We will also focus on specific components or conversion of specific questions within the interview.

The papers which accompany this session have been collated and comprise the next section of the proceedings document.

### Part

1. Background and overview - *Rebecca Gatward*
2. Technical Design and Implementation - *James A. Rodgers*
3. Converting the survey to mixed mode data collection - *Rebecca Gatward*
4. Blaise 4.8 to Blaise 5 specification conversion - *Rhonda Ash*
5. Example one - Converting rosters from Blaise 4 to Blaise 5 - *Jason Ostergren*
6. Example two – The Word List: a Blaise conversion challenge - *Rhymney Weidner*
7. Designing and implementing a web component - *Andrew L. Hupp*
8. Adapting the sample management system to Blaise 5 - *Marsha Skoman*
9. Data processing and data delivery - *Laura D. Yoder*

# Part 1 - Background and Overview

*Rebecca Gatward, Survey Research Center, University of Michigan*

## 1. Background

The Health and Retirement Study (HRS) is a longitudinal panel study that began in 1992. The HRS surveys a representative sample of more than 20,000 people aged over 50 years in America and is supported by the National Institute on Aging (NIA) and the Social Security Administration. The study explores the changes in employment status and the health transitions that individuals undergo toward the end of their work lives and in the years that follow.

Every other year HRS participants are asked to complete an in-depth interview. During the year between interviewing, participants are also asked to complete a mail study or other supplemental survey. Until the most recent wave of data collection (which began in April 2018), interviews were completed in telephone or face to face mode, with respondents alternating between modes each wave. During the face to face interview participants complete a series of anthropometric measures (i.e. body measurements) and bio specimen collection. The telephone interviews are conducted by field interviewers and interviewers in a centralized telephone interviewing facility.

In 2018, web was introduced as a completion mode for around a quarter of participants who were due to be assigned telephone mode. Development work for the new web mode has taken place, top various levels of intensity, over the last five years – intrinsic to this was transitioning the survey to Blaise 5.

## 2. Introducing a web mode

The introduction of a web as a mode of data collection was primarily for monetary reasons. Funding remained constant, however, work scope and effort needed to maintain high response rates have increased; therefore decisions had to be made about how to reduce costs. One of the agreed methods was to offer web completion as an alternative to telephone.

## 3. Timeline

The decision to introduce a web mode was made in 2012, at the time HRS was using Blaise 4.8. Due to Blaise 5's ability to handle a multimode survey in one data model, work began to transition the study to Blaise 5 shortly afterwards.

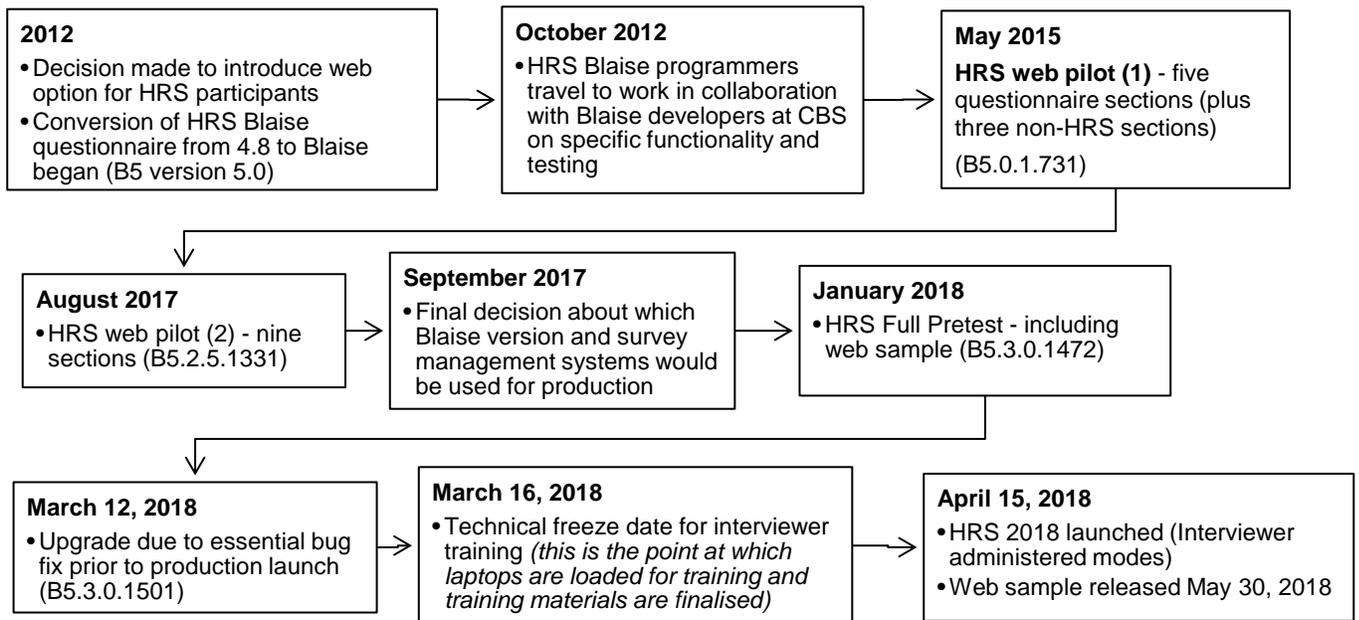
A summary of the development timeline and key decision points are included in Figure 1. The preproduction period for the 2018 wave of data collection extended much longer than is usual for a new wave of data collection and included more piloting and testing phases. The main pilots took place in May 2015 and August 2017 but there were numerous testing phases of systems and questionnaire changes.

The objectives of the initial pilot were to test the questionnaire sections that had been converted to web and Blaise 5 by this point in time, to gather feedback from respondents from a design and usability perspective and also on the acceptability of self-completion. In addition we needed to test the

functionality available in the survey management system. The pilot also provided a first chance to review data collected using Blaise 5.

The second pilot was a broader test of systems and survey rules. One of the main items was a structured test of the contact protocol we planned to use to invite participants to complete the survey and then follow up with non-responders. In addition, the pilot gave us a second opportunity to test the web interface, further functionality available in the survey management system and the conversion of HRS specific protocol to web mode. Again, the data was reviewed and analyzed to identify any potential mode effects across a variety of measures.

Figure 1. Development timeline



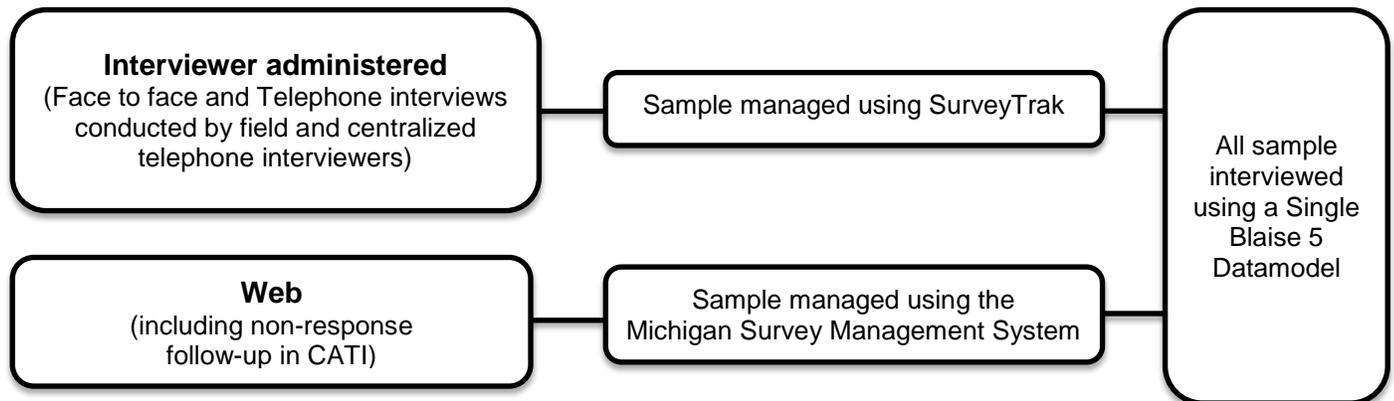
#### 4. System design

In parallel to the work required to transition HRS to Blaise 5, the Survey Research Center was developing a multimode survey management system - the Michigan Survey Management System (MSMS). At the time development was focused on building functionality required to manage online and telephone data collection.

Moving the HRS to a new version of Blaise and a new management system concurrently was high risk. To control this risk, our initial decision was to move towards this goal, we agreed, however, to set predetermined points when we reviewed progress and agreed next steps.

Our final decision point was in September 2017. At this point we decided to move forward with Blaise 5 across all data collection modes and to manage the web sample (including telephone non-response follow-up) in MSMS. The sample assigned to interview administered modes would be managed using the existing sample management system used by HRS (SurveyTrak) which was adapted to use with Blaise 5.

Figure 2. Systems used for launch of production



## 5. Piloting and testing

The preproduction period for the 2018 wave of data collection extended over a longer period of time than is usual for a new wave of data collection and included more piloting and testing phases. The main pilots took place in May 2015 and August 2017.

The objectives of the initial pilot were to test the questionnaire sections that had been converted to web and Blaise 5 by this point in time, to gather feedback from respondents from a design and usability perspective and also on the acceptability of self-completion. In addition we needed to test the functionality available in MSMS. The pilot also provided a first chance to review data collected using Blaise 5.

The second pilot was a broader test - including, a more structured test of the contact protocol we planned to use to invite participants to complete the survey and then follow up with non-responders. In addition, the pilot gave us a second opportunity to test the web interface, further functionality available in the survey management system and the conversion of HRS specific protocol to web mode. Again, the data was reviewed and analyzed to identify any potential mode effects across a variety of measures.

## 6. Scope of work involved in transitioning the HRS to Blaise 5 and mixed mode

The following is a summary of the tasks involved in transitioning the HRS to Blaise 5 and a mixed mode study.

- Convert the HRS questionnaire to Blaise 5.
- Adapt the questionnaire to mixed mode – including the development of a web instrument.
- Upgrade all current systems, that interact with Blaise, compatible with Blaise 5 - this includes systems designed to handle testing and test case management, questionnaire documentation, data processing and data quality control, interviewer quality control, systems involved in transmission of data to and from interviewers, process used to record interviews, the legacy survey management system .

- Develop screen design and layout templates for Blaise 5 – interviewer administered and web interface.

The series of papers in this session describe components of this transition process in more detail along with the technical design for systems and protocol. Throughout the process we maintained a close collaboration with the Blaise development team at Central Bureau of Statistics (CBS) to resolve technical issues – we greatly appreciate the team’s quick response to our many questions and willingness to work with us to resolve the issues we encountered.

## Part 2 – Technical Design and Implementation

*James A. Rodgers, Survey Research Center, University of Michigan*

### 1. Introduction

As described in the part 1 of this session, cost saving strategy was to move only a portion of the panel's respondents into a mixed mode protocol. This paper describes the technical design of the new sample management system that was used for that portion of the sample that was moved to a mixed-mode protocol. Most sample remained in a single-mode protocol. A later paper (part 8) will describe the modifications that were made to integrate Blaise 5 with Survey Research Center's (SRC) main single-mode system.

The addition of web as an available mode in the Health and Retirement Study (HRS) core survey for the 2018 wave brought with it several technical requirements that had not been a significant part of previous waves:

- Issue login credentials to respondents.
- Monitor web activity and manage respondent access to the survey based on that activity.
- Move cases to non-response CATI follow-up based on specific conditions.
- Remove cases from non-response follow-up immediately if the respondent completed the survey through the web.
- Deliver cases to CATI interviewers based on specific rules that take recent activity into account.

To accomplish these objectives, HRS selected the Michigan Survey Management System (MSMS) for the web/CATI non-response follow-up portion of the 2018 wave.

### 2. Major System Architecture Elements

MSMS is designed to manage mixed-mode protocols by coordinating the following functions:

- Execute specific data collection protocols.
- Interact with Blaise 4 and Blaise 5.
- Interact with email and text message service providers.
- Provide source data to SRC reporting systems.
- Manage sample contact information, such as addresses, phones, and emails.

MSMS is built around three key elements.

1. Break work into tasks and structure the system around them.
2. Use an automated task rules engine to manage manual and system tasks easily within a single project.
3. Use web- and API-based, real-time communications between internal and external applications and services.

#### 2.1 Tasks

Tasks are the unit of work within the system. For HRS 2018, the single data model represented two tasks in MSMS: ConductWeb and ConductCATI. Each survey session (open and close of the instrument) in Blaise contained a value that indicated the mode in which the session was conducted, which allowed that

session to be mapped to one of the two tasks. Through this mechanism, MSMS records all Blaise activity against the appropriate task.

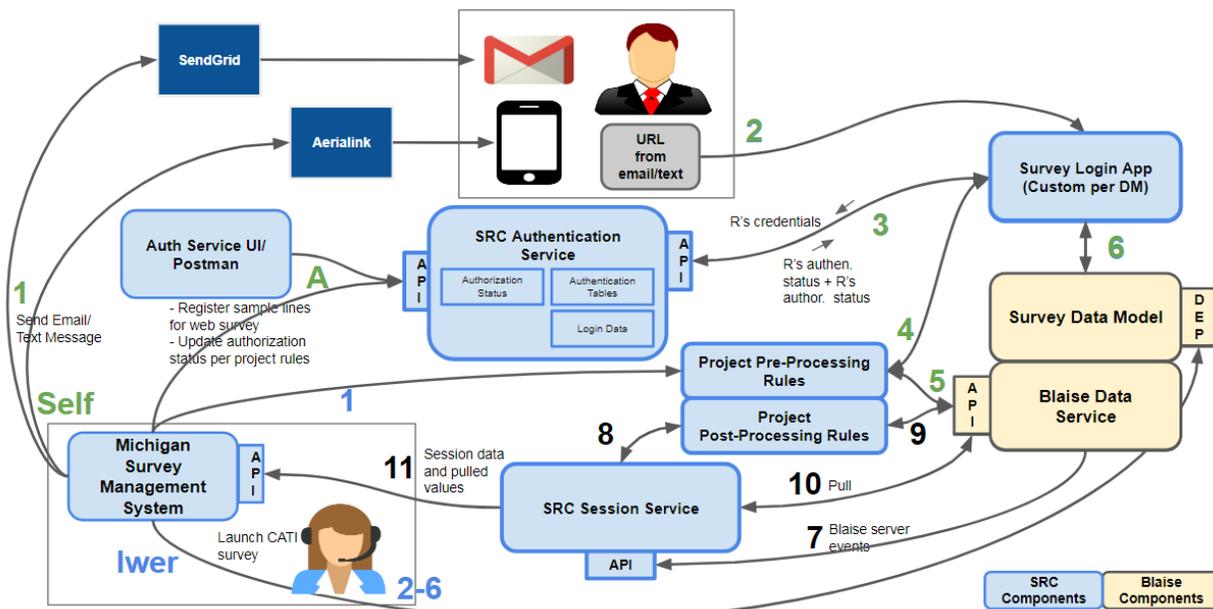
## 2.2 Automated Task Rules

MSMS includes an automated task rules engine, which assigns tasks to the appropriate personnel or system based on the project’s protocol. Automated rules are particularly good at triggering system tasks, such as sending emails or turning on access to a web survey, freeing staff to focus on higher value work. Automated rules can also signal staff when more manual work, such as sending letters, must be done. Automated rules are simply a set of if-then conditions. When they include a wide range of condition and action options, you can build powerful workflows that leverage your resources. Automated rules also allow reproducibility. During our adjustment to automated task rules that would incorporate web into an effective mixed-mode protocol, we learned several lessons.

- Automated rules require at least a moderate amount of up-front definition, which was new to us and may be new to some organizations.
- Think about worst-case scenarios. If a logic error in the rules or a manual error acts on a large batch of cases in an unintended way, how will you recover from the error?
- Start small. Iterative and incremental became our mantra.
- It is easy to build rules that are too complicated to be feasible technically or operationally.
- Handling edge cases manually is completely appropriate, but staff can’t act on what they can’t see. Create visibility with automated task assignments to signal needed reviews.

## 2.3 Real-Time Communications

SRC has built a number of web service-based services and utilities to integrate MSMS with the server-based deployment of Blaise 5 that is used for web surveys. The communication flow is almost all in real time, because it is difficult to manage flows that include automated mode changes without real-time communications between system components. The components in light blue in the diagram below are those that SRC built to complement Blaise 5’s server-based implementation and are part of the technical design that SRC used for the web/CATI portion of HRS 2018.



These components each have specific responsibilities within the integrated workflow. For example, the flow for self-administered surveys goes through several components that are not included in the interviewer-administered flow. These are specific to the management of self-administered surveys. They control the respondent's access to the survey based on real-time status of both web and CATI activity.

- Email and text services (SendGrid and Aerialink, respectively) are vendor services that SRC uses to send emails and text messages to respondents and manage the flow of bounced and returned messages.
- Authentication Service confirms the respondent's credentials and the current authorization status for that case for that instrument.
- Custom survey login app contains the public-facing login page and logic for routing the respondent to the Authentication Service, other pre-survey validation, and pre-processing rules that may apply once the case is validated.

Both self and interviewer-administered surveys use the Session Service, which is downstream from Blaise. The Session Service is built around Blaise server events that are published by Blaise to subscribing web services. Blaise publishes StartSession and EndSession events. EndSession events trigger the Session Service to pull values from the Blaise instrument through the Blaise API. The package of pulled values and Blaise server event metadata is published to MSMS, which records a contact record on behalf of the respondent for self-administered sessions. For interviewer-administered sessions, the package awaits the contact record that is recorded by the interviewer in MSMS.

When contact records are recorded, the project's automated task rules are run for that case. The task rules change the statuses of existing tasks or create new tasks, based on whether the respondent completed the survey in that survey session and on values pulled back from the survey at the end of that session. Information flows in real time to the reporting system and is available in reports or for query by the project's operations and analysis teams. The case is then ready for the next action to be taken.

## Part 3 - Converting the survey to mixed mode data collection

*Rebecca Gatward, Survey Research Center, University of Michigan*

Incorporating web as an additional data collection mode to a long running longitudinal study required much development work and posed numerous methodological and procedural issues. This part of our series of papers describes the approach used to convert the survey to mixed mode data collection - the focus is the survey instrument but does include a description of the other components of the survey process and protocols that also required conversion.

### 1. Converting the survey instrument to mixed mode

The adaption of an interviewer-administered instrument for self-completion on the web in the context of a longitudinal study is not simple. It involves decisions about optimal screen design and layout, as well as modification of wording for some survey questions, converting interviewer instructions to respondent instructions, question level help functions and consistency checks to be appropriate for self-completion. All these decisions around adaption involve balancing the goal of minimizing mode effects and maintaining longitudinal consistency against making the web survey design as attractive and easy-to-navigate as possible for respondents. There are numerous issues and challenges that required consideration throughout the conversion process, these are summarized below.

#### 1.1 Design challenges

- Comparability of questions and responses, longitudinally and across modes. The introduction of a new mode (web) must not affect the data - it must remain comparable and mode effects between web and interviewer administered should be kept to a minimum.
- Comparability of data structure. Adding the web mode would inevitably involve adding new data fields, however, a single set of variables for both modes presented in a similar structure as previous waves was the goal. In addition, the majority of any reconfiguration of the data would be done in Blaise rather than in the post processing stage.
- Clarity and ease of navigation for internet. Providing the a good user experience for web respondents required changes to the interviewer administered version - these changes were only made if they also enhanced the interviewer version and maintained comparability.
- Complexity of instrument and implications for performance and maintenance. The HRS is a large, complex instrument. In order to reduce the burden of maintaining two versions of the questionnaire, the goal was to minimize differences between the web and interviewer versions of the instrument.
- Capabilities of Blaise 5. We were transitioning the questionnaire to Blaise 5 and adapting to mixed mode while Blaise 5 itself was in development this meant that some features were not available when needed. Work arounds had to be developed or we put programming on hold until the features were/are available.

#### 1.2 Design issues

The HRS questionnaire has evolved over the years and designed for interviewer administered - additional instructions have been added for interviewer reference which makes some screens cluttered, the questionnaire includes 'volunteer only' responses have range checks at some questions for unlikely

responses. None of these are in line with design guidelines for self-administration. Some of the specific design issues included:

- Item missing data, use of don't know and refusal prompts for self-completion and routing based on non-responses.
- Appropriate ranges for numeric variables and use of hard and soft checks.
- Accommodating multiple browsers and devices.
- Interviewer help features - how and when to convert interviewer instructions, definitions, question level help to self-administered.
- Volunteered response options – these responses that are not read to respondents, but are listed in the code frame for the interviewer to select if mentioned by the respondent.
- Display of response options - avoiding the need to scroll.
- Question rewording; 'or what' terminology.
- Confidentiality: we preload responses from the respondent's previous interview there was concern about displaying preloaded information in case someone other than the respondent is completing the survey of seeing the screen.
- Multiple response formats - the interviewer administered version offers multiple ways of recording a response, for example, responses provided for when stopped smoking are 'age', 'years ago', or 'calendar year'. This would be complex for self-completion.
- 'Gates' - there are points within the questionnaire where the interviewer is not able to back up once they have moved passed a question or mark the beginning of a section and provide a point from which routing can be based.
- For most content areas, the same questions will be asked in the interviewer-administered and web modes – although this is not desirable or possible for the cognition section. We use tests in the face to face and telephone interview that are not well suited to the web. Web also offered the possibility of using visual based tests.

### **1.3 The design process**

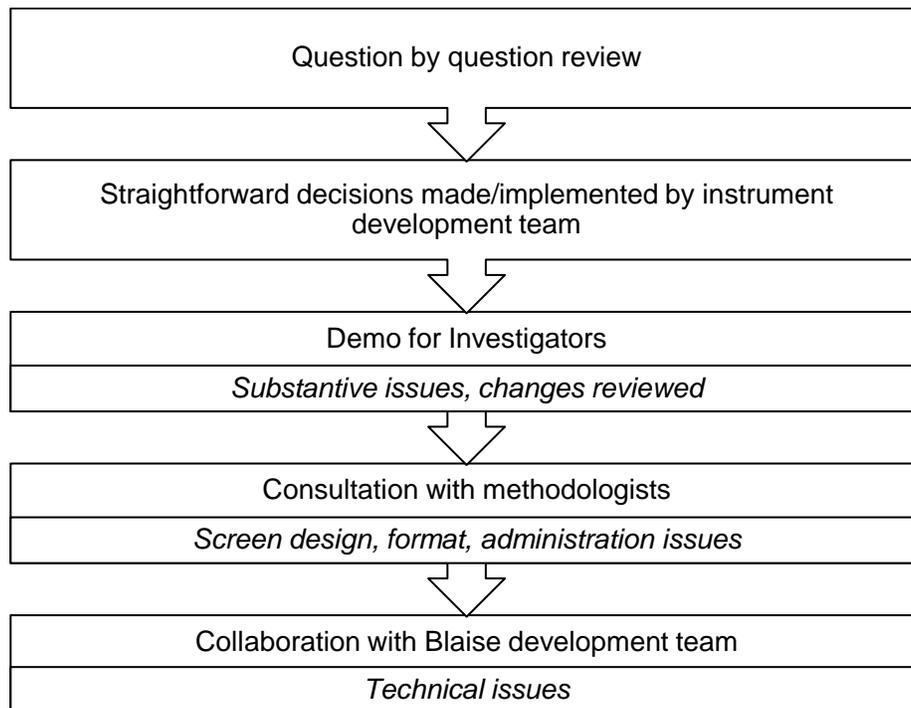
With all of these issues and challenges to consider and a large team of programmers and content experts it was necessary to develop a process to provide some structure and efficiency around the design process.

The process began with the HRS instrument development team who are involved in design, programming and testing of the Blaise instrument. This team made a first pass at reviewing each section, question by question to determine if any changes were required to question wording, response options, interviewer instructions, on-screen definitions, and/or question level help. Recommendations for substantive changes to question wording or response options were discussed with the Co-investigators who are the content experts for that section before implementation.

Each questionnaire section was demonstrated to for the Co-investigator working group. The team also worked closely with Methodologists who are experts in web survey design, question format and administration issues.

The team developed two documents which helped manage any design decisions that were relevant to the full questionnaire. The first was a record of ‘Global standards decisions’ and focused on layout, the second was a document which was produced primarily for the Design Coordinators who were creating specifications for the Blaise programmers (these documents are discussed in greater detail in part 5 of this section (Ash, R (2018))

**Figure 2. The design process**



## 2. Accommodating existing protocol and survey design

The HRS study design requires understanding household and family structure and collecting data accordingly. Protocols have been developed that relied upon system structure and tools overseen by the Interviewer. We needed to translate these protocols so that they could be handled within the web self-completion mode. Some examples of these specific protocol or features are listed below.

- **Shared preload** - Preload within the household is shared for a couple and is dynamically updated for the second person being interviewed in the household according to complex rules. Sharing preload means we have rules about when the partner of the first respondents, in a coupled household, can begin their interview. These rules mean concurrent completion is not an option and until introducing the web mode, we could rely on the Interviewer to control these access rules.

- **‘Flipping rules’** - Relationships are stored in the preload as they were collected last wave - i.e. if Bob was the first R to be interviewed then all relationships in the household are recorded as relationships to him. The flipping rules ensure that the preloaded relationships are presented appropriately for the first R who is interviewed - so if Bob's spouse, Mary, was interviewed first, all relationships would be presented for her e.g. Bob is Mary's spouse, Bill is Mary's son - rather than Mary being Bob's Spouse and Bill being Bob's son.
- **Spawning rules** - Original sample members are followed and interviewed every wave, regardless of whether they stay married/partnered to one another. If the original sample member gets remarried/repartnered (or married/partnered, if previously single), we also will conduct an interview with his or her new spouse/partner (aka non-original sample member). A sample line for the new spouse/partner will be added automatically in SurveyTrak (the sample management system). Pre-mixed mode, we asked the interviewer to ensure that they completed at least Section A of the existing respondent's interview, this would then trigger a new line will be created automatically in SurveyTrak .
- **Respondent type** - The study seeks information on both individual and couple levels. In single respondent households, the respondent is asked all sections of the questionnaire. In couple households, however, each respondent may receive a different set of questions. Questions referring to couple-level information (such as family structure (family), housing, or income (financial) need only be asked of one person. The burden of what otherwise might be a lengthy interview is consequently split between the two people in a couple household. The importance of keeping the same reporter for each wave of a panel study lies in the comparability of the data, which aids in accurately assessing the measures of observed change. The process of changing the assignment of family or financial questions to a respondent is initiated by the interviewer – a new process of checking the assignment was appropriate was necessary for the web mode.

The HRS interview is long - telephone interviewers are on average, 100 minutes and face to face enhanced (including the collection of anthropometric measures and bio specimen collection) are an average of 140 minutes. Converting the entire HRS questionnaire (24 sections) took over four years to complete. The systematic conversion process, which was imposed early in the conversion process, helped make the process as efficient as possible.

# Part 4 - Blaise 4.8 to Blaise 5 Specification Conversion

Rhonda Ash - Survey Research Center, University of Michigan

## 1. Overview

The Health and Retirement Study (HRS) is a national longitudinal survey on the health concerns and economics of aging and retirement. The HRS has, in the past, utilized a computer assisted interview (CAI) for biennial interviews of one to three hours in length given to around twenty thousand participants. Several years ago, HRS made a decision to convert to Blaise 5 so that we would be able to take advantage of the multi-mode features that are available.

There were many decisions involved in the conversion from Blaise 4.8 to Blaise 5. Some of the major decisions involved: the effects of language, mode and role changes; dropdown list vs. look up tables; signals vs. checks; how to handle grouping; and the handling of interviewer instructions. This was a complicated process that required hours of discussion before the first beta application. Following the beta-application, there were hours of testing, changing, re-testing, more discussion, changing, and re-testing again. We started with one of the “easiest” sections and spent 3 months until completion. Through that exercise we worked through most of the standards that would be used in later sections. The purpose of this paper is to describe some of the major changes that were needed to convert the HRS from Blaise 4.8 to Blaise 5, and outline some of the challenges and obstacles that were encountered along the way.

## 2. Decisions to be made

Very quickly we found that there were many opinions about how questions or fields should be presented and what effect multi-mode presentation would have on each screen. With a longitudinal survey, several decisions had to involve Co-PI input to ensure the content of the subject matter was not altered. In order to organize the vast number changes that needed to be considered, a chart was made to track decisions as they were made. There were oftentimes several decisions made on a single issue. All changes were tracked in an Excel spreadsheet that ended up being several tabs and many pages long. This spreadsheet proved to be a useful tool to track decisions and review from time to time (Figure 1).

Figure 1. 2018 Global Standards Decisions Spreadsheet

Item	Keywords	Affected Sections	Affected Fields	Question, Issue or Standard	Reviewed by Mick, Co-PI or SRO?	Thumbnail Example	Final Decisions (most recent at top)	Links to Notes or Screenshots	Dev Items
Signal 1 Indent	All			Should signal text be indented to align with Q text or with response options?	no		11/11/2016 Signal text should be indented to align with response options; KT, JR and RW via email, see link.	<a href="#">2018GlobalDecisionsScreenshots&amp;Notes.docx</a>	
				Should the size of the "other-specify" text field be limited to 300 rx, as it			11/11/2016 Keep "other specify" field size limited to 300 rx. KT, JR and RW via email.	<a href="#">2018GlobalDecisionsScreenshots&amp;Notes.docx</a>	

### 3. Making Changes to the Specifications

#### 3.1 A Migration Tool

We chose to write a migration tool (BlaiseSource) to help in the conversion process. This proved valuable as we were able to make changes quickly when the “standards” direction changed, and it did so often in the 3 year process. Much time was saved by allowing this program to make alterations like translation from ‘@’ to <tag>, changing language tag (CorEng, PrxSpn vs. Eng, Spn), and adding banners.

As we went through the design and decision process we were able to add and alter different parts of the code using the BlaiseSource program. The procedure list grew rapidly (GENERATEIWERSELFTAGS, ADDROLES, ADDLANGUAGES, MOVELABTOROLE, FINDZEROS) and we found that we needed to be able to import code from other sources and alter specs for cleanup purposes (IMPORTNEWDESCRIPTORS, LISTDEFINEDTYPES, REMOVECOMMENTS, REPLACELANGUAGECONDITIONS).

Because of the multi-mode application, we needed to insert a new Web language into the specifications (GENERATEIWERSELFTAGS). Alterations to signals and help text all had to be re-tooled for a self-interview as well (REMOVEFIHELPNOTICE).

#### 3.2 The Blaise Compiler

Changes had to be made to the Blaise compiler so it would correctly interpret the specifications. For example, an ‘empty’ or ‘zero’ value is evaluated differently in Blaise 4.8 and in Blaise 5. Reviewing compiler warnings after the compilation was needed (Warnings about ‘empty’ and Warnings about ‘zeroes’) (FINDZEROS ). This needed to be done on a field by field bases to ensure what the correct intent was.

With the Blaise version changing so often we had to pay attention to the warnings that were presented when compiling. See Figure 2 for examples of commonly seen warnings.

Figure 2. Common Compile Warnings

```
12:48:49 : WARNING: Generated Page 1 (Layoutset Interviewing1) contains 2 Layout errors
12:48:49 : WARNING: Generated Page 1 (Layoutset Interviewing1, Parallel PRIMARY) contains 1 Layout error
12:49:41 : WARNING: Self Reference: Field Relations of Block Type BA.BA_Relations is put on route with
generated parameter 'Relations.A023_PWSpALive' that contains an instance of the same type
12:49:21 : WARNING: ==> Warning Line 3941 (51) - The result of the expression is always 'true'. Use = RESPONSE
instead. : C:\SVN\production\2018HRS\source\HRS18_T.incx
12:49:20 : WARNING: ==> Warning Line 692 (32) - The meaning of this expression differs from Blaise 4. It is only
'false' when the left hand side has been assigned the value '0'. : C:\SVN\production\2018HRS\source\HRS18_S.incx
```

#### 3.3 Blaise Version Upgrades and Changes

We found that with this large of a survey and the changes in the Blaise program (sometimes weekly) we had to test most features with each release in order to ensure they were not affected by the version upgrade. In order to efficiently test the areas of greatest concern, we made a mock survey (waterpark) that contained most of the areas we were concerned with (dropdowns, screen formatting, and selected grouping). This process sped up the testing and gave us a way to notify Stats if a problem was found.

### 3.4 Specification Upgraded for Language

Back in 2002 when HRS moved to Blaise 4.3, HRS designed the specifications to use language to control mode and role attributes. This allowed HRS to maintain only one set of specifications for the whole survey. HRS had not only a Core, Proxy and Exit questionnaire in English, but each counterpart in Spanish as well. The main advantage to this being that only one location need be updated when there was a change. See Figure 3 for the language breakdown in the 2016 wave.

Figure 3. Language and Roles Breakdown in the 2016 Wave

```
2016
LANGUAGES =
    CORENG "SELF - English", CORSPN "SELF - Spanish",
    PRXENG "PROXY - English", PRXSPN "PROXY - Spanish",
    SPPENG "SP PROXY - English", SPPSPN "SP PROXY - Spanish",
    EXTENG "EXIT - English",  EXTSPN "EXIT - Spanish"
```

With the new features of Blaise 5 we were able to break down mode, roles and language. Mode is whether the interview is self-administered or interviewer-administered, language is whether the interview is conducted in English or in Spanish, and roles are whether the interview is a core, proxy, spouse proxy or exit interview (Figure 4). We were still able to give one location to be updated when there was a change needed, but were now taking advantage of the new multi-mode fracture in Blaise 5.

Figure 4. Mode, Roles, and Language Breakdown in the 2018 Wave

```
2018
MODES =
    SELFADMIN DESCRIPTION ENG "TAKEN BY RESPONDENT",
    IWERADMIN DESCRIPTION ENG "ADMINISTERED BY INTERVIEWER",
LANGUAGES =
    ENG "ENGLISH",
    SPN "SPANISH"
ROLES =
    COR, PRX, SPP, EXT,...
```

This meant that most of the specifications had to undergo a major overhauling. Every field where there was a difference between the presentation of ‘interviewer’ (Figure 5) and ‘self’ (Figure 6) mode needed to have that defined in the specifications.

Figure 5. Interview-Administered Mode Specifications and Screen Display

Interviewer mode

{CR} ENG "How much time do you spend traveling or commuting to and from your work on a typical day?  
 <newline><newline><INST><img source=diamond>Enter 996 if R works all or mostly from home.  
 <newline><img source=diamond>Enter 997 if R reports that the length of commute varies.</INST>  
 <newline><newline><INST>Minutes:</INST>  
 <newline>OR  
 <newline>Hours:"

Figure 6. Self-Administered Mode Specifications and Screen Display

Self-Administered Mode

SELFADMIN

ENG "HOW MUCH TIME DO YOU SPEND TRAVELING OR COMMUTING TO AND FROM YOUR WORK ON A TYPICAL DAY?"  
 PRELAB ENG "MINUTES"  
 PRELAB SPN "MINUTOS"  
 TEMPLATES "SPECIALANSWER"

## 4. Switching the Specifications to SELFADMIN Mode

When considering how we could convert our interviewer-administered survey to a web friendly self-administered survey, our goal was to create a layout that was friendly to someone who had never taken a survey online, while also maintaining the interviewer-assisted application. By leaving the default as IWERADMIN, we were able to add to the current specs and change the text to adjust for SELFADMIN. This feature, along with major manipulation in the BLRD file, let us show one set of text for a Self-administered and another for the Interviewer-administered (Figure 7).

Figure 7. Interviewer Administered vs. Self-Administered Specification

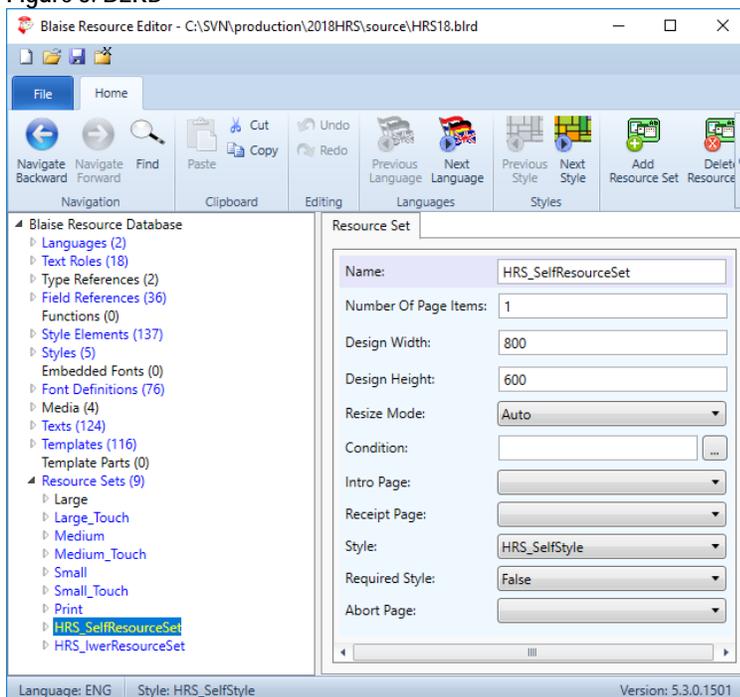
```
B126_ (B126)
ENG "We'd like to know more about how old you were when you had the health condition(s) we just talked about.
<newline count=2><INST><img source=Diamond>Select continue.</INST>"
{-08}

/"Childhood Health Follow-ups Intro"
SELFADMIN
ENG "We'd like to know more about how old you were when you had the health condition(s) we just talked about.
<newline count=2><INST><img source=Diamond>Please select \"Next!\" to continue.</INST>"
```

### 4.1 Managing Two Different Worlds in One Place - BLRD and Templates

Template control became very important. How would we work on both modes of the application at one time? – We split the BLRD into several slices by using layout sets (divide and conquer). This allowed both modes development and testing to go on together. We had to quickly develop a naming convention so the layout sets could be worked on by two different programmers and merged back together in one BLRD (Figure 8).

Figure 8. BLRD



Using two layout sets to control mode caused the file size to more than double, which prompted concerns. The compile time extended to 1.5 hours. After talking to Stats about this, we started using the “Compile – pages” and “Compile – No pages” feature. The “Compile – pages”, (or FAST as we nicknamed it) took 1.5 hours to compile, but the survey responded in real time without generating the page as the survey was presented. Since there was such a time lag with the compile delay we also used the “Compile – No pages” (or SLOW as we nicknamed it). That allowed faster turnaround time for smaller tests. We would make many SLOW compiles throughout the day and a FAST once a day.

#### 4.2 BLRD - Using Automated Templates and Layout Sets

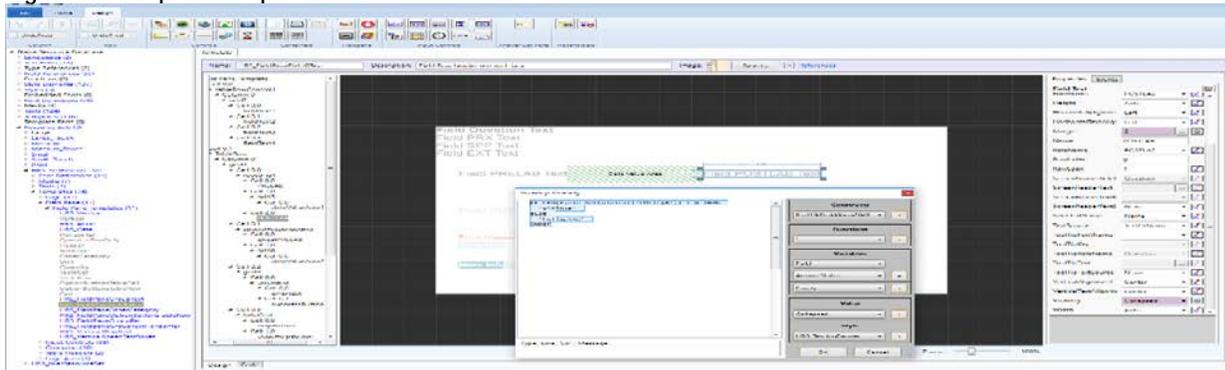
There was a large learning curve to understanding controlling the BLRD features. We needed to know it well enough to automatically assign templates and mode based on field type or grouping. Without using this feature we might have never gotten our survey out to the field in Blaise 5. By using templates named to reflect the CAWI and CAPI mode, a naming convention was developed to allow automated usage of these template rules (FIELDQTEXTS, \_GROUPTEXT, \_GROUPTEXTWITHOR, \_OTHERSPECIFY). After a template was developed and tested we could call it in the specifications and would know how it was going to present the screen. In the example in Figure 9, the template call in the code tells Blaise to use the attributes of the MaskLarge template for N014\_, and nothing more is needed. Similarly, if we wanted to present a pre or post label on the screen at N014\_ we would only need to specify that in the specs and the template would know to look to a role text of PRELAB and POSTLAB (Figure 9).

Figure 9. Template Usage in the Specifications

```

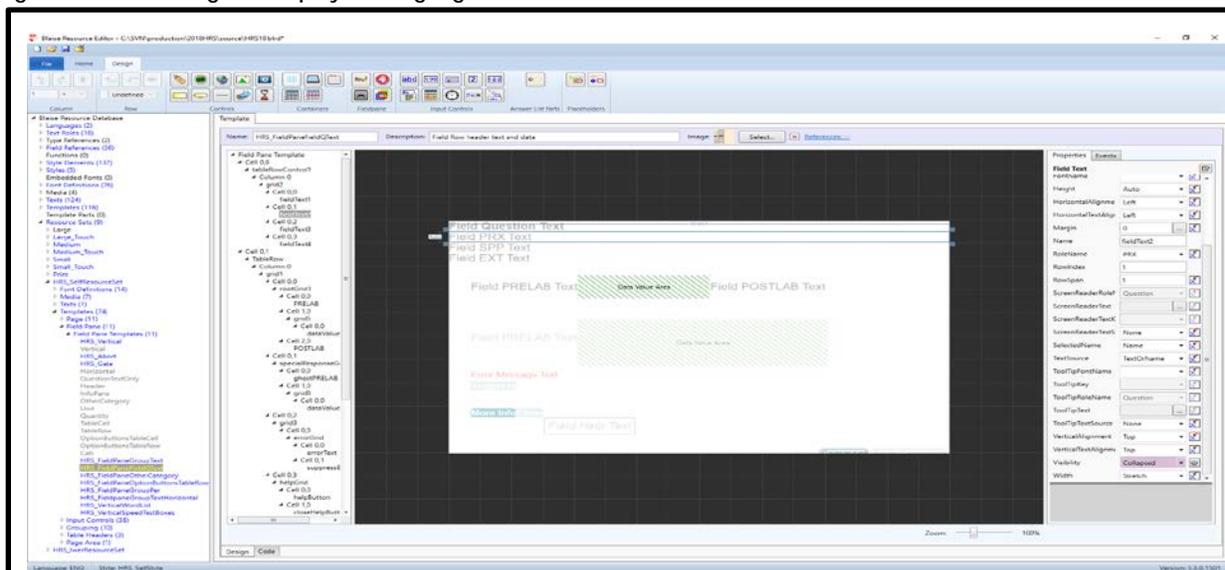
N014_ ("N014_")
  PRELAB ENG "$"
  PRELAB SPN "$"
  POSTLAB ENG ".00"
  POSTLAB SPN ".00"
  Templates "MASKLARGE"
  
```

Figure 10. Template setup



Another example of the BLRD control is using Expressions through the Visibility Property to control the cells of a template, allowing the display of one language role versus another (Figure 11).

Figure 11. Controlling the Display of Language Roles



### Expressions code

```

IF (IWType.EnumerationValue = 1 OR IWType.AnswerStatus = Empty) OR
((LEN(Field.GetRoleText('SPP')) = 0 AND LEN(Field.GetRoleText('PRX')) = 0) AND
LEN(Field.GetRoleText('EXT')) = 0) THEN
'Visible'
ELSE
'Collapsed'
ENDIF
    
```

### 4.3 Grouping

A new feature to Blaise 5 is the concept of groups. Blaise help defines the purpose of the group section as: “Ties together related fields in order to allow a horizontal or tabular display and/or a special behavior on the screen.” Of course, HRS has to add more complexity by having groups within groups (Figure 13).

In order to make templates for several different types of groups, the standards had to change and we had to think with a multi-mode brain. The BLRD was used for a lot of the control. These templates controlled the look and feel of several features, including the other specify screens, drop downs, tables, scales, and rosters. Again, we had to split these templates into the two modes to allow different presentations.

Figure 12. Blaise Resource Database



Figure 13. Template Examples



Figure 13. Example of Group in Group code (outer group- blue, inner group green)

```

Group Group_F176_GROUPTEXTPER
{CR}ENG "How often do you get together with people in or near the facility just to chat or for a social visit?
<newline><newline><INST><img source=Diamond> If 'almost never' or 'never,' enter \"0\" at number of times, and select
\"Next\" to continue.</INST>"
    Templates "GROUPTEXTPER"

GROUP GROUP_F177_OTHERSPECIFYPER
    Templates "OtherSpecifyper"
FIELDS
F177_FreqGetToget ("F177_")
    PRELAB ENG "Per"
    PRELAB SPN "Por"
    AlienArgs "SignalOverride"
    Templates "OtherSpecifyper"
    /"NUM TIMES GET TOGETHER WITH PEOPLE- PER"
    : TTimeUnit5Alt
F178SFreqGetToget_S (F178S)
    ENG "Other (specify):"
    SPN "Otro (especifica):"
    /"NUM GET TOGETHER W/PEOPLE- PER- SPECIFY"
    : OPEN
rules
    F177_FreqGetToget
    IF F177_FREQGETTOGET = OTHERSPECIFY THEN
        F178SFreqGetToget_S
    ENDIF
EndGroup

Fields
F176_NumGetToget ("F176_")
    AlienArgs "SignalOverride"
    PRELAB ENG "Number of times"
    PRELAB SPN "Número de veces"
    /"NUM TIMES GET TOGETHER WITH PEOPLE"
SELFADMIN
    Templates "GROUPTEXTPER"
    :0..995

Rules
F176_NumGetToget
    NumberMask (NO,F176_NumGetToget, Amount_Masked)
IF (F176_NumGetToget = RESPONSE AND F176_NumGetToget <> 0 AND IWERADMIN) OR SELFADMIN THEN
    GROUP_F177_OTHERSPECIFYPER
ENDIF
IF F176_NumGetToget = 0 THEN
    F177_FreqGetToget := EMPTY
ENDIF
Endgroup

```

## 4.4 Special Answers

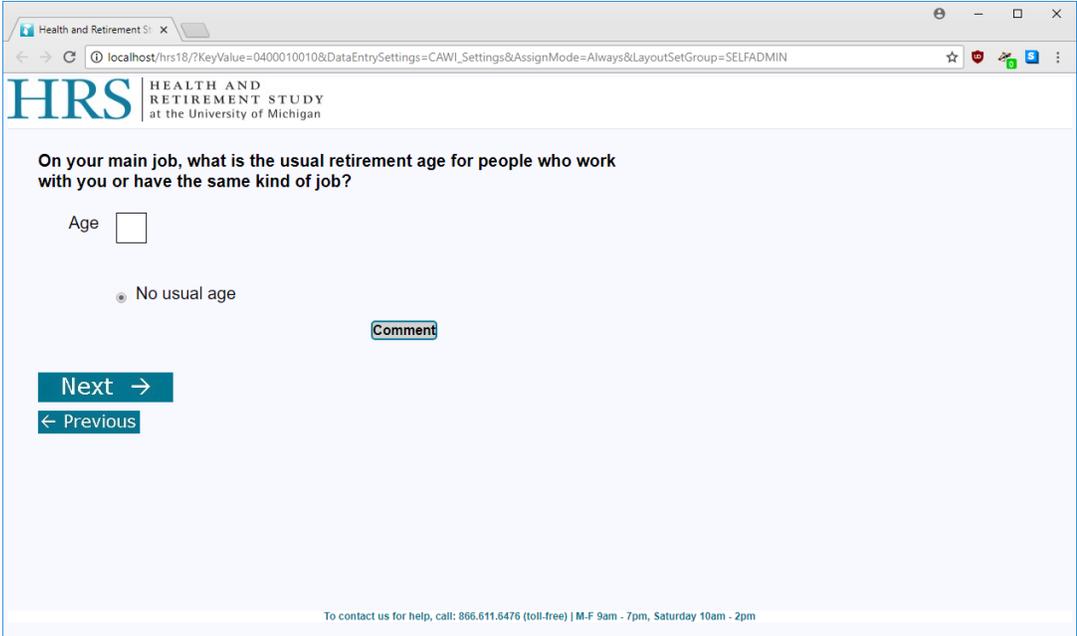
We used the special answer feature when there was a common answer over and above a range or codeframe. This allowed us to define data values that had special meaning other than the allowed range. We have an INCX file that holds all the “Special Answers” for HRS. (HRS18SpecialAnswers.incx). We applied the same naming convention to the answer types.

(SAV7A\_V732 = “Special Answers”, section V7A, Field V732). These “Special Answers” show on screen as code frames to be clicked if needed (Figure 14).

Figure 14. Special Answers Specifications and Screen Display

```
SPECIALANSWERS =
  SAJ564_95 ENG "No usual age" SPN "No hay una edad típica" {SP-18},
  NoAmount ENG "No amount" SPN "ninguna cantidad", {used in J529, J521}
  SAV7A_V732_996 ENG "Works all or mostly at home " SPN "Trabaja todo o la mayoría del tiempo en casa" {SP-18}
  SELFADMIN ENG "I work all or mostly at home",
  SAV7A_V732_997 ENG "It varies a lot" SPN "Varía mucho" {SP-18}
  JSELFADMIN ENG "My length of commute varies" ,

SPECIALANSWERSETS
  DefaultAttrib =
    SELFADMIN: DONTKNOW, REFUSAL, EMPTY
    IWERADMIN: DONTKNOW, REFUSAL
  CAWINODKNORF =
    SELFADMIN: NODK, NORF, EMPTY
    IWERADMIN: DONTKNOW, REFUSAL
  CAWI_NO_RF =
    SELFADMIN: NORF, EMPTY
    IWERADMIN: DONTKNOW, REFUSAL
  SAJ564 = SELFADMIN: SAJ564_95, DONTKNOW, REFUSAL, EMPTY   SAV7A_V732 =
  SELFADMIN: SAV7A_V732_996,SAV7A_V732_997, DONTKNOW, REFUSAL, EMPTY
```



Health and Retirement Study  
HRS HEALTH AND RETIREMENT STUDY at the University of Michigan

On your main job, what is the usual retirement age for people who work with you or have the same kind of job?

Age

No usual age

[Comment](#)

[Next →](#)

[← Previous](#)

To contact us for help, call: 866.611.6476 (toll-free) | M-F 9am - 7pm, Saturday 10am - 2pm

For a self-interview mode, we added the special answers for DK (Don't know) and RF (Refused) to several fields to allow the self-interview the option (Figure 15).

Figure 15. Don't Know and Refused Options for a Self-Administered Survey

Health and Retirement Study

localhost/hrs18/?KeyValue=0400030010&DataEntrySettings=CAWL\_Settings&AssignMode=Always&LayoutSetGroup=SELFADMIN

**HRS** HEALTH AND RETIREMENT STUDY at the University of Michigan

Suppose that there were **part-time** jobs available to you that are suited to your skills and that you could take any time you wanted. In this case, what are the chances that you would be working for pay at some time in the future?

Number

Don't know

Rather not answer

[Comment](#)

[Next →](#)

[← Previous](#)

To contact us for help, call: 866.611.6476 (toll-free) | M-F 9am - 7pm, Saturday 10am - 2pm

Like the DK and RF these special values are assigned into the data and data processing will convert them to values for public viewing.

#### 4.5 Checks and Signals

A decision that was made early on was that the Self-Administered mode had to allow the respondent to leave fields empty whereas the Interviewer-Administered mode did not. This meant that we had to adjust the signal and check logic so they were not presented with a warning if a field was left empty in self mode. Additionally, in order to reduce respondent burden, most checks became signals (Figure 16).

Figure 16. Specifications for a Self-Administered Signal

```
Signal F008_YRMADIED <= piINITA501_CurDateYear  
or Selfadmin  
"Year entered is greater than today, if correct press suppress"
```

We also wanted to be able to change the text of the Signal/Check between modes (Figure 17).

Figure 17. Specifications for Differing Text Between Self-Administered and Interview Administered Signal

```
SIGNAL F002_MaAge IN [60..120] OR F002_MaAge = NONRESPONSE INVOLVING (F002_MaAge)
  "Values between 18 and 59 or between 96 and 120 are unlikely - please check"
SELFADMIN
  "Just to confirm, you entered ^{F002_MaAge}. If this is correct, select \"Next\" to continue.
  Otherwise please make the correction before continuing."
```

#### 4.6 Instructions

In addition to changing signal text, instructions throughout the instrument had to be removed or softened between interviewer and self modes. See Figure 18 for an example of such an instruction.

Figure 18. Specifications for Differing Text Between Self-Administered and Interview Administered Instructions

```
F196_F036_ParST
("F036.5_")
  {CR}ENG "In what state do they live?
  <newline><newline><INST><img source=Diamond> Type 'OT' to enter 'other country'.</INST>"
Selfadmin
  {CR}ENG "In what state do they live?
  <newline><newline><INST><img source=Diamond>If they do not live in U.S., select 'Other country'
  from the bottom of the list.</INST>"
```

#### 4.7 Codeframes

Much like instructions and signal text, modifications were also made to codeframes so that self interviews referenced the first person and interviewer mode referenced the third person (Figure 19).

Figure 19. Specifications for Differing Text Between Self-Administered and Interview Administered Codeframes

```
MOTHMOVEDINWITHR (1)
  {CR}ENG "Mother moved in with R"
SELFADMIN {CR}ENG "My mother moved in with me" ,
RMOVEDINWITHMOTH (2) {CR}ENG "R moved in with mother
SELFADMIN {CR}ENG "I moved in with my mother" ,
```

#### 4.8 Help Text

Because a self-interview does not have an interviewer present to answer questions, we added more help text for the self mode. Help text for self-interviews was specialized to remove any reference to the “R”, was softened and addressed directly to the respondent (Figure 20).

Figure 20. Specifications for Differing Text Between Self-Administered and Interview Administered Help Text

HELP ENG "There can be some complicated ownership arrangements here. Simple ones are owns or rents, but the farm could legally be a separate business that the R (or someone else) might own. In this section, we are concerned with the house and the immediately surrounding land; if it's owned by a business, it isn't a home owned by the R."

SELFADMIN  
HELP ENG "In this section, we are concerned with the house and the immediately surrounding land; if it's owned by a business, please don't count the home as owned by you, personally."

## 5. Conclusion

Decisions had to be made, designs changed, and tools had to be altered to allow the multi-mode longitudinal survey to transform to the Blaise 5 world. HRS still maintains one set of specifications that include not only multiple languages (ENG, SPN) and roles (Core, Proxy, Exit), but now includes Mode (IWER, SELF) as well. Programming the HRS survey is much like building a labyrinth with its many layers that all have to work together. This transition was a challenge that took several years to accomplish. With a lot of help from the Blaise team, along with the Survey Research Operations and HRS staff, we are now out in the field collecting surveys both with Interviewer assisted and self modes.

# Part 5 - Converting HRS from Blaise 4 to Blaise 5 – the Example of Rosters

*Jason Ostergren, Survey Research Center, University of Michigan*

## 1. Background

Historically, The Health and Retirement Study (HRS) has had a handful of cases where it presents arrays of people or pensions in table form in Blaise 4 versions of the instrument. These tables can be filled with preloaded answers from prior waves for confirmation, and also must allow for the addition of new rows to accommodate people or pensions that are new or that previously were missed. These include children and household members of the respondent, which are asked about in separate tables near the beginning of the survey. The information gathered there is used in many subsequent sections - for example, to drive loops and fills about money transfers or as options in enumerated type questions about help provided. HRS later presents a similar table about siblings of the respondent, used in similar ways. Finally, HRS has a table for pensions and associated employer information. That is crucial for setting up the lengthy sequences devoted to understanding respondents' pensions and gathering related employer info to verify information in some cases. Pension questions often seem very difficult for respondents because the subject matter is often not well understood. The survey design must do as much as possible to avoid adding additional complications due to user-friendliness issues. People rosters seem comparatively easier because respondents face fewer conceptual problems in answering these questions, but we have found that it's not uncommon for things to go off track even there, especially when the survey is not easy to use. The main HRS survey has always been interviewer administered, so while user friendliness issues may cause confusion and delays, the interviewer has always existed as a backstop who can help get things back on track most of the time. HRS 2018 includes the first self-administered interviews in our main survey, however, and that has required a stepped up effort to make sure that these questions are presented effectively.

## 2. Rosters – more detail

In each case, these sequences are built around underlying arrays of blocks, which may be preloaded before the survey starts by external tools, and which are passed on to later sections of the survey for gathering additional data in follow-ups or for organizing later sequences. In general, the array works as an input to these sequences, and there is usually a matching array that constitutes the updated output for later use (we don't simply write back to the original, both because there is sometimes need to compare and also because we have found that writing back in Blaise usually produces complications). This array data is meant to be concise and to transmit only the required information between waves and between parts of the survey. It is not organized in a way that is optimal for presentation as questions. For that, we populate an array of blocks that contains this basic data as well as fields which allow it to be confirmed or collected in more natural questions. For example, we may have a single variable containing compound coupled status in the preload array, but we add additional variables in the asked array to allow for a sequence of questions asking first about continuation of any known prior relationship, then marital status, then non-marital partners. Data is initially unpacked from the compound variable in order to prefill answers in the split ones and then it is re-packed into a corresponding compound variable in a new copy of the preload array for use in later sections. The preload array also contains variables which are essentially administrative (such as tracking numbers) that do not have corresponding asked questions, but must be passed along to the output data.

## 3. Rosters – the solution

We have chosen to present these arrayed questions for asking grouped on screen as a table or grid rather than simply a looped series of questions. The reasoning behind this is twofold. First, we think that this presentation of the looped data allows for better situational awareness about the whole set being asked

about. Though small in number, we have historically had a great deal of trouble reconciling duplicates and other anomalies in these arrays. We believe that the likelihood of problems would turn out to be much greater if the screen didn't show, at a glance, which people or pensions were already accounted for. Secondly, we want to make it easy to return to an earlier item in the roster to correct mistakes. An alternative which required walking back through every question across multiple items has always seemed like an invitation to mistakes. In Blaise 4, the Table construct made it fairly easy to use arrow keys to move to previous items without touching each intervening question and, crucially, to be able to see how to get there. Due to the interviewer-administered nature of our Blaise 4 instruments, it was also possible to cram all of the asked fields into the table on one screen by careful management of column width. In Blaise 5, we were looking to retain an overview and navigability, while actually only displaying one question (as is our standard practice in Blaise 5) or one rows' worth of questions per screen.

The screenshot shows the HRS Questionnaire interface. At the top, there is a navigation bar with options: Forms, Answer, Language, Testing, Help, View Consents, Show Watch Window. Below this is a yellow instruction box with the text: "• Ask or record" and "Does JOHN live with you?". Below the instruction box is a list of radio button options for relationship status:

- 1. Resident
- 2. Away/inst: the person is temporarily away (in school, jail, rehab, etc.) and does not have any other permanent address
- 3. Away/other: the person is temporarily away (for another reason, such as travelling) and does not have any other permanent address
- 4. (Died/Passed away)
- 5. Nonresident
- 6. Have no contact
- 7. Delete
- 8. Duplicate

Below the options is a table with the following columns: FIRST NAME, REL, SP, SEX, RES, MO, YR, DUP, SAM, WH?, MAR, PRT, NEW SP, FIRST NAME, SP, SEX, SP, RES, COMMENT. The table contains three rows of data:

	FIRST NAME	REL	SP	SEX	RES	MO	YR	DUP	SAM	WH?	MAR	PRT	NEW SP	FIRST NAME	SP	SEX	SP	RES	COMMENT
	KEVIN	3	3	1	1						1		Polly		2	5			
	KAREN	6	6	2	4	1	2016												
1	JOHN	3	3	1															

At the bottom of the window, there is a status bar with the following text: 0400510010 | Version Date: 1/8/2018 | Version Time: 12:00PM | SecA2.ChildTab.Child[3].X056AResStat | CORENG | 9/17/2018

Figure 1. A child roster in the Blaise 4 HRS instrument.

Our new plan was strongly influenced by one of the methodologists here who leaned in the direction of using dashboard-type screens for the new design in Blaise 5. Roughly, the notion is that you start with an overview screen on which each item (person or pension) is represented by a clickable button or link which would bring up another screen where editing could take place or new questions could be administered. Completing the detail screen would then automatically return you to the dashboard, with some indication of which item had just been completed and of which ones still needed to be examined. Once all items were marked as examined, the button to allow movement past the dashboard would finally appear. We made some efforts to adopt this idea directly, which will be mentioned briefly below. However, in the end, we effectively reversed it, essentially going through the detail screens one by one and placing a stand-in for the dashboard at the end of the sequence in the form of a summary screen, which will be discussed in detail below.

Because we had some experience with customizing the page handler in Blaise 4 IS, and had also made early efforts to get into customizing the Blaise 5 browser data entry client (while it was in the older Asp.Net architecture), our first attempt to implement new rosters involved customizing the data entry client to add a roster page built entirely from our own javascript code. (HRS has adopted the practice of referring to all data entry customizations as "routers" which will continue below for brevity's sake.) This turned into a working prototype of the pension roster which showed a partial grid vaguely like our Blaise 4 design, but emphasized manipulation of the line items in the grid with obvious visual cues for adding, deleting and editing. In addition, detail screens were in-page modal boxes with a set of related input boxes (in this case employer details in a form layout). The general notion was that data would be loaded into the javascript from the preload array, would be manipulated entirely within the javascript, and then saved to the post-roster copy of the preload array similar to the way it was handled in the past. While on the screen, user interaction would not need to be constrained by the Blaise rules engine or layout schemes, making it easier to potentially add manipulations like drag-and-drop functionality, for example. In the end, concerns about whether the plan would be very future-proof caused us to shelve it, and that turned out to be prescient. Originally, we held out hope that we would be able to do both our interviewer-administered and self-administered surveys in the browser client and we thought that the early work we did would be usable in some form in the version we fielded. However, we ultimately were asked to split our work between Windows DEP and browser DEP, which would have required two separate routers, doubling the workload. In addition, the performance advantages of the MVC architecture change which arrived late in the game (or rather performance downsides of the Asp version) would have forced us to abandon any code based on that original design anyway. In the end, we decided to rely much more heavily on out-of-the-box capabilities of Blaise 5 to build the rosters, but the lure of the design freedom offered by the router approach will likely cause us to revisit the idea when we start looking at the next wave of HRS.

HRS ultimately found a compromise solution, which adopted a detail-page summary question strategy that had previously been used only for (the more conceptually complicated) pensions, but now would be used for all rosters in the new survey (our terminology can get confusing here: the "summary" question is a sort of shortcut on a detail page, but the "summary" screen which follows all the detail pages aggregates all the roster information on one table for group review). The original idea from our Blaise 4 instruments was that the respondent would be read a description of each item preloaded from a prior wave (a 'summary' of each item, which might sound something like: "you had a 401K plan from Ford, where you worked from 1980 until 2002, that you called your 'Ford plan'") and then allowed to confirm that nothing had changed with this single answer (thus skipping the remainder of the detail questions), or to deny that the information was correct and edit the item in detail. Following the summary questions for preloaded items, the respondent would get asked if they wished to add a new item as many times as needed until they said 'No' at which point the loop and thus the questions would end (there is a maximum number of items specified, but it is set considerably higher than the largest amounts we have gathered across many waves). Once all the identifying information had been collected or confirmed, the respondent would immediately enter a new loop through the same items for a series of follow-up questions (a short series in the case of people and a very long series in the case of pensions). Other than layout, the major new component in the Blaise 5 design was a summary screen between these two parts, allowing final validation and navigation for corrections and acting as a gate, closing of the task of compiling what was essentially a full list of identifiers.

This came about in part because HRS put a lot of effort into avoiding the need for significant vertical scrolling and eliminating any possibility of horizontal scrolling. In most of the instrument, we present only one question per page. We had to do a fair amount of experimentation to see what questions could practically fit in a detail screen, and that naturally pushed us towards splitting up (in the manner previously described) questions that in prior waves had fit in one table. We also were pushed in this direction (splitting the old table) by the need to improve the clarity of questions and answers for self-

administered respondents, since prior designs had sacrificed ease of understanding in order to cram more question-columns into the Blaise 4 roster.

We finally landed on a conceptual basis for this split as follows. One of our co-PIs did an analysis of the frequency of changes in status between waves among the various roster questions. She proposed that we design based on a distinction between variables that change more rarely or due to mistakes (such as first name), and those that might be expected to change, perhaps more than once, over multiple waves (such as marital status). This happens to map on to the fact that the former may be of lesser (or sometimes, no) use to users of HRS data (they exist more to allow tracking and identification internally). Additionally, we not only have immediate follow-ups where keeping track of which person or pension one is referring to may be tricky; we often return to the same set of people later for follow-ups in other sections covering different content areas. So the first set of variables constitute a sort of contract between the respondent and interviewer and/or survey designer on a common reference set for immediate and much later follow-ups.

In Blaise 5 what this looked like was a separate single question screen for each person or pension, making it easy and clear for a typical respondent who simply needs to verify that we have the correct set of children, pensions, etc. The implementation was tricky, however, as we wanted the detail questions to appear on the same screen if the respondent answered 'No' to the summary/verification question. In other words, the respondent should initially see a question asking if a set of information is correct; upon answering "No, it needs editing" and clicking the "Next" button, we want the respondent to see the editable questions appear below on the same page. This was a difficult proposition since HRS had set a requirement that nearly all fields would allow "Empty" in our self-administered version, meaning that the engine is prone to skip right past a bunch of empty questions; not to mention that frequently these questions will be pre-filled by preloaded values. Blaise 5 offers ways to force routing checks within a browser client page, but when we looked at them, we decided they wouldn't be easy to maintain in our complicated loops due to overhead in layout. That is, HRS is too big to practically use the Blaise 5 layout editor (the "Layout" tab while the .blax file is open), we have multiple programmers working on the same survey, and we use define statement to remove parts of the instrument from testing as needed which has the side effect of ruining any work spent on the layout editor by permanently eliminating the temporarily removed bits. We were able to achieve this desired result unexpectedly with simple routing through code alone, with the quirk that the final question on the page had to be an enumerated type. We assume that underneath, Blaise 5 is failing to leave the page due to the appearance of more on-route questions and this works fine for us, but this may prove to be a brittle solution in the long run if it turns out that we are exploiting a mistake in Blaise.

```
3359     IF X055APPN = EMPTY THEN
3360         A208ANewPerson
3361     ELSEIF A221AConfirm <> NO THEN
3362         IF SELFADMIN AND X058AFName <> MISSINGNAME AND X058AFName <> MISSINGNAME_SPN THEN
3363             A221AConfirm
3364         ELSE
3365             A221AConfirm := NO
3366         ENDIF
3367     ENDIF
3368     IF A221AConfirm = NO OR A208ANewPerson = yes THEN
3369         GROUP_EditChild_GROUPTXT
3370     ENDIF
```

Figure 2. Simplified code snippet for child detail summary page.

```

117 <RouteItemLayoutInstructions RouteItemName="SecA2.ChildTab">
118 <Instructions>
119 <GridInstruction Locator="Before" RouteItemsPerPage="3" />
120 <TemplateInstruction Locator="Before" TemplateName="HRS_SectionIndex" TemplateTarget="MasterPage" />
121 </Instructions>
122 </RouteItemLayoutInstructions>
123 <RouteItemLayoutInstructions RouteItemName="SecA2.ChildTab.GROUP_CHILDGATE">
124 <Instructions>
125 <GridInstruction Locator="Before" RouteItemsPerPage="1" />
126 <TemplateInstruction Locator="After" TemplateName="HRS_MasterPage" TemplateTarget="MasterPage" />
127 </Instructions>
128 </RouteItemLayoutInstructions>

```

Figure 3. Entries for switching master page TemplateTarget and RouteItemsPerPage in the .layout file (manually added).

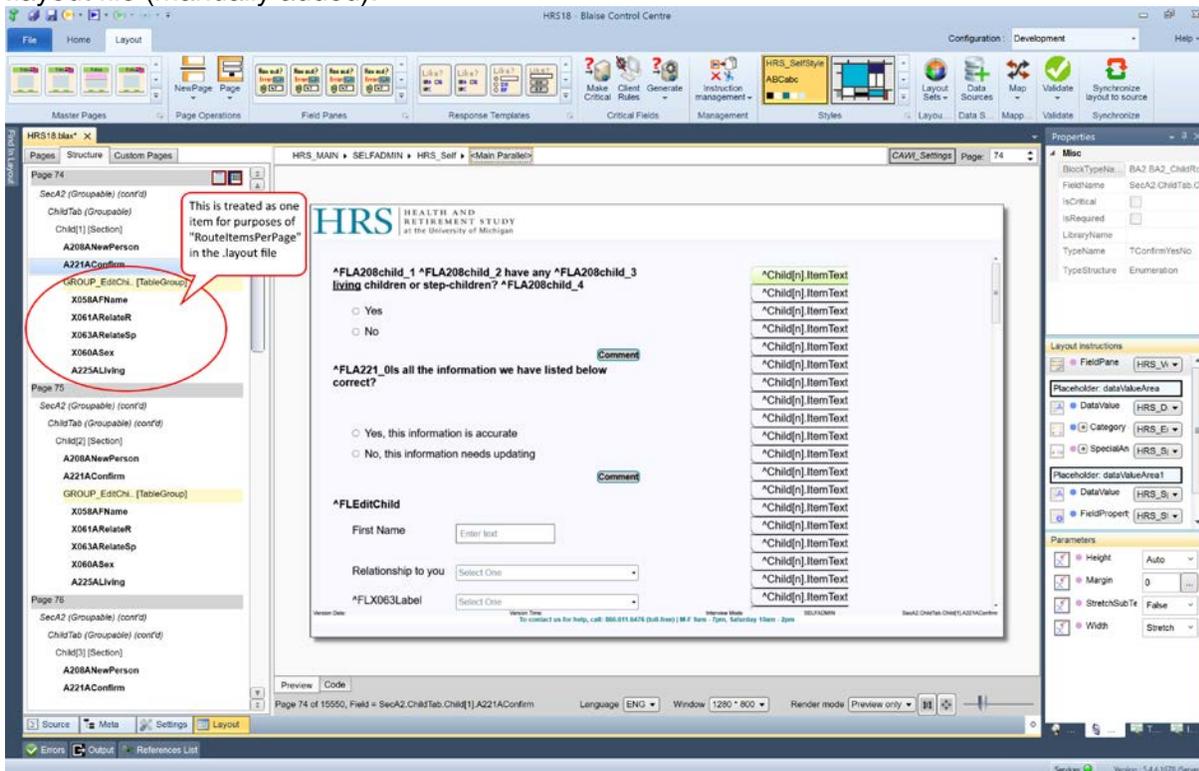


Figure 4. Even though we do not use it for editing, we can however use the layout view to see edits we make manually to the .layout file (namely switching to 3 questions per page in the detail sections) – the above screenshot shows the detail summary and detail question set page for one child.

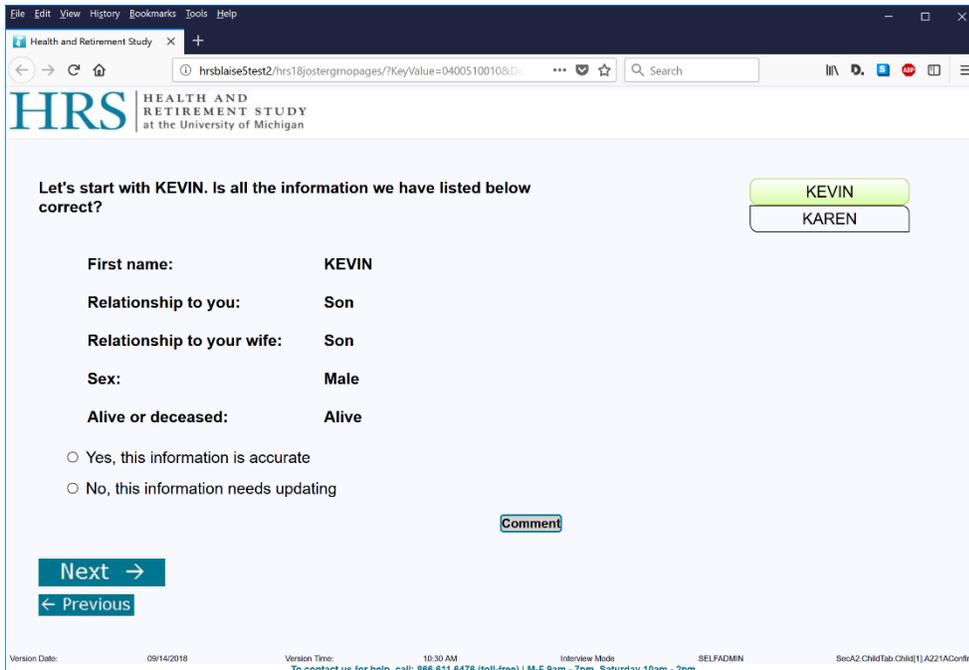


Figure 5. This is what our child detail summary screen looks like for a preloaded child reported in a previous wave. Note the names on the right are clickable navigation buttons and the current child is highlighted.

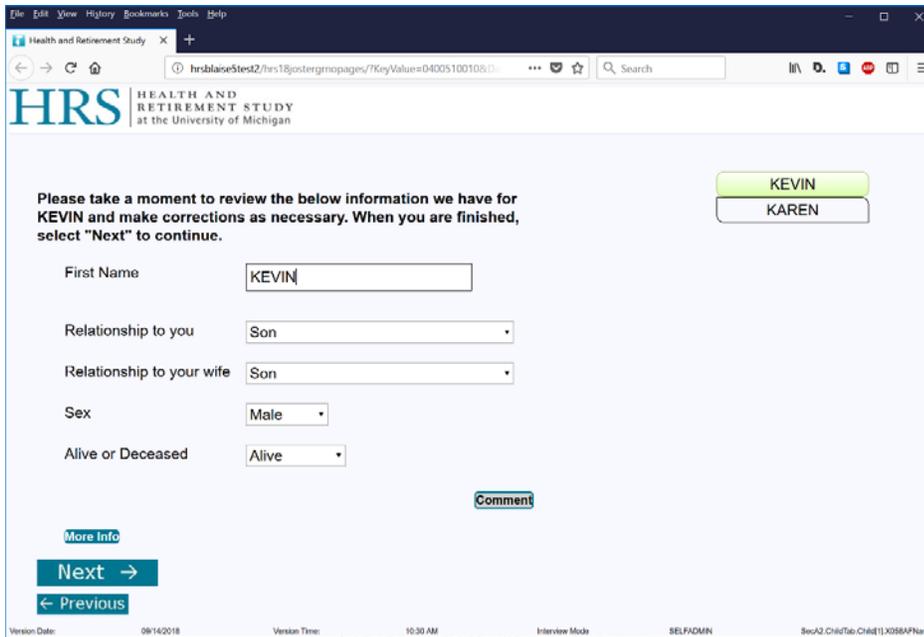


Figure 6. If the detail summary question is answered “No”, after clicking “Next,” it is removed from the route and this group of detail questions appears in its place. If it had been answered “Yes” instead, the survey would have proceeded to the next child detail summary page.

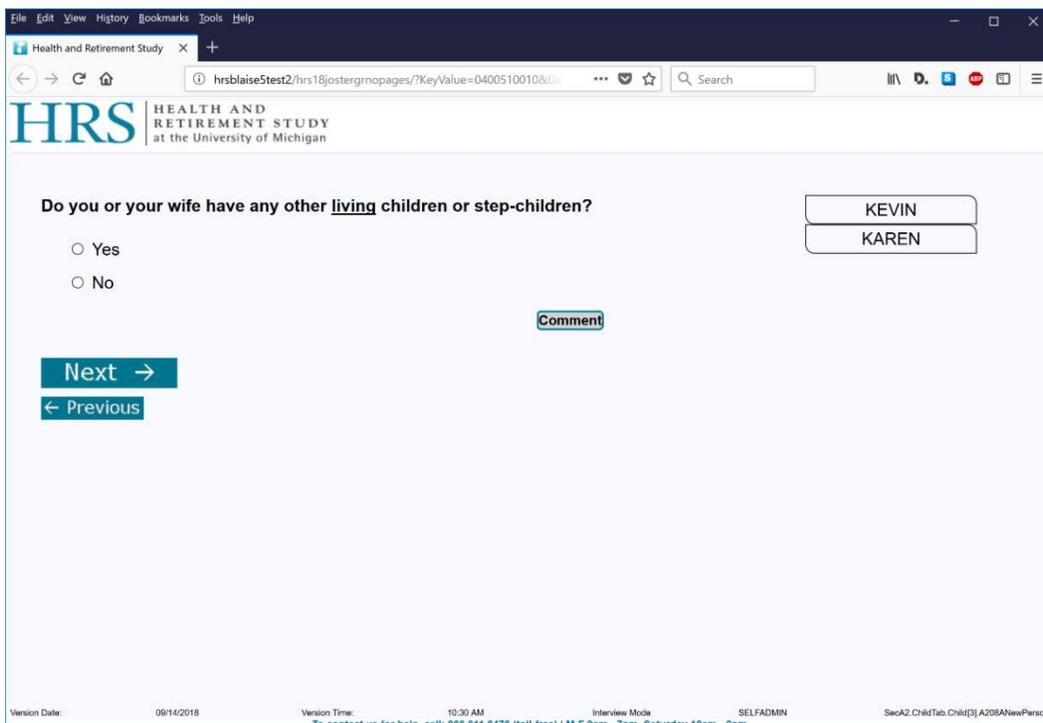


Figure 7. After pages for any preloaded children have been verified or edited, the survey displays pages allowing new children to be added. Similar to the detail summary question, one answer (“Yes,” in this case) opens up a group of questions (see Figure 5); the other answer proceeds to the roster summary screen. Note that no button on the right indicates the current page at this stage.

Because HRS had been unable to make a satisfactory dashboard design work, we opted instead for a roster summary screen following all of the verification, new item and detail questions. The summary screen needed to be able to return the respondent to any individual item if they discovered a corresponding mistake in the summary. In addition, when on the individual item screen, it needed to be possible to return to the summary screen with one click (i.e. not by advancing page by page back through multiple potential items). We also already wanted to have the list of identifiers on the screen so the respondent was able to see who they had added and navigate backwards even before they reached the roster summary. Finally, the one click roster summary button needed to be hidden until they actually arrived at the roster summary (which was a problem to implement for a long time, but finally solved using the 'IsVisited' property). These navigation buttons were originally placed on the left of the screen so that they would catch the respondent's attention. Later we moved them to the right side in order to privilege the question itself. In the end, we came to see the buttons less as navigation options for the first pass through and more as a list to help with workflow (and one that updates if the names are edited or new rows are added). Once the respondent reached the summary screen, instructions would then point out the navigation buttons because they would be necessary to reconcile problems found at that stage in an efficient manner.

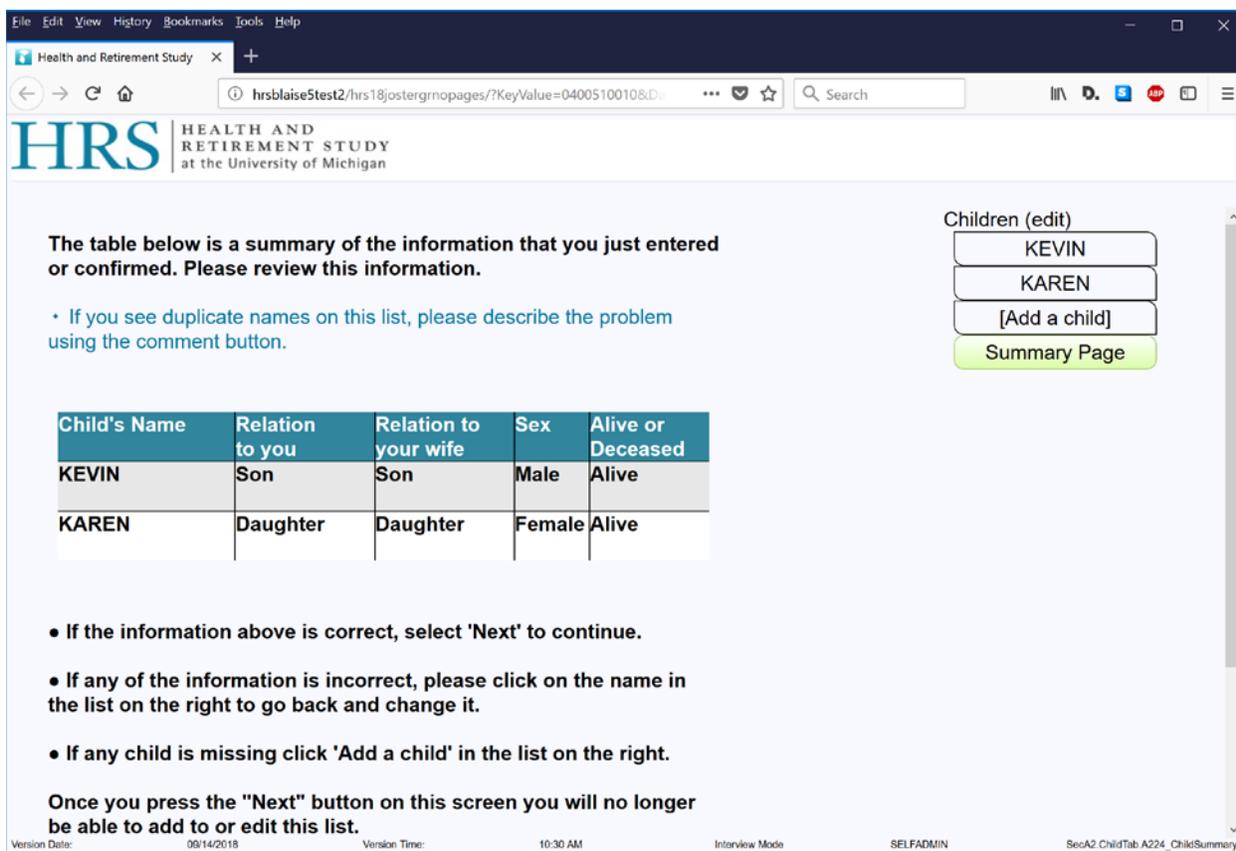


Figure 8. The roster summary screen. Despite our efforts, the “Next button is just off the screen to the bottom, necessitating some vertical scrolling. Note that, having arrived at the summary page, the buttons for “Add a child” and “Summary Page” have appeared in the list on the right (the buttons were always there, but they were hidden until the “IsVisited” property of the summary page was true). After this point, if the respondent goes back to earlier pages, these extra buttons remain visible to facilitate easy navigation.

HRS chose the Section Index Page template pattern in the Resource Database for its roster design as it allowed for normal routing through the questions, while at the same time providing an alternate means of quick navigation. Parallel blocks would have been an alternate possibility, but the Section Index templates ultimately worked out a bit better. As is often the case with our more complicated sequences, we had to go against the grain of what seemed to be intended. The section index templates provide the quick button navigation we need, but appear to be intended for large “sections” of a whole survey, not single pages. In other words, under the hood there appears to be one big section index list for the entire survey – you can’t define multiple section index lists. As a result, we have to gate off each roster from the others or else buttons for previous rosters appear in later ones. Gating off rosters and other sequences is not new to HRS, but we had been trying to move away from doing that for the sake of self-administered surveys, since gates might prove confusing to respondents trying to navigate the interview by themselves. Gating off the rosters caused some new additional problems as well, which will be described below. However, since most of the functionality for the navigation buttons is built in, we had relatively little trouble adding this functionality on top of our existing template designs in the resource editor.

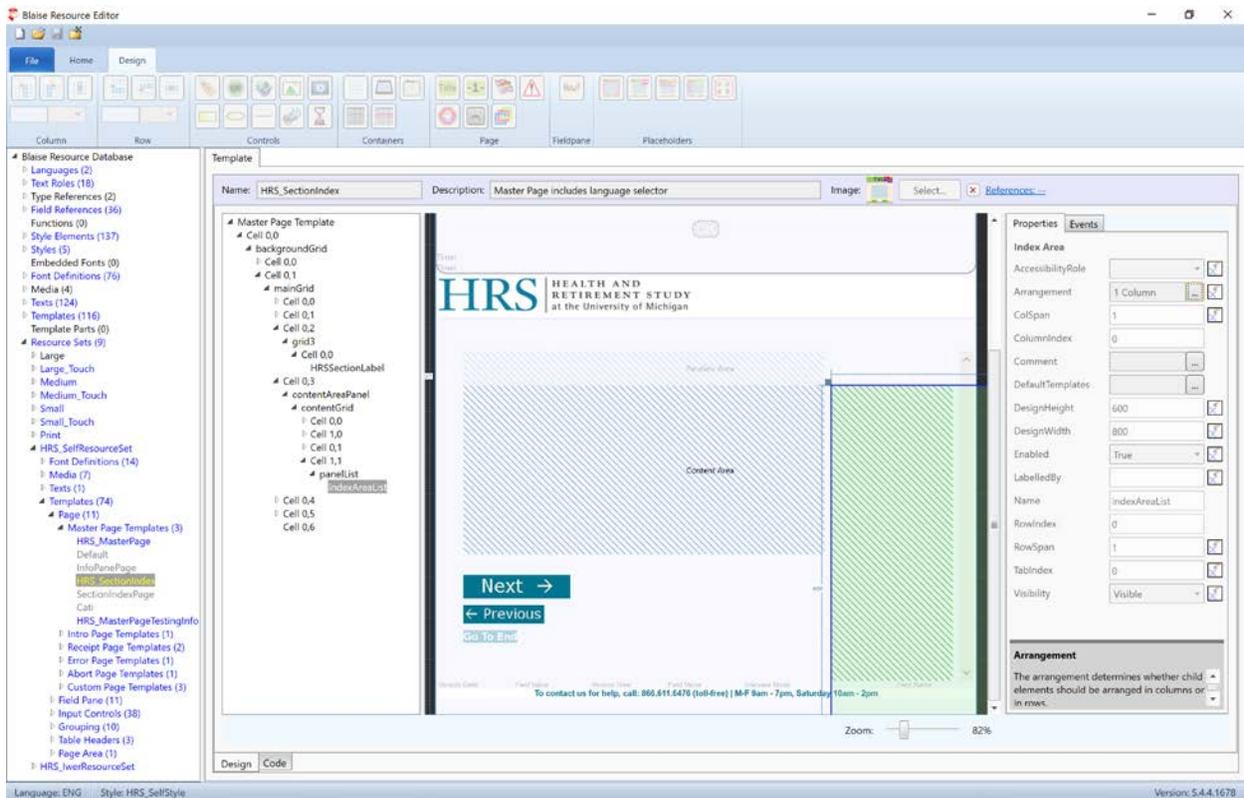


Figure 9. The IndexAreaList master page template.

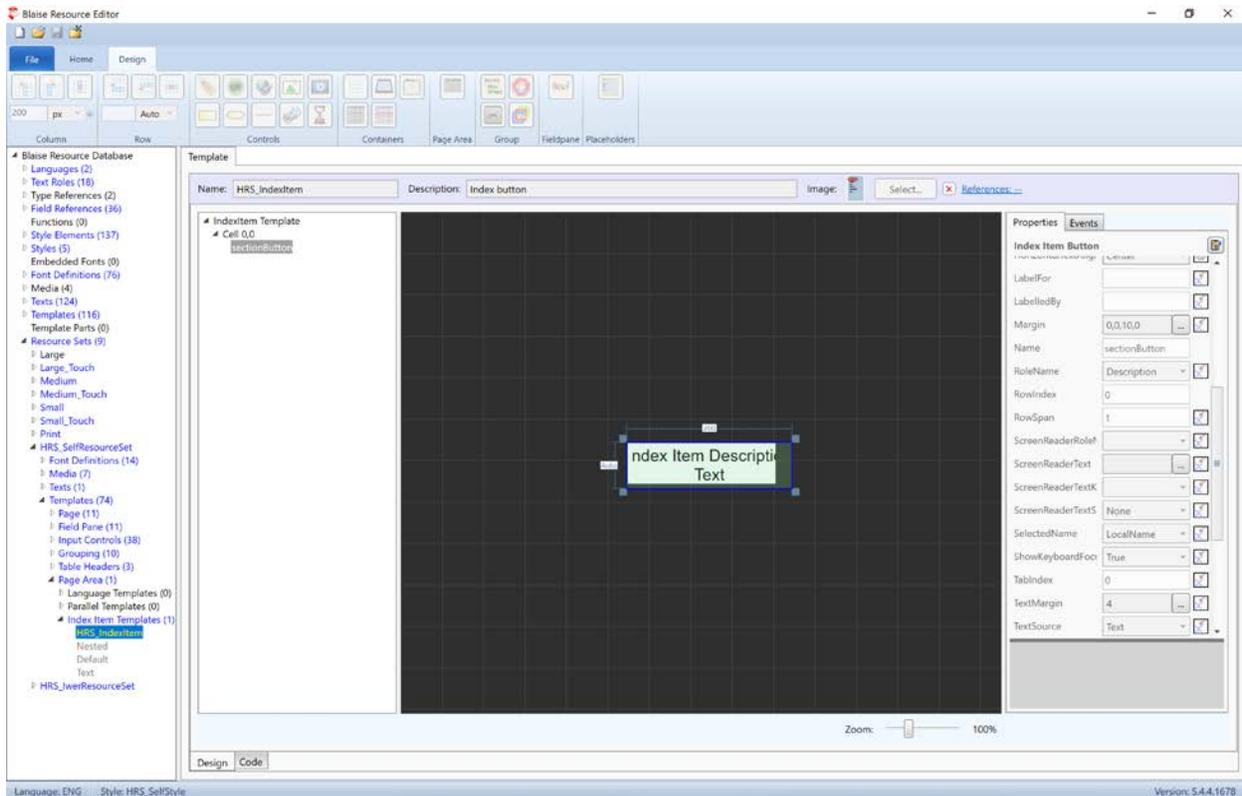


Figure 10. The Index Item button template.

Gates posed one particularly big problem that resulted in a lot of spinning of wheels on solutions that never worked out. We want a gate to trigger based on a screen outside of a loop (basing it on a loop screen poses its own special set of problems because you can't predict at design time exactly where the loop will end in a session). Typically such a screen is a summary or an informational screen with no answer items, only a "Next" button (and "Previous" button). In Blaise 4 and in the interviewer-assisted mode of HRS in Blaise 5, these screens would have a simple codeframe with only one option ("1. Continue") and the "noempty" attribute ensured that a value of "1" would always be entered. This was important for later routing that depended on this gate being closed and it was important for external tools which HRS uses that depend on the value being recorded in the audit trail as well. However, self-administered interviews in Blaise 5 were beset with a number of problems in this case.

First, no answer item would be presented to the self respondent both because that represented an extra burden and because the requirement that every screen allow empty meant that it was not going to be possible to police whether the answer was actually given. Inconsistent answers on this screen would make the gate useless. What we wanted was to assign "1" to the gate question when the "Next" button was pressed, without even showing the respondent the "1. Continue" code in the first place. This turned out to be a fairly easy thing to do when it came to the "Next" button itself. However, the respondent can also advance past the screen in other ways, such as swiping, or pressing enter (we do want to support keyboard use for respondents who may have trouble with a mouse). There is no provision in the resource database for attaching actions to pressing enter to leave the page. What this meant is that a self respondent pressing enter on that screen would fail to close the gate, which would result in many incorrect skips. In the end, with direct help from CBS, we found a fairly convoluted workaround to this problem which would trick Blaise into making the assignment. This is where we discovered the "IsVisited" property, which was part of the solution.

```

4550     A224_ChildSummary
4551     A224_ChildSummary.AlienActionEvent := '1'
4552     IF A224_ChildSummary <> 1 THEN
4553         A224_ChildSummary := FUNC_SetGateFieldsOnPreviousFieldIsVisited('A222_ChildGate', A224_ChildSummary.IsVisited)
4554     ENDIF
4555     A222_ChildGate
4556 ENDGROUP
4557
4558 FUNCTION FUNC_SetGateFieldsOnPreviousFieldIsVisited : INTEGER
4559 PARAMETERS
4560     IMPORT
4561         piCheckCurrentField: STRING
4562         piTriggerFieldIsVisited : TIsVisitedFieldProperty
4563     RULES
4564     IF POSITION(piCheckCurrentField, CurrentFieldName) > 0 AND piTriggerFieldIsVisited = Yes THEN
4565         RESULT := 1
4566     ENDIF
4567 ENDFUNCTION

```

Figure 11. Simplified code snippet for gate.

This solution correctly made the assignment whether the “Next” button or the “Enter” key was used, but our troubles were not over yet because it failed to record the assignment in the audit trail. In the end, we solved this problem with a router in the new MVC architecture which we had adopted by that time. In short, we overrode the nextPage and nextField events (which covers both leaving the page via button and enter key) to check whether a specific property was present. If so, it would add an event to the audit trail that would account for the assignment, which we are able to read later from our tools.

```

protected nextPage(): IActionKeyValuePair {
    this.signalSupressService.suppressSignals();
    let nextPageResult = super.nextPage();
    if (nextPageResult) {
        this.alienActionEventService.checkForAlienActionEvent(0);
    }
    return nextPageResult;
}

protected nextField(actionObj: IAction, controlId: string): IActionKeyValuePair{
    let nextFieldResult = super.nextField(actionObj, controlId);
    if (nextFieldResult) {
        this.alienActionEventService.checkForAlienActionEvent(5);
    }
    return nextFieldResult;
}

```

Figure 12. Code snippet triggered by both events which can cause a forward page change from custom-action.service.ts, a new ActionService.

```

public checkForAlienActionEvent(kind: number) {
    if (kind === 0 || kind === 5) {
        const actionNameAssignField: string = 'AssignField';
        const fieldPropertyName: string = 'AlienActionEvent';
        const actionKind: string = (kind === 0) ? 'NextPage' : 'NextField';
        const detected: string[] = this.detectFieldProperty(fieldPropertyName);
        if (detected.length > 0) {
            this.auditTrailService.logActionEvent(actionNameAssignField, detected.join(','), AuditTrailLevel.None);
        }
    }
}

```

Figure 13. Code snippet writing an extra record to the audit trail service when triggered on a field with the correct property from alien-action-event.service.ts, another new class.

## 4. Conclusion

The development of these rosters for HRS in Blaise 4, starting back in 2001, had always been a rather complicated endeavor. Redesigning them for Blaise 5 proved far more time-consuming than we expected, and we did not expect it to be easy. As described above, the complications emanated from a number of sources. We would not have been doing this if we were not trying to run the whole interview in a self-administered web mode, and that, by itself, upended many of our long-held design assumptions and forced many new design requirements upon us. Blaise 5 itself presented a two-fold problem: we had to discover what options we had in Blaise 5 and learn how to implement them, while at the same time struggling with the fact that the system was new and that we frequently ran into bugs that blocked our progress until they were fixed, or requests for new features or workarounds in Blaise 5 which also took time to sort out. This discussion touched only briefly on the additional complications of having an interviewer-assisted mode (driven by some very different design assumptions) in the same instrument, but the end result was a difficult balancing act. With thousands of interviews now behind us, we can say that we seem to have largely succeeded in building rosters that worked reasonably well for both modes. We already have plans to dive in again next year, though, with the expectation that we will have a more solid foundation for the next effort having been through this process.

## **Part 6 - The Word List: A Blaise 5 Conversion Challenge**

*Rhymney Weidner – Survey Research Center, University of Michigan*

### **1. Abstract**

The Health and Retirement Study is a longitudinal study with a wide variety of question types. Transferring those varied questions into Blaise 5 has provided several challenges especially in the area of template design. The cognition section in particular has encountered this issue quite frequently. Even in Blaise 4.8, the cognition section had to utilize alien procedures to develop the question series known as “the word list”. For this question, a list of ten words are displayed on the screen for two seconds each. After they are displayed there are two “recall” screens where interviewers can enter the words that respondents remember. Both of these functions were achieved using alien procedures. With alien procedures for the MVC structure still a work in progress and a lack of development time, we had to develop a design for the word list series that could function entirely within the existing Blaise 5 structure. This paper will examine the challenges presented by this particular Blaise 4 to Blaise 5 conversion in three main areas: maintaining a similar look/feel to what was used in Blaise 4, maintaining data structure, and designing a suitable self-administered interface.

### **2. Introduction**

The Health and Retirement Study went into the field with Blaise 5 for the first time in 2018. A massive redesign effort was undertaken prior to fielding the survey in Blaise 5. Blaise 5 presented many new functions as well as taking away some old ones. In addition to changes in the software itself, the survey had to be adjusted to work for both interviewer administrated and self-administrated surveys. All of these changes had to be considered and accommodated during the survey development process. Each of the HRS survey sections presented unique challenges while programming in Blaise 5. The cognition section especially stretched the limits imposed by Blaise 5. Within this section there are several specialized questions that required special design considerations. While many of the other sections were able to use the templates designed for general use, the cognition section had some questions that required the design of a specialized template. Most important among these questions, and the most challenging in terms of design, was the word list. The word list displays a series of ten words on the screen for two seconds each. After the display there are “recall” screens where interviewers can enter the words that respondents remember. The recall screen design must allow for the interviewer to quickly and accurately enter words as the respondent recalls them. In Blaise 4.8 the word list recall was designed using alien procedures.

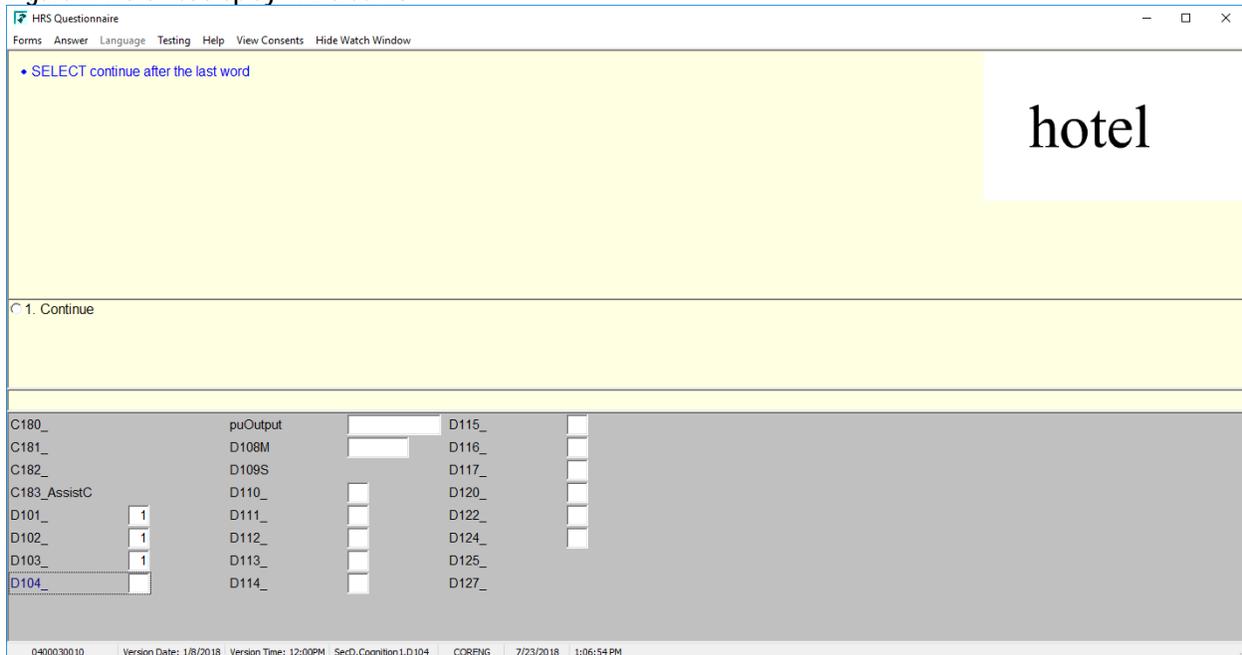
Due to a development time crunch, and the evolving nature of MVC Custom applications in Blaise 5, it was decided that the word lists would have to be developed using only Blaise 5 out of the box abilities. Thus began the whirlwind development of the word list.

### **3. Maintaining Blaise 4 Functionality/Appearance**

As mentioned previously, the word list consists of two equally important parts. First, the actual display of the word list. Second, the screen where respondents have the option to recall the words presented.

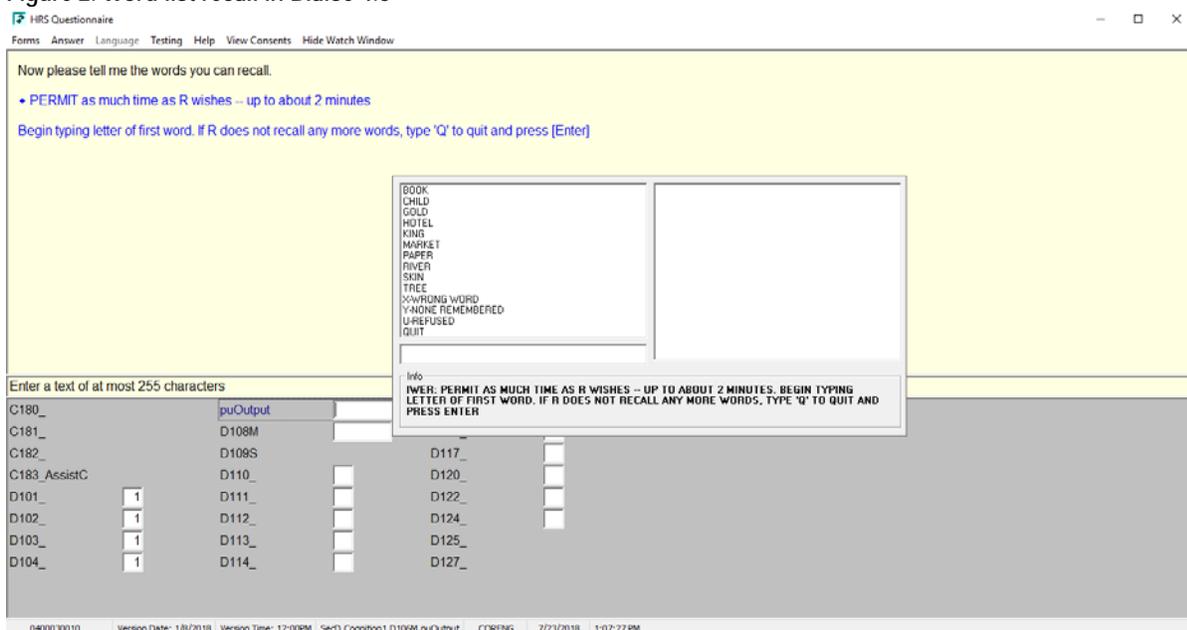
In Blaise 4.8 the list of words was displayed using an AVI file (Figure 1).

Figure 1. Word list display in Blaise 4.8



The recall portion of the word list was achieved through an alien procedure programmed in Visual Basic. It created a list of the words that allowed the interviewer to either make a selection by double clicking on a word using the mouse, or by typing the first letter of the word and pressing “enter”. Selected words were moved to the right hand side of the screen (Figure 2).

Figure 2. Word list recall in Blaise 4.8



Rather than go through the process of creating another AVI file for use in Blaise 5, it was decided to go in a direction that would provide more flexibility in the event of changes to the word list. A special template was created that made use of a timer and the ability to map to fields. One field was assigned an enumerated type where each code frame’s text was a fill, with ten enumerated options. A procedure

assigned the fills for each code frame based on which word list should be shown. The cognition section has four different possibilities for the word list and the fills allow all of them to use the same template. Using fills also made it so the word list can easily be updated or changed by simply updating the procedure that does the assigning. The template is automatically assigned only to the field where the word list should be displayed by using a template role that contains specific trigger text. Figure 3 shows an example of the timer and one label used to display the first word in the list. When the timer ticks it sets the first word to collapsed, the second word to visible, and reset itself for another two second display time. By utilizing these Blaise 5 features it was possible not only to recreate the functionality of the word list from Blaise 4.8, but to also improve upon it. The final word list appearance in the DEP can be seen in Figure 4.

Figure 3. Word list display template in Blaise 5

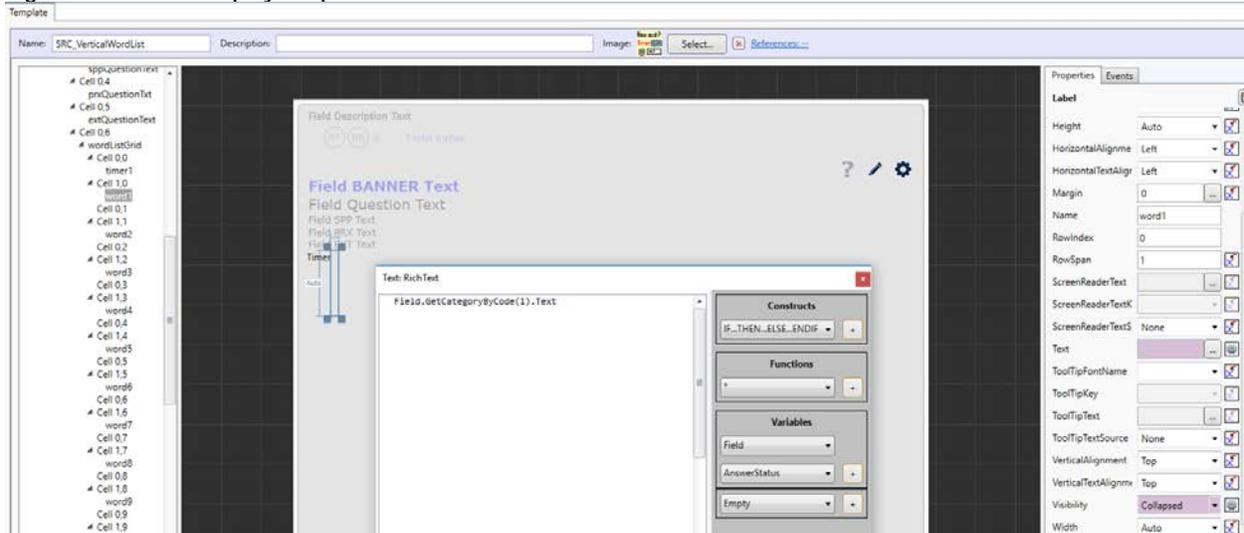
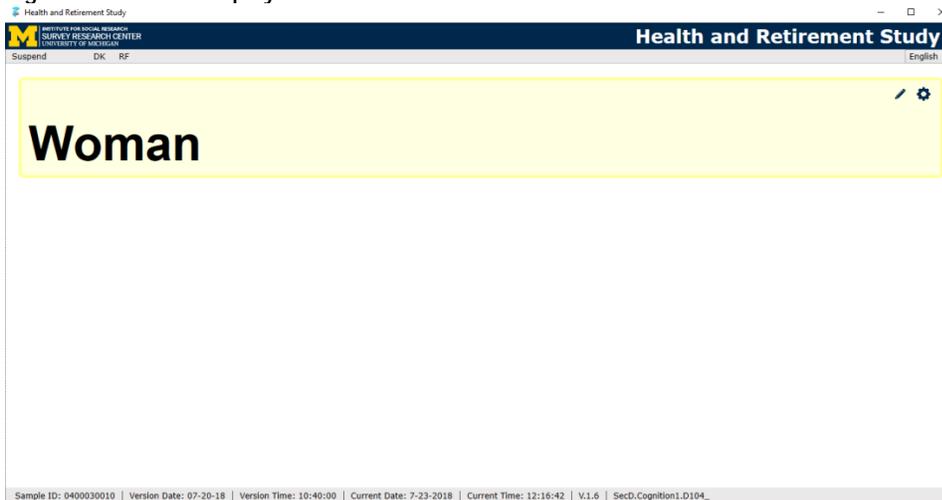


Figure 4. Word list display in Blaise 5 DEP

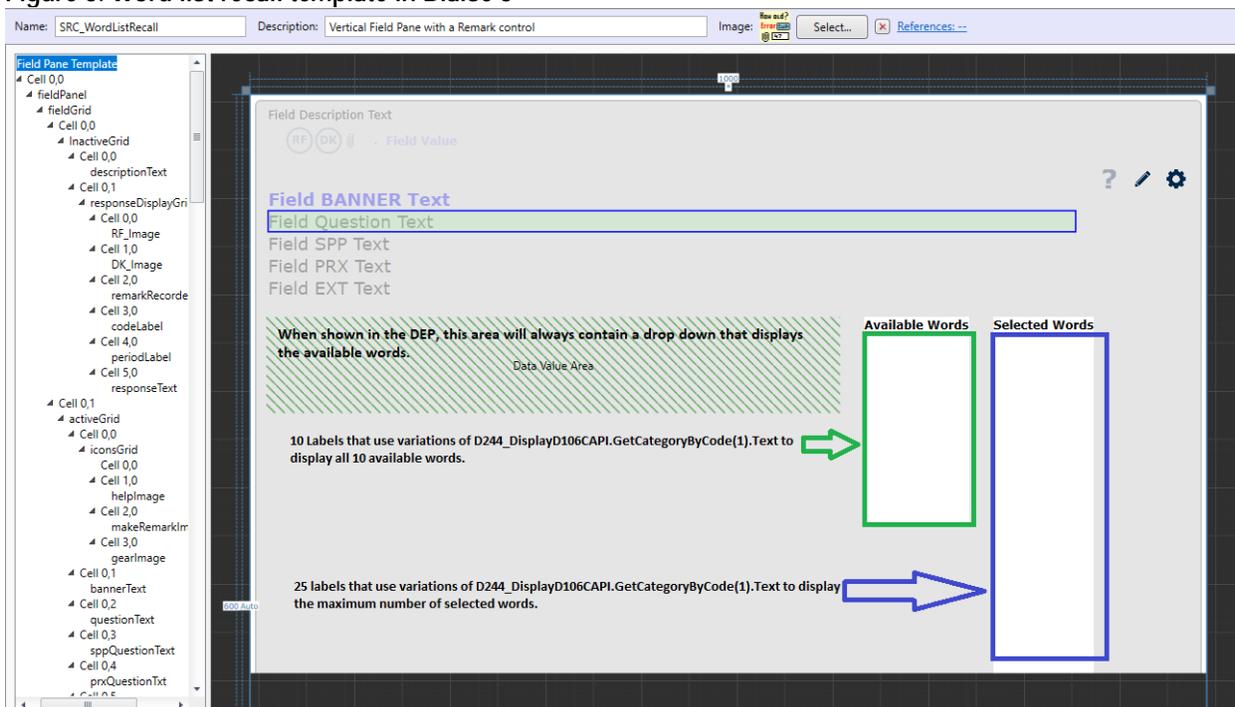


With only a two week time frame to develop the word list in Blaise 5, using MVC custom applications was not an option for creating the word list recall. Fortunately, Blaise 5 had some very handy out of the box features that could be implemented to recreate a similar functionality. As with the word list display screen, a specialized template was created that would be used only for the word list recall screen. Due to the heavy use of mapped fields in the templates, two separate word list recall templates were created. The

word list recall is shown to respondents twice, and each time different fields had to be mapped to record results, hence the need for two templates.

Each of the recall templates displayed a list of the words that allowed the interviewer to either make a selection by double clicking a word using the mouse, or by typing the first letter of the word and pressing “enter”. This template was far more complicated than the word list display and contained a multitude of parts.

Figure 5. Word list recall template in Blaise 5



To recreate the appearance of moving available words to a list of selected words, two groups of labels were added to the template. One column, “Available Words”, displayed all of the ten possible words in alphabetical order. The ability to map to fields was again utilized so that the labels reflected the code frame text for a given field. Rather than using the same field as the one that displayed the initial list, another field had to be used. This second field used the same idea, ten code frames with code frame text that was a fill populated through a procedure, but the word list had to be displayed in alphabetical order to make it easy for interviewers to locate words as the respondent listed them. As shown in Figure 5, there were ten labels on the template to display the ten enumerated responses for this ‘alphabetical field’. A second column, “Selected Words”, displayed words as they were selected. Words could be selected either by clicking on them in the “Available Words” list or by typing the first letter of a word to highlight it in the dropdown and pressing enter. The latter option is the one that interviewers were most likely to use, since they were trained to avoid using the mouse. Much like the original display of the word list, the text for each “Selected Word” came from mapping to a field and using the code frame text. A slightly complicated expression on the visibility for each of the “Selected Words” controlled whether they were displayed.

```
IF RemovedWords.ValueAsText = '' THEN
  IF POSITION('-', +
    TOSTRING(D244_DisplayD106CAPI.GetCategoryByCode(1).Code) + '-';
    D334_.ValueAsText) > 0 THEN
```

```

        'Visible'
    ELSE
        'Collapsed'
    ENDIF
ELSE
    IF POSITION('-' +
    TOSTRING(D244_DisplayD106CAPI.GetCategoryByCode(1).Code) + '-',
    D334_.ValueAsText) > 0 AND POSITION('-' +
    TOSTRING(D244_DisplayD106CAPI.GetCategoryByCode(1).Code) + '-',
    RemovedWords.ValueAsText) = 0 THEN
        'Visible'
    ELSE
        'Collapsed'
    ENDIF
ENDIF

```

Translated simply, if no words have been chosen and removed (the code is in a field called RemovedWords), the given label is shown if the word displayed in that label has been chosen (if the code is present in the final chosen values field, which is D334\_ in this case). If there have been words removed, then the label is only shown if its code is chosen (in D334\_) and *not* present in the removed words field (RemovedWords). This same expression is modified to reflect the appropriate code (1-25) for all of the possible selected words. Within the Blaise source code, there is a very long, complicated set of FOR loops and IF statements to control the addition of codes to the fields used by these statements in the resource database. Clearly, this complicated method of display is not ideal in terms of maintenance or when adjustments are required, but it does allow for the replication of the function achieved in Blaise 4.8 through alien routers.

The final result is a screen that echoes the important functionalities from Blaise 4.8, while also making some improvements (Figure 6).

Figure 6. The final word list recall screen in Blaise 5 DEP



The drop down recreates the ability to type the first letter of a word and press enter. The template and background code makes sure that the screen updates with the newly selected word listed in the “Selected Words” column. Prior to settling on the drop down, a look up was considered for this part of the template. However, after much consideration and testing, it was decided that the look up didn’t function with the speed required for an interviewer to be able to enter the words as quickly as they were given by the respondent.

As in Blaise 4.8, the only selection that can be made more than once is “X – Wrong word”, which allows the interviewer to enter if the respondent has said multiple words that were not really on the list (Figure 7). Signals prevent the interviewer from making conflicting selections or leaving the page without making any selections (Figure 8).

Figure 7. The final word list recall screen with words selected in Blaise 5 DEP



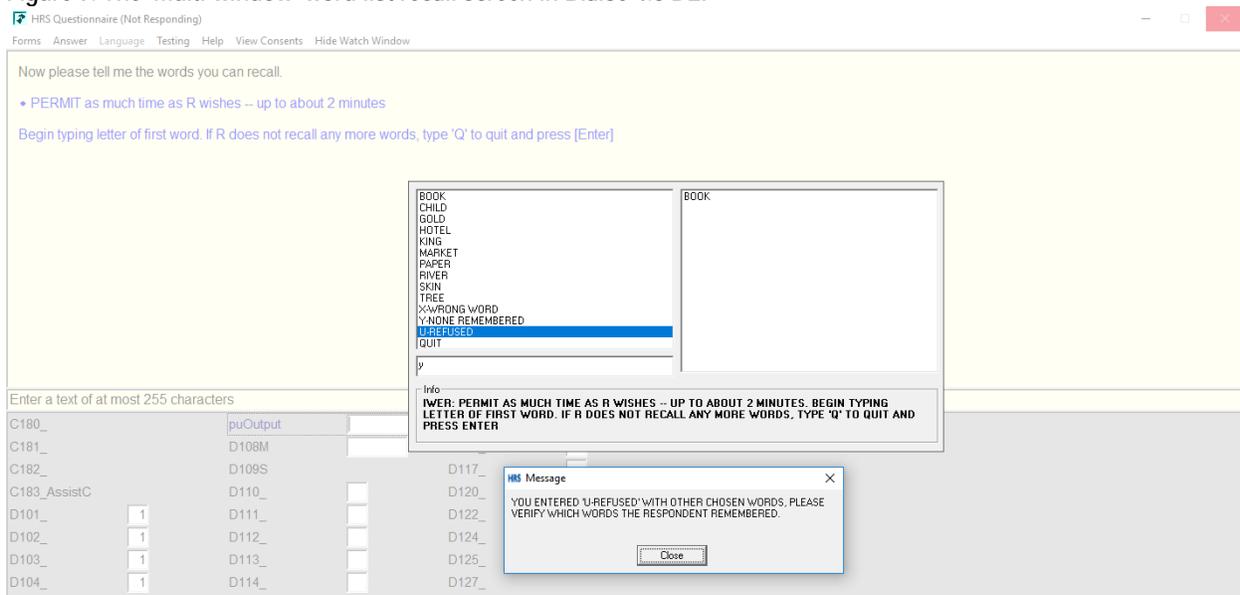
Figure 8. The final word list recall screen with an entry error in Blaise 5 DEP



To exit, the interviewer simply types “Q” and presses “Enter” on the keyboard or clicks on “Quit” under “Available Words”.

There were a few advantages to creating a screen with out of the box Blaise 5 features. The primary advantage was that the word list no longer opened in a new screen/window. This improvement created a more seamless survey experience for the interviewers and eliminated the focus issues experienced with such an arrangement (Figure 9).

Figure 9. The ‘multi-window’ word list recall screen in Blaise 4.8 DEP



Another advantage to using out of the box functionality was that it reduced maintenance requirements. Rather than having to modify an alien procedure, now the entire word list can be maintained within the data model. The new design also allowed for the use of signals/checks, rather than having to program signal like warnings in a custom application.

#### 4. Maintaining Data Structure

The Health and Retirement Study is a longitudinal study, which made it essential that the data structure was maintained with the redesign. One requirement was that the final data needed to be stored in a string. Since the template gathers data in a drop down (an enumerated field), a special sequence of code had to be created to transfer the data to a string. This sequence also had to correctly translate the potential for multiple occurrences of the “X – Wrong Word” selection into only codes of “11”. These selections could have values of 11 or 14-25 as they were recorded in the drop down.

In the Blaise 4.8 alien procedure, the “Refusal” option for the word list was recorded as ‘99’. In Blaise 5, a refusal is recorded as ‘98’ by default. To negate this difference the value was back assigned to the final field as ‘99’ whenever “Refusal” was selected.

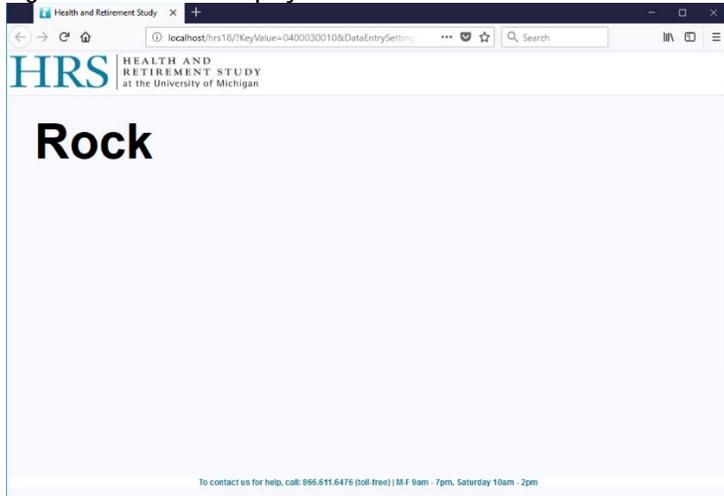
An additional challenge, in terms of data structure, was what to do about changes required to make the word list friendly to users taking it via the web, our self-administered surveys. The selected solution was to create a separate field that is only shown to the self-administered interviews, so that it was possible to gather two different types of data for essentially the same question.

## 5. Designing the Self-Administered Interface

Designing the self-administered interface for the word list was actually easier than designing the interviewer administered interface. A self-administered version of the survey had never been done, so designers were less opposed to trying new layouts.

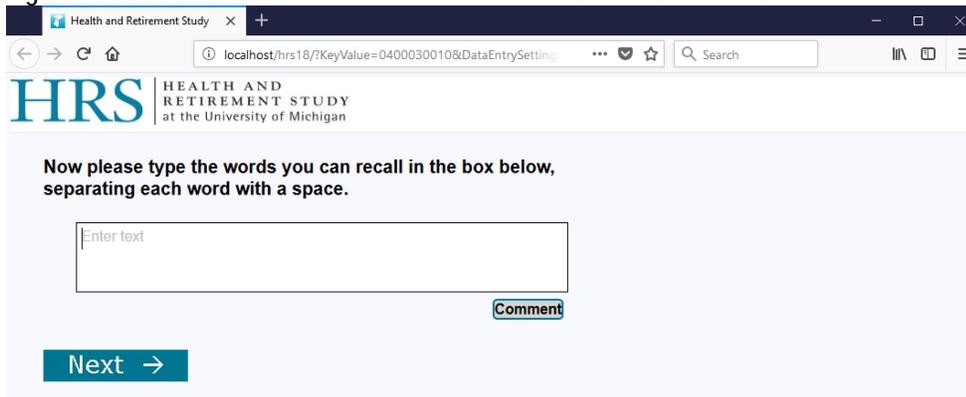
The display of the word list for self-administered was essentially the same as the interviewer administered version. Both show the words in a large font, again making use of labels and timers to display each word for an interval of two seconds (Figure 10).

Figure 10. Word List Display in Firefox for a self-administered interview in Blaise 5



The recall portion of the word list was much simpler for self-administered surveys. Since the question is testing if respondents can remember the words they were shown, the design could not include the original word list as the interviewer administered version did. Self-administered respondents were given an open text box where they could type the words they remembered, separating each word with a space. Within the Blaise source, a procedure evaluated how many correct words they entered. While the procedure could eliminate case differences by using the UPPERCASE function, it could not correct for typos or misspellings. The procedure then assigned the self-administered values back to the string used for interviewer administered data so that the final data was a string of numbers corresponding to the codes for each word typed. Writing the values back in this manner kept the data as it had been for many waves, rather than the entered string of words (Figure 11).

Figure 11. Word List Recall in Firefox for a self-administered interview in Blaise 5



## 6. Conclusion

The HRS conversion to Blaise 5 was and continues to be a process filled with challenges. Massive changes to the design of screens provided some of the biggest hurdles and required creative thinking to devise solutions. While these design obstacles were difficult, the results sometimes provided opportunities for improvement over Blaise 4.8 designs. The word list is one such example. The Blaise 4.8 word list design required the use of alien routers, which complicated code maintenance and created an interrupted feel to the survey with the word list appearing in another window. While the Blaise 5 design is quite complex, it took advantage of out of the box Blaise 5 features to eliminate the need for an alien router. This change eased code maintenance requirements as well as provided interviewers and respondents a smoother survey experience in the cognition section. In conclusion, a transition to Blaise 5 may require some resourceful programming, but the capabilities of the new Blaise holds a great potential for improvements over its Blaise 4.8 counterpart.

# Part 7 - Designing and Implementing a Web Component

Andrew L. Hupp - Survey Research Center, University of Michigan

## 1. Background

As noted earlier, the Health and Retirement Study (HRS) is a longitudinal study of over 20,000 adults age 50 and over that collects health and economic information. Prior to the 2018 main data collection, the HRS has collected its data using traditional interviewer-administered telephone and face-to-face modes. As part of the 2018 main data collection, the HRS introduced a self-administered web response option to the mix.

## 2. Design

### 2.1 Sample

A subset (2,247) of cases assigned to the telephone mode for the 2018 wave were selected to receive the web response option. Cases were randomly assigned to one of four replicates (see table 1) based on when all members of the household completed their interview (related lines are assigned to the same replicate). The first replicate was further sub-divided into two parts (a & b). The smaller replicate (1a) was used to ensure the system was working as expected before releasing additional cases. Cases came from across the different cohorts (i.e., age groups; AHEAD is the oldest, LBB the youngest) in the HRS.

**Table 1: Replicate Assignment**

Cohort	Replicate 1a	Replicate 1b	Replicate 2	Replicate 3	Replicate 4	Total
AHEAD	0	5	12	6	0	23
CODA	0	12	7	8	1	28
HRS	51	153	157	169	16	546
WB	0	93	79	69	20	261
EBB	0	173	178	186	40	577
MBB	0	217	260	242	92	811
LBB <sup>1</sup>	0	1	0	0	0	1
Total	51	654	693	680	169	2,247

### 2.2 Experiment

As part of the 2018 data collection a two factor non-contact experiment was implemented. Reminder interval timing (short (~6 day) v. long (~12 day)) was contrasted with mode switch timing to phone (short (6-week) v. long (12 week)). Cases were assigned to one of four non-contact sequences.

1. Short interval (~6 days) between reminders; switch to calling after 6 weeks
2. Short interval (~6 days) between reminders; switch to calling after 12 weeks
3. Long interval (~12 days) between reminders; switch to calling after 6 weeks
4. Long interval (~12 days) between reminders; switch to calling after 12 weeks

All cases are assigned to web first. Non-complete cases are followed-up with by telephone (phone is available immediately to accommodate call-ins, but are not actively delivered for calling until the 6 or 12 week mark depending on their assignment). Once contact<sup>2</sup> has been established, reminder follow-ups are

<sup>1</sup> The Late Baby Boomer cohort was excluded from the web since there baseline interview occurred during the 2016 data collection. One case was selected since it was part of a household with a member in another cohort that was selected.

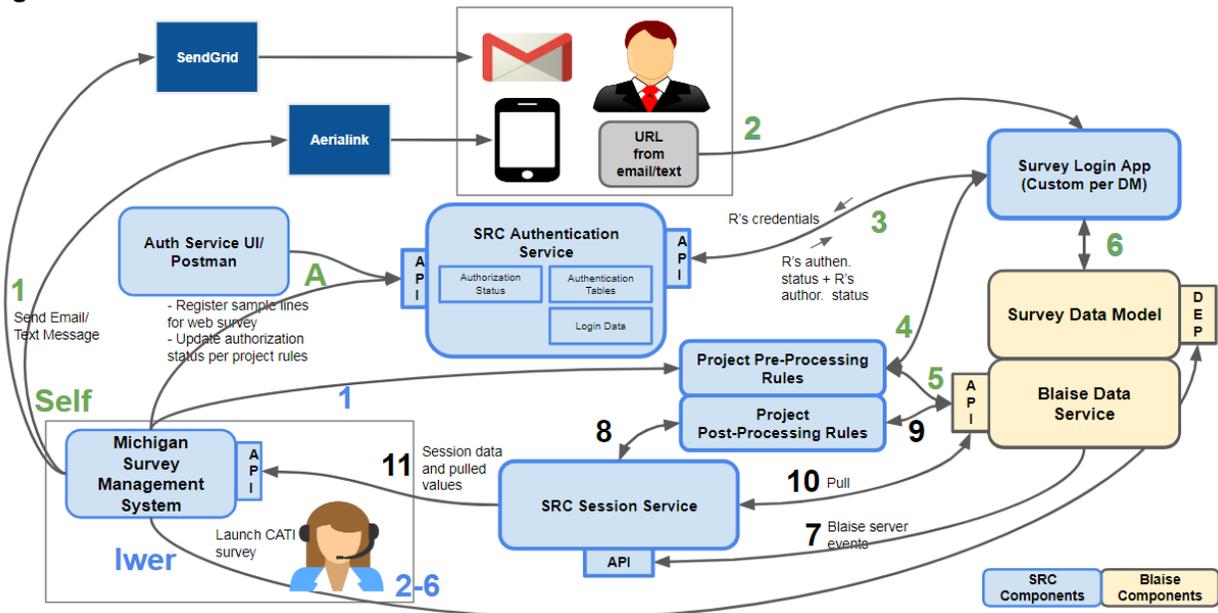
<sup>2</sup> Defined as completion of Sections A and A2.

based on their last activity (e.g. days since they last accessed the survey) rather than at a prescribed time. Contact cases follow the pattern of non-contact sequence one. They may receive the same amount of communications (2 letters and 6 emails) as that group but the timing of those reminders is dependent on their actions (or inactions). If they methodically answer a few questions every day after the first email, they will never receive another reminder from us. If the survey is not completed via web they eventually switch modes to phone to be followed-up with by an interviewer. That transition will always be longer than 6 weeks since there was at least a one delay based on having been in the survey.

### 3. Systems

The technical ecosystem used for the web component can be seen in figure 1.

Figure 1: HRS Online Flow



#### 3.1 Michigan Survey Management System

The Michigan Survey Management System (MSMS) (see figure 1 in lower left in blue) manages the study protocol. It communicates with third party services to send communications (emails, text messages, etc.) when criteria in the protocol have been met to send the next communication. Interviewers launch Blaise to administer the phone interview from MSMS. Contact attempts entered by the interviewer or created by the system are stored in MSMS for interviewers and managers to see. MSMS also stores information pulled back from the instrument at the end of each data collection session. Some of this information is used to determine the next step in the protocol.

#### 3.2 SendGrid

SendGrid (see figure 1 in upper left in blue) is the third party email provider used to send emails. The protocol loaded in MSMS determines when the next email to a case should be sent. When the criteria have been met MSMS communicates with SendGrid (via an API) and sends the relevant information (email address, email template, etc.) needed to send the email. SendGrid sends status information (sent, not sent, opened (if this feature is enabled), etc.) back to MSMS.

### **3.3 Survey Login App**

The survey login app (see figure 1 in upper right in blue) and its associated services are needed to determine whether a case trying to access the survey should be allowed in. The survey login app provides feedback if the case is not allowed in (invalid credentials, out-of-date browser, survey completed, etc.). The survey login app captures the device and communication (which email) used during the login attempt and the result of that login.

### **3.4 Project Processing Logic**

The HRS has project specific processing logic (see figure 1 in middle right in blue) that does two things. Prior to the launching of a case it checks to see if there is a related sample line and whether than survey has completed sections A and A2. If a case does exist and those sections are complete is takes the shared information between the cases and updates the values based on the first respondents responses. At the conclusion of a survey the logic checks to see if there are any adjustments to the household roster. If a new spouse has been entered a new sample line is generated for that newly entered person.

### **3.5 Blaise 5**

Blaise 5 is used for the instrument (see figure 1 in middle right in tan). Prior versions of the HRS were programmed in Blaise 4.8. Given the mixed mode nature of the 2018 design, the instrument was upgraded to Blaise 5. There is one data model for both the web content and phone content. Most of the survey content is the same across modes, but there are questions/content that are mode specific.

## **4. Data Collection**

Data collection began May 30, 2018 and goes through the first quarter of 2019. The instrument is lengthy. On average, it takes about two hours to complete. While this may seem long, these respondents are members of a longitudinal study and know how long the survey takes.

Given the length, one might expect the respondents to suffer from fatigue and complete the survey in multiple sessions. This however, is not the case. About 55% of completed cases complete the survey in a single session. Over 75% of completed cases complete in two sessions or less.

Respondents are using a variety of devices to complete the survey. Over 85% complete the survey using a PC, about 11% using a tablet, and over 3% using a smartphone. It should be noted that the survey is not optimized for the smaller screen sizes of smartphones. When a respondent lands on the sign-in page, the device being used is detected and a message informing them that the survey is best viewed on a device with a larger screen. A respondent is not prohibited from continuing with the survey on the smaller screen.

Respondents also use a variety of browsers to complete the survey. Over 60% of cases complete the survey using Chrome, about 12% using Internet Explorer, about 15% using Safari, and about 12% using Firefox. It's important to know the browsers being used as Blaise 5 does not support older versions of some browsers. The browser being used to attempt the login is detected. If it's a version not supported the sign-in page informs the respondent of that. The respondent can then call-in an request assistance in downloading a newer version or ask to complete the interview via phone.

Data collection has been successful to date with over 60% of cases completing the survey via the web.

## 5. Challenges/Lesson Learned

While data collection has gone relatively smoothly we have encountered some challenges along the way. This final section details the challenges, lessons learned, and things a survey practitioner need to keep in mind when implementing a mixed mode study that includes a web component.

### 5.1 Mode Switching

On the surface this may seem like a straightforward thing to do. On the HRS we have a sequential design where cases are assigned to web first and then switch to phone. As mentioned earlier we have phone available immediately in the event of a call-in requesting a switch from web to phone, but do not actively deliver a case to be called by an interviewer until a specified time. The instrument is designed to take advantage of the mode (web or phone) of administration.

For example, for the phone interview we require responses to all questions, on the web we allow the respondent to skip (not answer) questions if they choose. This creates problems for suspended cases, especially in the switch from web to phone. Since questions in the phone interview are required, any skipped questions from web need to be re-asked over the telephone. This may lead to questions being asked out-of-context or a question that does not make any sense. For instance, the HRS has a series of questions that ask a respondent to subtract a certain number from the previous amount. If the respondent suspends during that battery of questions, they will have no idea how to answer the question since they may not remember what their previous response was during their last session.

The HRS instrument also has questions meant for the interviewer at the end of each section that ask if the respondent needed any assistance. Those questions are now on route in the phone survey and need a response. This adds to the interviewer and respondent burden trying to easily and quickly get back to the last question the respondent was presented.

To address the issue, interviewers need to be trained how to handle such situations, and/or extra coding in the data model may be needed to assign values to skipped web questions.

### 5.2 Pulling and Updating Information

In our implementation, we pull back and use information from the Blaise instrument to determine the next action in the protocol. We have had instances where a respondent inadvertently started their spouse's line. We have a process for resetting those instruments. Doing this only addresses the Blaise data, not any of the data pulled back to the management system. The data in the management system may also need to be updated if information was used from the instrument that is no longer valid.

### 5.3 Survey Access

Time needs to be spent thinking through survey access. A good survey login app is valuable. Most importantly, it allows the survey organization to control access to the web instrument. We use it to provide useful information to the respondent. As mentioned earlier, we detect which device and browser they are using to provide feedback that will help in aiding them with having the best experience. One needs to make sure they are passing in the proper information to make sure the correct display is shown to the respondent. During testing we discovered that it was possible to see the interviewer layout set during a self-administered survey. The survey login app was retaining the last layout set used (make sure you test!).

Since we have multiple respondents in a household we spent time thinking about access to the survey. The HRS has some shared preload amongst household members. That information is set in the first two sections of the instrument. If a respondent is currently in one of those two sections the spouse is blocked

until the respondent has either completed those two sections or the server timeout has occurred due to inactivity.

Even though the design is sequential (web first, then phone), we allow a phone interview as the respondent's request. The management system has built in capabilities to only deliver cases that do not have an active Blaise session. We implemented this feature for two reasons, 1) We didn't want the interviewer to bother a respondent who was working on the survey, and 2) we did not want the respondent following along in the web version inadvertently updating data since there is only one data model.

#### **5.4 Timeouts and Redirects**

Timeouts are essential for keeping data secure. Initially the Blaise (server) timeout was set to 20 minutes. It eventually was reset to 21 minutes due to an issue that occurred with additional features we created related to the timeout. We alert respondents two minutes (at 18 minutes) prior that their session was about to timeout for security reasons. A pop-up message alerted the respondent that the session was about to end. If the respondent clicked "Ok", the clock reset and they could continue. If the timeout occurred, the respondent was sent to a custom GoTo page that contained a link that redirected them to the survey login app. There timeout was happening inconsistency with our added features so we increased the Blaise (server) timeout to 21 minutes and had a project specific timeout of 20 minutes, with a warning at 18 that triggered before the server timeout happened.

# Part 8 - Adapting the Sample Management System to Blaise 5 for the Health and Retirement Study (2018)

Marsha Skoman - Survey Research Center, University of Michigan

## 1. Brief Introduction to SurveyTrak

The University of Michigan Survey Research Center (SRC) uses *SurveyTrak*, one of its sample management systems, to conduct Blaise 5 surveys in the field. SurveyTrak runs on Windows laptops and the surveys run as thick client/standalone. The Blaise Control Centre is not installed on the laptops; only Dep.exe, Manipula.exe, and a few other files are on the laptops.

The Blaise BDBX is created on the fly in the field for each case using an ASCII to Blaise Manipula script. The BDBX and the corresponding AuditTrailData database only ever contain one case. When the survey is exited, SurveyTrak zips up the BDBX and the AuditTrailData.db and inserts them into the SurveyTrak database for later processing. Then the files for that one case are deleted from the laptop. The next case is treated as a fresh, new case.

Interviewers use a feature of SurveyTrak called “send/receive” to upload survey data and to download data model files.

## 2. HRS and SurveyTrak

The Health and Retirement Study was the first study to use Blaise 5 with SurveyTrak. This turned out to be fortunate for the implementation of Blaise 5 into SurveyTrak. HRS is a complex, lengthy survey. If SRC had started with a shorter, less complex survey, the SurveyTrak development decisions would not have been ideal for the long term.

### 2.1 Data Model Size

The HRS 2016 data model, written in Blaise 4.8, is 24MB in size. The BPKG file alone for the HRS 2018 Blaise 5 data model is 896MB. The dramatic increase in size of the HRS data model causes three issues:

1. Time to download BPKG: It takes much longer for interviewers to download the HRS data model via SurveyTrak send/receive.
2. SurveyTrak limitation: SurveyTrak send/receive itself has file size limitations. The workaround for this is using a file splitting utility to split the BPKG into chunks (currently 27). (The chunks are reassembled on the laptops after send/receive.)
3. Time to unzip BPKG: The first time SurveyTrak launches a case for a particular survey, SurveyTrak unzips the BPKG. This takes several minutes.

### 2.2 Survey Launch Time

During initial development to integrate Blaise 5 into SurveyTrak, it took up to eight minutes to launch a survey. This time has been significantly reduced by using the workarounds outlined below.

Using Manipula to create the BDBX on the fly creates an incompatible BDIX file. To get past this problem SRC was instructed to use the following DEP command line parameter:

*-OverWriteDataFile:IncompatibleData*

This workaround caused another problem. When this command line parameter is used, a backup of the BMIX file is created. The HRS 2018 BMIX file is 1.47GB in size and the backing up of this large file increased launch time even more. The workaround to the workaround is to copy off the original BDIX file and use it to replace the BDIX file created by Manipula.

The 3 databases *RuntimeConfiguration.db*, *RuntimeSessionData2.db*, and *ServerManagerDatabase.db*: When launching a survey from SurveyTrak the first time, it takes a long time for these 3 databases to create themselves. The initial workaround was to, instead of deleting the databases between cases, keep those databases in place between cases.

However, later in development, this had to change. When SurveyTrak was being programmed to write data to Blaise, it was discovered that in order to write to the BDBX, the RuntimeSessionData2.db cannot exist but instead has to be allowed to create itself after the BDBX is updated.

The ultimate solution for decreasing survey launch time was to add 8GB RAM (for a total of 12GB) to the laptops and to replace the spindle drives with solid data drives.

### 2.3 Other Differences in SurveyTrak Between Blaise 4.8 and Blaise 5.3

It is typical for SRC to release updated versions of a data model during data collection. Each version is stored in a date/time folder as shown below:

C:\BLProj\HRS2018\Storage\2018-04-25,16,10,00  
C:\BLProj\HRS2018\Storage\2018-05-10,14,23,00  
C:\BLProj\HRS2018\Storage\2018-06-21,16,23,00  
C:\BLProj\HRS2018\Storage\2018-08-03,09,35,00

For Blaise 4.8 data models there is also a “Work” directory, e.g., **C:\blproj\HRS2016\Work**. Data models are copied to the work directory and the survey is launched from the Work directory. This Work directory has several advantages:

- The original data model folder stays clean and untouched.
- Easier for Interviewer Help Desk staff to troubleshoot if interviewers call in about survey problems.
- Manipula scripts used to create the BDB point to the Work directory and are created only once no matter how many data models are released during data collection.

With Blaise 5, the BMIX file takes too long to copy, so the survey is launched from the Survey directory. This requires that the Blaise programmer create Manipula scripts for each data model.

Also with Blaise 4.8, since the data model are so small, SurveyTrak is able to zip up all the data model files (BDB, BMI, etc.). This is very convenient if later troubleshooting is required.



## DATAMODEL InHrs2018Data

### FIELDPROPERTIES

Remark: Open  
IsVisited: TIsVisitedFieldProperty  
AlienActionEvent : string

ATTRIBUTES = DONTKNOW, REFUSAL

INCLUDE "HRS18SpecialAnswers.incx"  
INCLUDE "HRS18\_Type.incx"  
INCLUDE "HRS18\_SCV.incx"  
INCLUDE "HRS18\_Basis\_Tables.incx"

### FIELDS

SampID /"SAMPLE ID" : STRING[10]  
HHID /"HOUSEHOLD ID" : STRING[10]  
*{tShared}*  
Preload\_RTab : ARRAY [1..2] OF B\_RTab  
Preload\_HH : B\_HOUSEHOLD  
Preload\_Respondents : ARRAY [1..3] OF B\_People  
Preload\_Children : ARRAY [1..50] OF B\_People  
Preload\_HHMembers : ARRAY [1..20] OF B\_People  
*{tCAPI}*  
Preload\_SCV : B\_SCV  
Preload\_RVARS : B\_RVARS  
Preload\_PastPens : ARRAY [1..10] OF B\_PastPens  
Preload\_Job : ARRAY [1..10] OF B\_Job  
Preload\_Hlth\_Plan : ARRAY [1..3] OF B\_HlthPlan  
Preload\_RSiblings : ARRAY [1..20] OF B\_Siblings

### AUXFIELDS

FLJ535, FLJ005 : STRING

ENDMODEL *{InHrs2018Data}*

### INPUTFILE

MyInputFile: InHrs2018Data ('preload.asc', ASCII)

#### SETTINGS

SEPARATOR = '^'

### OUTPUTFILE

HRS18Output: HRS18 ('HRS18.bdbx', BLAISE)

#### SETTINGS

MAKENEWFILE = NO

### MANIPULATE

HRS18Output.WRITE

ENDSETUP//HRS2018\_ServerPreload

### **1.3 Considerations Regarding Survey Preload**

While both of the above processes work, there is a lot of preparation involved to prepare two different sets of preload and to keep them up to date. First, the Manipula scripts need to be recompiled each time there is a new data model, even if the preload has not changed. Second, updating preload requires a change to the caret delimited string (offline) and the creation of a new Manipula script that references only the fields that need updating (online). Both of these processes are prone to error and require substantial testing any time a change is made.

As we move toward working in one sample management system that handles self-administered surveys (online), pooled CATI surveys (online), and distributed CAPI and/or CATI surveys (offline), the preload process will likely be reduced to only having to load and maintain the server instrument.

## **2. Survey Migration**

At certain times during the data collection period, a new version of the instrument may need to be released. This could be due to an error in the original instrument, the need for an additional field, or many other reasons. The HRS calls this a survey migration or update.

To implement the change, data must be migrated from the old version of the instrument to the new version. If a respondent has already started the survey in the old version, we need to ensure that their data are retained and that they are able to resume the survey from the same question where they originally suspended. Like with preload, this process is different depending on whether we are dealing with the offline or online instrument.

### **2.1 Offline Survey Migration**

Whether offline or online, a Manipula script is needed to complete the migration process. For offline, a Blaise to Blaise script is sent to the laptops, and runs the next time the instrument is launched by the interviewer. In the case of multiple migrations, it is important that each laptop have a Manipula script that migrates from each older version to the newest version. This is needed in case for instance, a sample line is started in version 2 and is not touched again until version 5. The migration process then needs to update that instrument from version 2 to 3, 3 to 4, and finally 4 to 5 before the interview can be resumed.

### **2.2 Online Survey Migration**

For the online instrument, the same Blaise to Blaise script that is used offline can be used during the migration process. However, instead of sending the script down to the laptop and calling it upon launch, we choose a time to migrate the data on the server to the newest version. We deactivate the instrument on the server, allowing anyone who is currently taking the survey to complete but not allowing any new respondents to start, and then run the migration Manipula script. We do not need to maintain multiple versions of the script on the server side because all of the data are upgraded with each migration, regardless of whether a respondent has started or not.

### **2.3 Considerations Regarding Survey Migration**

There are some additional considerations to be taken into account when doing survey migration. Most importantly is in regards to the session database. In Blaise 4.8, we only had to complete the above steps to do a survey migration, but the introduction of the session database in Blaise 5 caused us to rethink our process. Since it is import that we allow our respondents to resume the survey from where they suspended, we had to do two things:

1. Program the instrument to save data to the main database upon survey suspend as well as survey completion.

Startup Data Entry Rules Session

**Audit Trail Level**

Keyboard

**Client Features**

Send GPS coordinates with each request

**Session Timeout**

Server Timeout

Sessions do not expire

Survey specific Timeout: 60 minutes

**Data**

Data is read-only

**Save**

On Session Timeout

On Quit

2. Delete the session database after migration so that upon resume, the session data are recreated from the main database.

Additionally, custom programming logic was added to the instrument to ensure that partial respondents did not have to have to resume the survey from the beginning and click through to their last visited field. These steps have allowed us to preserve data for respondents who have started the instrument, and to eliminate their burden of having to start over.

### 3. Survey Merge

Because of some design decisions made for the HRS, as well as the design of Blaise 5, we must merge the survey data into a single database before we analyze the data. This is similar to our process using Blaise 4.8. To accomplish this goal, we use a combination of a custom application and Manipula scripts.

#### 3.1 Offline Survey Merge

Survey data in the offline environment are stored as individual .bdbx and audit data are stored in individual SQLite databases. In order to combine data from a single interview with other interview data, the Interview Data Merge application is used. Each instrument has its own set of criteria for when to merge a case and where to put the final data. A merge Manipula script is run to combine the individual .bdbx with the master data, and the SQLite data are converted to SQL Server data, by the Interview Data Merge application, and stored in a single SQL database on the Blaise server. Below is an example of this script.

SETUP HRS2018\_Merge

SETTINGS

DESCRIPTION = 'BLAISE to BLAISE'

USES

InputMeta 'HRS18'

OutputMeta 'HRS18'

INPUTFILE InputFile1: InputMeta ('\\...\Storage\2018-08-03,09,35,00\HRS18', BLAISE)

OUTPUTFILE OutputFile1: OutputMeta ('\\...\MasterSurveyData\HRS18', BLAISE)

SETTINGS

MAKENEWFILE = NO

MANIPULATE

OutputFile1.WRITE

ENDSETUP//HRS2018\_Merge

### 3.2 Online Survey Merge

Since the online instrument is stored as one database on the server, there is no need to merge the data (i.e. it is already merged).

### 3.3 Considerations Regarding Survey Merge

Once the merge process is set up, it runs via a nightly batch process. While this requires very little overhead in the long term, it is important to note that there are many steps to set up the merge correctly. This includes having the appropriate ODBC connections to SQL Server databases, having a SQLite database reader installed, creating and compiling a Manipula script for each instrument and each migration, and making sure all the correct data locations are in place.

In the future, the survey merge will no longer be necessary as the main instrument will be on the server and the upload/download and sync processes will allow for all instrument data to be stored in one master .bdbx.

## 4. Main Data Delivery

Per request of our client, data are delivered as a single .bdbx of completed cases only. We also deliver remarks in a separate excel file. This happens every week at the start of the project and every two weeks once the project is more stable. In order to accomplish this delivery, multiple steps and Manipula scripts are required.

### 4.1 Offline Data Delivery Preparation

Due to the use of the Interview Data Merge application discussed above, little is needed to prepare the offline data for delivery to the client. The master .bdbx already contains only completed cases (criteria defined in the merge application) and has already been migrated to the most recent data model.

## 4.2 Online Data Delivery Preparation

The online data take a little more manipulation than offline for data delivery. Since all data (complete and incomplete) are stored in the server .bdbx, we must first exclude incomplete cases. To do this, we run two Manipula scripts, SIDOut and DeleteCases.

The SIDOut script reads the database and exports a text file list of all cases that are not complete.

```
DATAMODEL Subsetfields
ATTRIBUTES = DONTKNOW, REFUSAL
Type
    tComplete = (Done(1),NotDone(2))

FIELDS
    SampId    : STRING[10]
    Complete  : tComplete

ENDMODEL

UPDATEFILE IwData : MiniModel1 ('HRS18.bdbx', BLAISE)
SETTINGS
    AUTOCOPY = No

OUTPUTFILE subset : Subsetfields('SIDOUT.txt', ASCII)
SETTINGS
    SEPARATOR = '^'
    HEADERLINE = YES

MANIPULATE
IwData.READNEXT
WHILE IwData.RESULTOK DO
    IwData.COPY
    Subset.Complete := IwData.Complete

    IF IwData.SampID = 'XXXX'
    OR IwData.SampID = 'ZZZZ'
    OR IwData.Complete <> 1 THEN
        Subset.WRITE
    ENDIF

    IwData.READNEXT
ENDWHILE
IwData.Close
```

The DeleteCases script then reads that list and deletes any cases on the list that it finds in the main database (see below).

```
USES
HRS18

DATAMODEL Small
FIELDS
```

```

    SampID: STRING[10]
ENDMODEL

UPDATEFILE
BigFile: HRS18 ('HRS18', BLAISE)

INPUTFILE LookUpFile: Small('SIDOUT.txt', ASCII)
LINKFIELDS
    SampID

MANIPULATE
IF LookupFile.SEARCH(BigFile.SampId) THEN
    BigFile.DELETE

ENDIF

```

All of these steps are of course performed off of the Blaise server so as to not interfere with ongoing data collection. The resulting .bdbx is then ready to combine with the offline data.

#### 4.3 Combined Data Delivery and Considerations Regarding Data Delivery

To combine the offline and online data, we once again employ the Manipula script that runs as part of the interview data merge process. The script allows us to merge the online cases into the offline .bdbx to create one large database. A caveat is that both offline and online versions need to be on the same data model. If that is not the case (rare), the version that is older must be migrated to the newer version before the merge takes place.

Since our client requests the full .bdbx as output, in theory there is no need to go any further. However, we must also provide remarks and other field properties as separate excel files. To do this, we proceed with a basic data out Manipula script. All of the data are exported in wide format in a .txt file and a separate .fps file is produced. Finally, we read the .fps file into SAS and filter the file so that only the properties we want (Remarks and IsVisited) are included in our resulting excel files.

Due to a bug in previous versions of Manipula, this data delivery must be completed using version 5.4 even though our instrument is using 5.3.1501. Also, once both offline and online data are stored in the server .bdbx, we will only need to delete incomplete cases and process the field properties for data delivery.

## 5. Paradata Delivery

The audit data provided by Blaise are a useful source of data and provide valuable insight into the actions our interviewers and respondents take. Once processed, the audit data can be used in many ways including, but not limited to, troubleshooting/QC, instrument timings, and reports. For the HRS, we provide the raw audit data in SQL Server databases as well as parsed data that have been combined with data from other sources.

SessionId	InstrumentId	TimeStamp	Content
1 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:39:56.8460000	<StartSessionEvent Width="11372" Height="774" Device="WindowsDesktop" Language="ENG" KeyValue="0456302020" Platform="Windows" />
2 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:40:19.8460000	<GotoUriEvent Uri="C:\TechSmith\Camtasia Studio 7\CamRecorder.exe" />
3 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:40:19.8590000	<ActionEvent Action="CurrentPage()" ControlID="1" />
4 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:40:27.7200000	<UpdatePageEvent LayoutSetName="HRS_Iwver" PageIndex="1" />
5 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:40:27.7200000	<EnterFieldEvent FieldName="SecA.StartInterview.A006_" AnswerStatus="Empty" />
6 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:40:49.5620000	<KeyboardEvent KeyStrokes="1" />
7 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:01.2710000	<LeaveFieldEvent FieldName="SecA.StartInterview.A006_" Value="1" AnswerStatus="Response" />
8 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:01.2710000	<ActionEvent Action="NextField()" />
9 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:01.7870000	<UpdatePageEvent LayoutSetName="HRS_Iwver" PageIndex="1" />
10 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:01.7870000	<EnterFieldEvent FieldName="SecA.StartInterview.A007TRAlive_A" AnswerStatus="Empty" />
11 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:03.6240000	<KeyboardEvent KeyStrokes="1" />
12 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:03.8820000	<ActionEvent Action="NextField()" />
13 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:03.8820000	<LeaveFieldEvent FieldName="SecA.StartInterview.A007TRAlive_A" Value="1" AnswerStatus="Response" />
14 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:04.9510000	<EnterFieldEvent FieldName="SecA.StartInterview.A002_IwBegin" AnswerStatus="Empty" />
15 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:04.9510000	<UpdatePageEvent LayoutSetName="HRS_Iwver" PageIndex="2" />
16 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:07.5930000	<KeyboardEvent KeyStrokes="1" />
17 (000b77a6-38c-4785-b50e-947a67396232)	(099382c8f48d-4353-a95d-16175c9f2268)	2018-04-26 13:41:07.8680000	<LeaveFieldEvent FieldName="SecA.StartInterview.A002_IwBegin" Value="1" AnswerStatus="Response" />

We parse the audit data using a series of SQL stored procedures and then store the data in SQL Server databases for use by other applications. For more information about the process and output, see the paper Transforming Survey Paradata (Piskorowski, Simonson, Yoder, IBUC 2018).

SampleId	FieldName	UserFieldOrder	EnterTS	LeaveTS	FieldDuration	AnswerValue	LeaveFieldAction	LeaveFieldActionTS	SectionName	NextFieldBegTS
0456302020	SecA.StartInterview.A006_	1	2018-04-26 13:40:27.7200000	2018-04-26 13:41:01.2710000	33.5490	1	NextField()	2018-04-26 13:41:01.2710000	SecA	2018-04-26 13:41:01.7870000
0456302020	SecA.StartInterview.A007TRAlive_A	2	2018-04-26 13:41:01.7870000	2018-04-26 13:41:03.8820000	2.0940	1	NextField()	2018-04-26 13:41:03.8820000	SecA	2018-04-26 13:41:04.9510000
0456302020	SecA.StartInterview.A002_IwBegin	3	2018-04-26 13:41:04.9510000	2018-04-26 13:41:07.8680000	2.9160	1	NextField()	2018-04-26 13:41:07.8680000	SecA	2018-04-26 13:41:08.5030000
0456302020	SecA.StartInterview.A155_SelProxy	4	2018-04-26 13:41:08.5030000	2018-04-26 13:41:10.7580000	2.2540	1	NextField()	2018-04-26 13:41:10.7580000	SecA	2018-04-26 13:41:11.3920000
0456302020	SecA.StartInterview.A012_LangSwitch	5	2018-04-26 13:41:11.3920000	2018-04-26 13:41:13.0080000	1.6160	1	NextField()	2018-04-26 13:41:13.0080000	SecA	2018-04-26 13:41:13.6040000
0456302020	SecA.ContinuationInterview.A165_A013_	6	2018-04-26 13:41:13.6040000	2018-04-26 13:42:20.7940000	67.1890	1	NextField()	2018-04-26 13:42:20.7940000	SecA	2018-04-26 13:42:22.4590000
0456302020	SecA.ContinuationInterview.A013_Continue	7	2018-04-26 13:42:22.4590000	2018-04-26 13:42:30.1700000	7.7100	1	NextField()	2018-04-26 13:42:30.1700000	SecA	2018-04-26 13:42:30.7750000
0456302020	SecA.Relations.A165_A020TSameSp_A	8	2018-04-26 13:42:30.7750000	2018-04-26 13:42:41.9290000	11.1530	1	NextField()	2018-04-26 13:42:41.9300000	SecA	2018-04-26 13:42:42.8660000

The parsed audit data are then used to create timings reports for the client. We use SAS to aggregate and analyze the parsed data and to produce excel files that can be used for additional analysis or decision making.

By Section including Remarks (in minutes)	puSuspend	EventHistory	SecA	SecA2	SecB	SecC	SecD
	Avg (TEL)	0.99	0.44	3.22	4.44	1.68	13.82
Median (TEL)	0.00	0.33	2.75	3.77	1.42	12.67	10.78
Avg (FTF)	1.04	0.55	4.35	4.95	1.81	14.73	12.08
Median (FTF)	0.00	0.34	3.36	4.14	1.47	13.86	12.01
Avg (FTF-E)	0.91	1.99	3.61	4.31	1.72	13.59	11.15
Median (FTF-E)	0.00	0.55	2.92	3.65	1.41	12.59	10.81
Avg (Web)	0.71	0.41	7.45	9.10	2.94	27.92	27.49
Median (Web)	0.00	0.34	5.89	7.10	2.16	23.44	24.29

## 6. Summary

In the end, while transition of the HRS from Blaise 4.8 to Blaise 5 is still in progress, we have found many ways to adapt existing processes to the new world. It has been frustrating at times, often because of both our own decisions and the limitations of working with a new product, but we hope that the processes we put in place for the HRS can be used across other projects.



# Using Blaise 5 to Solve Multimode and Multi-Device Challenges

*Kathleen O'Reagan and Nikki Brown, Westat, United States*

## Abstract

A concurrent-mode mail and web survey was implemented in Blaise 5. The design included a mail out of a paper survey to all respondents, who were also given the alternative to use the web if they preferred. For this study, it was important to optimize the online layouts to support multi-device use. Blaise 5 enables respondents to use iOS, Android and Windows devices with varying screen sizes, browsers and other specific controls. The challenge of designing a web interface for respondents who may also be referring to accompanying hard copy when completing a survey had to be considered in the layout design for Blaise 5. Many of the questions had been designed for larger screens and tabular presentation and required adaptation. Multiple layouts had to be created to accommodate varying screen sizes and the presentation varied between the DEP, browser, tablet and smartphone. The layout issues involved color, design, images, and interactive content as well as the presentation of the information to be usable and readable. This paper discusses how the survey was managed and design changes needed to maximize the flexibility of Blaise 5 such as allowing respondents to initiate the survey on their personal devices thru the use of a link provide by text.

## Overview

This project had been in production as a paper only (PAPI) survey when it was decided to provide a WEB component, which would include the use of mobile devices. Because the respondents would be in possession of a hard copy survey while participating in the WEB survey, it was decided that the look and feel of the WEB survey should remain consistent with the paper (PAPI). This was done to the extent possible making allowances for changes due to space constraints on smaller devices and requirements for WCAG 2.0 (508 compliance). The database also had to conform to the SQL database already in existence for the paper form. This kept us from using Blaise special attributes and caused some questions to be collected as two fields instead of one. Because in paper you have no control over what data is actually captured on the form, everything in the previously collected data was allowed to be empty (unanswered) Data Cleaning rules were applied to assign missing values. In the WEB, we also did not require most questions to be answered. The client provided specifications to determine what paths to follow for skip patterns where the lead in questions were allowed to be empty. We were provided specifications that included the following:

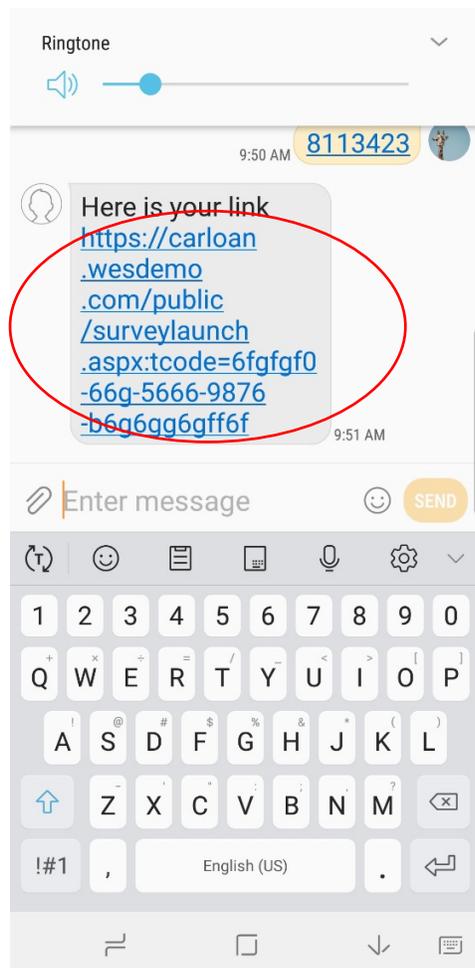
- A word.docx of the paper survey
- A file of all variables names and the allowed values for them
- An on-line flow document that contained specs for colors, fonts, skip patterns for empty values, and management screens.
- The graphic to be used in the header

## Management

### Management System

The management system, Multi-mode management system known as “M3” is a proprietary system used by Westat for management of multi-mode surveys. In the case of this survey, one of the modes, CAWI, is in Blaise 5 and the PAPI (paper) is in Teleform. The sample is loaded into M3 and the mailouts of paper surveys and the CAWI instrument are managed by it. The reminder letters are only sent out if there is no response either by mail or on-line. The CAWI instrument has multiple layout sets so for ease of login, an alternate way of login was provided for mobile devices. From a desktop or PC, the respondent just types the url and then has to key in an ID and a pin number that were sent in the cover letter with the survey. From a mobile device, they can text to a phone number a different key (also included in the cover letter) which will send them a link to the survey that handles the login for them. (Fig 1.)

Fig. 1. Login text



### Management of Testing

Three layout sets were used for the CAWI instrument so testing needed to be done on multiple platforms. The contract called for usability in the five main browsers, IE, Chrome, Firefox, Opera and Safari.

Spreadsheets were made listing devices and browsers so that testing could be checked off for each. All of this then had to be repeated in Spanish to assure that none of the layouts needed to be modified for the sometimes longer Spanish text. When all layouts have been applied, the survey is opened for 508/WCAG2.0 testing and database testing.

- Desktop Windows (layout large browser – IE, Chrome, Firefox, Opera)
- Desktop MAC (layout large browser – Safari, Chrome, Firefox, Opera Mini)
- Samsung tablet (layout medium browser – Web Explorer, Chrome, Firefox, Opera Mini)
- IPAD (layout medium browser – Safari, Chrome, Firefox, Opera)
- Samsung phone (layout small browser – Web Explorer, Chrome, Firefox, Opera Mini)
- iPhone (layout small browser – Safari, Chrome, Firefox, Opera Mini)

Safari was only tested on the IOS devices and IE was tested on the Windows and Android devices.

## Layout

### Layout Design

The master page of each layout set was designed to the client’s specifications. Due to a lack of space on phones they chose to leave the header off the small layout set. Also, the left margin on the small layout master page was decreased to reduce the wrapping of text needed to display question and answer text on the screen without scrolling.

In the hardcopy many questions were displayed in a table like layout. These stem/leaf questions fit on both PC’s and tablets but not on phones without scrolling. We applied the FlatButton table template for both the large and medium layouts sets. (Figs. 2 & 3) Note that although the Large and medium layouts set use the same layout, the text wraps differently.

Fig. 2. Large Layout Set

Car Loans

For assistance with the survey, call 1-800-555-1212

When you began the process of applying for the loan, how familiar were you (and any co-signers) with each of the following:

	Very	Somewhat	Not At All
The interest rates available at that time	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The different types of loans available	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The loan application process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The down payment that may be required	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Income requirements	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Your credit history or credit score	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Money needed for loan re-payment insurance	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Previous Next

Fig. 3. Medium Layout Set

**Car Loans**

For assistance with the survey, call 1-800-555-1212.

When you began the process of applying for the loan, how familiar were you (and any co-signers) with each of the following:

	Very	Somewhat	Not At All
The interest rates available at that time	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The different types of loans available	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The loan application process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The down payment that may be required	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Income requirements	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Your credit history or credit score	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Money needed for loan re-payment insurance	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Previous Next

SAMSUNG

For the small layout set, we asked each line of the “table” individually. Because the stem question was not included in the field text for the individual lines, we used a role. (Fig. 4.)

Fig. 4. Stem question role

```
Q4_01 ("Q4_01")
  "When you began the process of applying for the loan, how familiar were you (and any co-signers) with each of the following:"
  "Cuando usted comenzó el proceso de solicitar el préstamo, qué tan familiar era usted (y cualquier co-firmante) con cada uno de
  los siguientes:\n"
  :TContinue
GROUP Grp_Q4
  FIELDS
    Q4A_01 ("Q4a_01")
      "The interest rates available at that time"
      "Las tasas de interés disponibles en ese momento"
      StemQuestionText "When you began the process of applying for the loan, how familiar were you (and any co-signers) with each of the following:"
      "Cuando usted comenzó el proceso de solicitar el préstamo, qué tan familiar era usted (y cualquier co-firmante) con cada uno de los siguientes:\n"
      :TVerySomeNot1

    Q4B_01 ("Q4b_01")
      "The different types of loans available"
      "Los diferentes tipos de préstamos disponibles"
      StemQuestionText "<FONT Bold=FALSE>When you began the process of applying for the loan, how familiar were you (and any co-signers) with each of the following:</FONT>"
      "<FONT Bold=FALSE>Cuando usted comenzó el proceso de solicitar el préstamo, qué tan familiar era usted (y cualquier co-firmante) con cada uno de los siguientes:\n</FONT>"
      :TVerySomeNot1
```

This role was filled with the stem question for each line. For the first question in the table the role text was in the same font as the field text (Fig. 5). For the subsequent lines the role text was not bolded. This ensured that the respondent always had the stem question available if needed. (Important when many lines in a table) (Fig. 6).

Fig. 5. Bold text

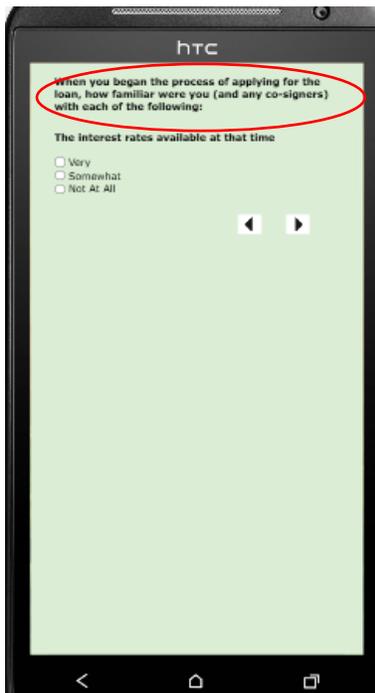
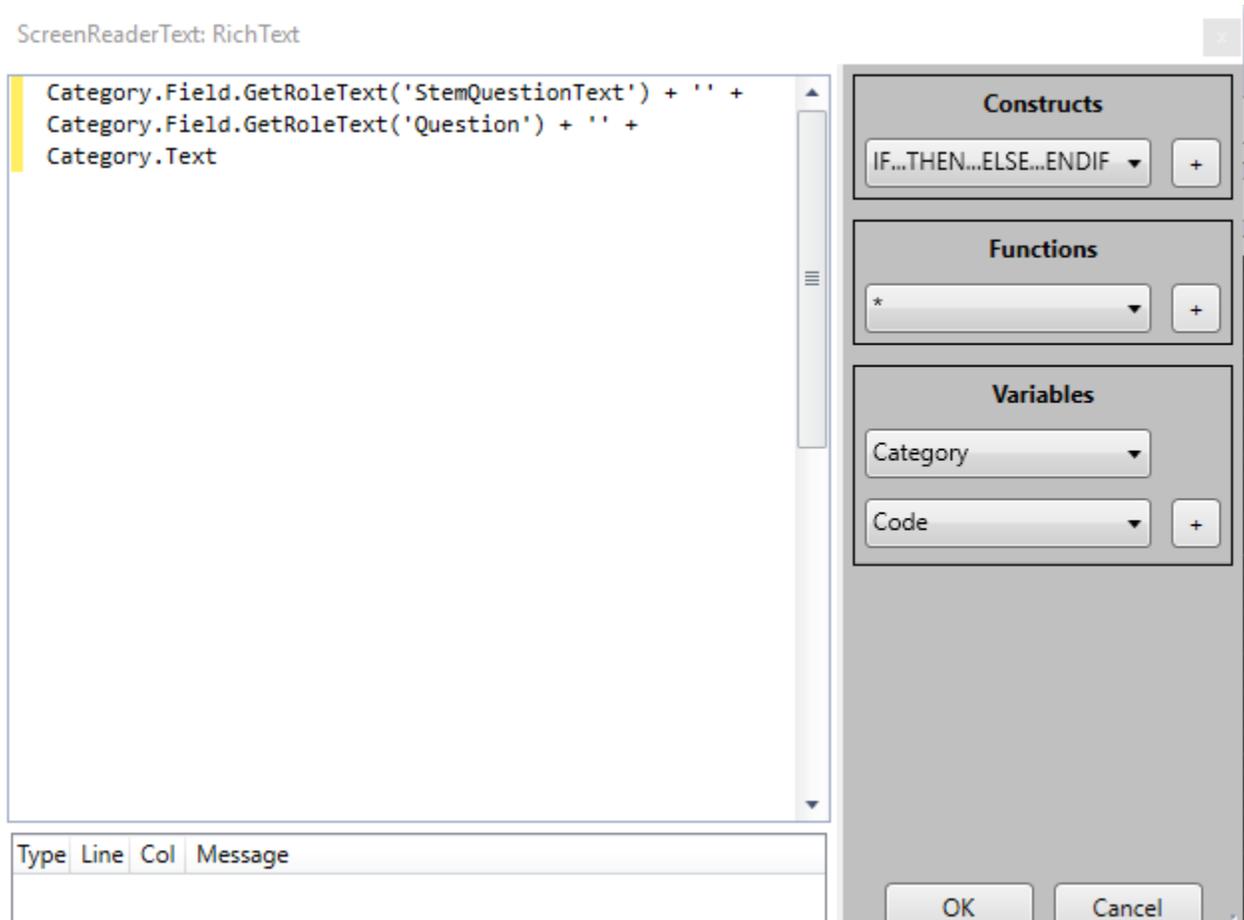


Fig. 6. Not bold text



This role text had another job. It was also used to make sure the stem question text was read by screen readers for 508/WCAG 2.0 compliance. (Fig. 7.)

Fig. 7. Expression for screen reader text



Some of the stem leaf questions had other specify fields as well. For these we just put two fields on the screen, the option Buttons table and a vertical template for the Other specify field.

In the original iteration of the WEB survey to match the database that was already in existence, any questions using a numeric field that had a check box on the form for DK had to be asked as two questions on the WEB instead of using a numeric type with a DK attribute. This meant that we also had to implement a hard edit saying that only one of the questions could be answered. Due to the more stringent requirements of 508/WCAG 2.0 we returned to collecting it as one field making use of the attributes and removing the need for an edit. However, this entails a back end modification to the database. To achieve the checkbox instead of the default radio button for the DK attribute we applied a layout other than the default for the special answer category template. (Fig 8.)

Fig. 8. Numeric with special attribute checkbox



**Car Loans**

For assistance with the survey, call 1-800-555-1212.

**Your Loan**

What was the dollar amount of your most recent car loan?  
(Please enter numbers only, no commas or spaces.)

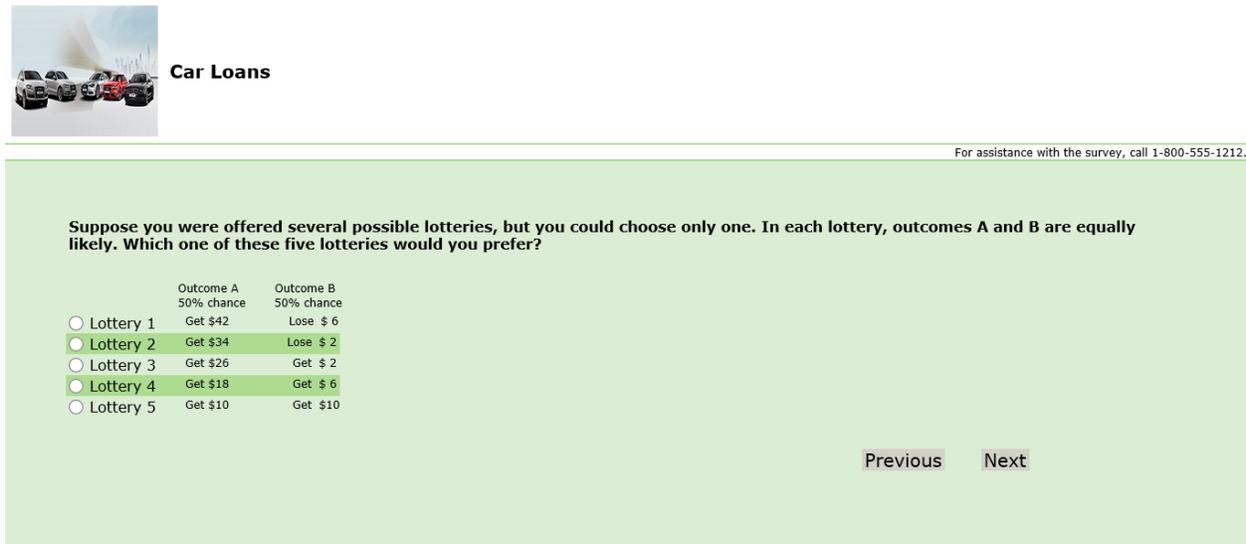
\$  .00

Don't know

Previous Next

One question in the hardcopy was displayed in a table like layout but is actually a regular enumeration type question. This layout entailed the use of roles as well as multiple slightly modified templates. (Fig. 9.)

Fig. 9. Enumeration responses appearing as a table



**Car Loans**

For assistance with the survey, call 1-800-555-1212.

Suppose you were offered several possible lotteries, but you could choose only one. In each lottery, outcomes A and B are equally likely. Which one of these five lotteries would you prefer?

	Outcome A 50% chance	Outcome B 50% chance
<input type="radio"/> Lottery 1	Get \$42	Lose \$ 6
<input checked="" type="radio"/> Lottery 2	Get \$34	Lose \$ 2
<input type="radio"/> Lottery 3	Get \$26	Get \$ 2
<input checked="" type="radio"/> Lottery 4	Get \$18	Get \$ 6
<input type="radio"/> Lottery 5	Get \$10	Get \$10

Previous Next

The Demographic section of the questionnaire included many questions that were asked of both the respondent and/or their spouse/partner. For these we used a modified version of the abreast template. (Fig. 10.)

Fig. 10. Gender with spouse/partner



**Car Loans**

For assistance with the survey, call 1-800-555-1212.

**Gender:**

<b>You</b>	<b>Spouse/ Partner</b>
<input type="radio"/> Male	<input type="radio"/> Male
<input type="radio"/> Female	<input type="radio"/> Female

[Previous](#) [Next](#)

If the respondent stated that they were never married and had no partner, the screen only showed the question for the respondent. (Fig. 11.)

Fig. 11. Gender without spouse/partner



**Car Loans**

For assistance with the survey, call 1-800-555-1212.

**Gender:**

Male

Female

[Previous](#) [Next](#)

## **Conclusion**

Blaise 5 has a robust set of COTS templates that can be used as default layouts. However, there are circumstances where customized layouts are requested. In these cases, we can modify the Blaise layouts to achieve the desired results. When making modifications you should always be aware of 508/WCAG 2.0 requirements. In addition, thorough testing is recommended to avoid unintended consequences when changing templates.



# Mobile usability on household survey on Blaise 5

*Petri Godenhjelm, Pyry Keinonen, Anna Niemelä, Statistics Finland*

This paper discusses the experiences gained from the designing and testing the Blaise 5 household questionnaire on the EU Survey on Income and Living Conditions. Statistics Finland is implementing web mode as a one mode in mixed-mode survey designs and especially mobile first principles is followed on survey design.

In 2017-2018 we have been developing mobile-layout and re-thinking ways to present grid questions in questionnaires. Our aim is to improve user experience and usability of web-questionnaires regardless of what type of device the respondent uses. Responsive web design is an ongoing trend that guides our Blaise-development heavily. One of our goals is to make a fully responsive layout for Blaise web-surveys and get rid of multiple different layouts for multiple devices.

During the design process several new design features of Blaise 5 was learned and tested. The main task done so far is the scalability of Blaise 5-layout for the different size of screens. Also we have examined ways to break down grid questions to single questions without increasing response burden and using dynamic response buttons as much as possible instead of response fields that require typing.

The usability testing was also done through cognitive interviews with the concurrent think-aloud method and screen-recording of all the test interviews. This video material and usability guidelines gave the path for design choices during the re-design work of the web questionnaire.

The most challenging part has been to adapt for the implementation of the Blaise 5 version changes. On the other hand the thorough expertise of Blaise has developed during this process. And at the same time the new processes has been developed to optimally utilize Blaise 5 in the whole organization, especially in the mixed-mode design. The solutions designed for the formation of household can also be used in other questionnaires in future.

## 1. Implementing web mode in mixed mode survey design

Mixed-mode is one big strategic goal in all social and household survey development areas at the moment in many organizations and the same is in Statistics Finland. This includes finding ways to fight the growing nonresponse rates and creating possibilities to save data collection costs and at the same time to offer better customer service for respondents. This means also big efforts to develop tools for managing mixed mode surveys.

Current situation in our social surveys is more or less multicolored and every survey has its own tailored data collection process. At the moment using CAWI means two separate processes added questionnaires with two different tools (Blaise 4 and 5). Interviewer work is managed in two different systems (field and CATI) which means challenges in organization and division of work. In addition the whole interviewing system and Blaise 4 are approaching the end of their life cycle. In addition we are still tailoring our mixed-mode data collection software system to manage mixed-mode data collection in Blaise 5 more smoothly.

## **1.1 Mobile first principle**

There is a big growth in ownership and use of smartphones for many different kind of online activity. People expects that surveys, in addition to other services, are accessible by smartphones and other mobile devices. The use of mobile devices means challenges on many ways in survey production. This has been noticed and mobile respondents are taken into account in a survey process and this is a strategic goal on designing self-administered surveys. Mobile first principle is a new way to design questionnaires and raises a range of new design issues. Also this means ongoing analysis of survey data collected on different devices to research impact on data quality.

## **1.2 Respondent first & usability**

Respondent first approach is a principle which is followed in a development of mixed-mode survey questionnaires and other aspects of survey communication with respondents. This includes rounds of usability testing of the questionnaires. Especially in mobile design there is a need to carefully consider the length of questions, response options and response guideline/help texts. The target is to make questions more clear and to design in a way that respondents can manage the response process themselves compared to interview mode there an interviewer can help the response process. In our design, the lay out also defines quite a lot how questions are asked. The mobile layout sets certain conditions to question formulation, e.g. there is no space for long help texts, and the respondent is not guided by an interviewer. In order to minimize the need for extra clarifications, the aim is to simplify the questions as explicit as possible.

A responsive web survey design means an adaptation to screen size and devices and a new look for questionnaire. It is expected that questionnaire elements appear with appropriate sized for the device on which they are viewing the page on hand. In mobile approach this often means different design solutions compared to PC and lap top. For example grid form in PC could mean drop-down grid or a series of multiple-choice questions on smartphones and tablets.

In addition our design goals are to design the questionnaires mobile and respondent friendly, functional and still ensure high-quality of questions. This has implications to survey substance and the co-operation is needed among many experts on statistics production. Substance, data collection, questionnaire design and testing, usability and IT are all included.

## **1.3 Roadmap of implementation in Statistics Finland**

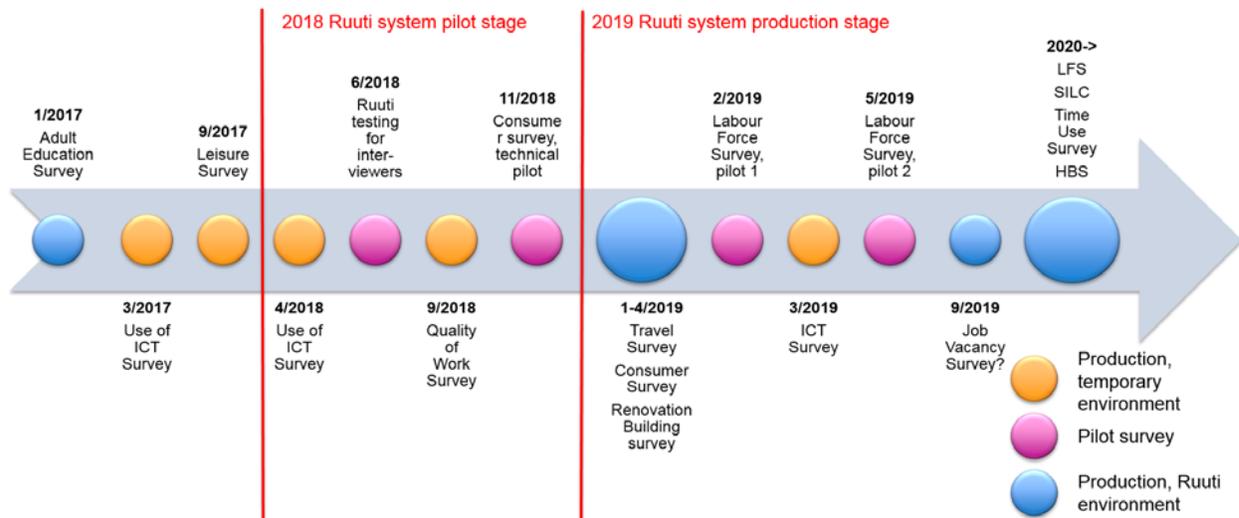
Overall goal is going to towards digital data collection. This means different measures in survey communication starting from contacting respondents and reliable identification to the response process itself with questionnaires and ending to feedback and rewarding. There are four main areas where the change management is essential. The whole data collection process is being reorganized, the new data collection system is in development phase, and also mixed mode questionnaire development and testing process is taking a new shape. And also statistical methods is developed to answer the emerging analysis and quality issues.

In mixed mode administered surveys a flexible work division between interviewers is important and the tools for effective management and monitoring surveys is needed. Developing this is also important area then moving to Blaise 5 as a whole. In a first phase in our development process field interviewers and CATI

interviewers use the same user interface in organizing their work. And later on a feasibility of Blaise CATI management will be considered.

In our road map there will be some pilot surveys and the first survey will be in a production in the new production system year 2019.

## Implementation schedule of mixed-mode data collection in social surveys 2017-2020



## 2. Mobile usability on Blaise

As a part of Statistics Finland's transition to use Blaise 5 in mixed-mode surveys, the aim is to harmonize the device-independent user experience for respondents. We currently develop Blaise 5 environment to be as responsive as possible and suitable for most common devices and browsers such as Android or IOS-based smartphones and tablets. The layout development we made during past year included not only re-designing our customized Blaise Resource Database layout (.blrd) but also re-thinking new methods to replace grid-type-questions that would not fit most of the smaller screen sizes.

We have done usability testing to the formats described below. The feedback of the layout is good. Layout work is made jointly by Blaise experts and questionnaire design experts from the cognitive laboratory of Statistics Finland. The process is iterative with cognitive and usability tests and re-design work.

### 2.1 Re-designing matrix questions and grids

Large grids and matrix questions/table format are very hard to scale for small screen sizes and the usability is weakened if the respondent have to scroll horizontally. In our solution the tables are dismantled into individual questions and the structure of the questionnaire order is changed, if necessary. The options that are in old format in the columns are placed as response options (response button) and the change on subject matter in question is demonstrated on bold fonts. This way it is possible to present grids in which the answer categories are down in response buttons and the (repetitive) question is above and the changing element in the question is highlighted.

Our design principle has been that there are no grid questions in a sense that we could present one grid per page. The grid formats have to be different because of the mobile layout format. This work has led us to redesign and rethink many parts of the questionnaires which includes grid structured questions. We have changed the question order and simplified the structure of questions to make the survey structure to work better with touch screen devices. Since there will automatically be some changes in timelines when shifting modes, we have decided to take very broad-minded approach to questionnaire changes and designing of questionnaires to make functional mobile and respondent friendly design without compromising on high-quality. This approach meant, that Blaise 4 designed questionnaires which included a grid structure couldn't be transferred into web and mobile friendly questionnaires without acknowledging there had to be done some major design changes.

## **2.2 Blaise Resource Database development**

In our Blaise Resource Database development we have taken a different kind of approach of using Resource Sets for different screen sizes or devices. In general we have only one master page-layout which is used in every Resource Set without unique settings needed per set. We have also started to use more Template Parts for maintaining the layout structure more easily.

In current solution we are interpreting client's screen width to determine which settings are used in that particular device or browser. In practice, we defined a maximum size for large resolutions and a scalable size for smaller resolutions. For example if client browser width is greater than the defined maximum value then there is extra margin around the page area which is filling the rest of the page while the actual content area is centered to the middle of the page view. And correspondingly with smaller resolutions there will be no extra-margin. The buttons will also use all available space for width with smaller resolutions. This enables the respondents to touch the answer buttons more easily with thumbs.

Our web and mobile layout is designed to allow more dynamic approach to grids. In web and mobile we have brought into use a dynamic response button (for enumeration or set of type questions). That means that we don't have traditional radio buttons in the radio button questions but response buttons and by pressing the response button the auto enter is performed and question skips to the next question. While in set of type questions we have also included check box inside the button to help the respondent see the difference between answer types. First we tested radio button and check box type questions without any visual clues about different type of questions inside the response button. That however interrupted the response "flow" when check box type of question appeared. After that we have developed visual radio button/check box inside the button and are hopeful that this will make the answering experience more convenient. This solution is being tested when writing this paper.

Our main goal is to have easily maintained layout which is also standard for every survey. If needed, this approach will ease replacing old Blaise Resource Database with updated one even on ongoing production surveys.

## **3. Household survey as a test case**

The EU Survey on Income and Living Conditions (SILC) is one of the household data collections in which the implementation of mixed-mode method is in progress. In 2017-2018, a mobile-friendly FI-

SILC questionnaire was designed (Blaise 5.2.5 and 5.3.0). Development of a solution to mobile household formation was part of questionnaire design.

In Finland SILC income and wealth data is possible to gather from the administrative registers. This means that the questionnaire is somewhat lighter compared to countries where this is not possible. Also our policy on statistics gives us a permission to use prefilled questions which make a response process smoother.

In this test case we are presenting our results by demonstrating mobile friendly household survey.

### **3.1 Basic concept for household formation**

Instead of a grid layout to collect household data from respondents, the household grid has been turned into a question by question proceeding in which the concept of household is delivered to respondent, and the respondent can add/remove household members. With this method it is possible to add new or remove existing or modify existing members to the household. For example when adding a new member to the household multiple question sets are asked in a loop-formation regarding the member added.

In our model collecting basic information such as name or birthday is defined as one set of questions (block) and add/remove/modify questions sets as their own blocks. These blocks are asked inside the loop-formation. After confirming added or removed member from the household a confirmation page is presented and respondent can check the member information and decide whether to continue the survey or edit the member data. In addition, in Finland some postal address based register data can be prefilled to the questionnaire to help the respondent. For example the persons living in the same postal address are prefilled so if there are no changes needed the respondent can accept prefilled information as her/his household as it is and move on to the next part of the questionnaire.

We have programmed the household formation to be easily implemented to other surveys need. The possible data needed from the household part to the other parts of the survey is passed via parameters to ensure independent programming between different sections of the questionnaire.

## **4. Conclusions and future plans**

The development is still a work in progress in Statistics Finland and we are constantly improving our layout and questionnaire structure for better usability. The results of usability tests indicate that the designed solution of grid questions and household formation on mobile devices works, and are promising and showing the way for future development. In addition, dynamic response buttons make mobile responding pleasant and fast.

We have discovered some layout and usability problems to tackle on. For example there are still ways to improve the respondent flow such as slowing down page change after auto-enter movement from question to next question. When respondent moves to the next question without clear visual implication it can disturb the flow. Especially this problem is showing on group of questions where the question text is same for all questions while actual question text changes according to the question variable. We are currently going around this problem by bolding the actual question text but some additional design solutions has to be invented to ensure better response quality. On the long run this matter could be solved

for example by slowing down the auto-enter command so the activated button would first activate (change color) and after that the page would change. This would require work from Blaise team in CBS.

In future we as survey practitioners need to ensure that a survey instrument meets respondents' expectations for how an electronic survey should operate on different devices. With Blaise we are confident that this will happen, while at the same time we are aware that the operating environment will change very quickly.

## Surveys with sensors. The future of data collection?

*Ole Mussmann (CBS) and Barry Schouten (CBS and Utrecht University)*

General population official surveys are subject to nonresponse and to measurement error. This has been the case since the early days of surveys and, in time, methodologists have explored and implemented several measures to effectively battle these survey errors. Official surveys have relatively high response rates, despite the response burden, and questionnaire design often involves various rounds of testing and revision. Nonetheless, response rates have been decreasing universally and measurement of certain survey topics remains hard. The decline in response rates has driven survey designers to use multiple modes and to adapt the data collection strategy to sample units. Within the online survey mode, the diversity of devices has grown; especially, smartphones have become omnipresent and are used intensively and frequently. Survey topics that require in-depth knowledge (like travel locations or health), that are cognitively burdensome (like expenditure or time use diaries) and/or that are simply hard to translate to questions (like physical condition or happiness) do not lend themselves for questionnaires. The emergence of sensors in mobile devices and wearables brought promising alternatives to questionnaires. All in all, there seem to be new chances and new possibilities. But will the general population agree, as the devices also bring new privacy issues?

### Topics for sensor surveys

In 2016, Statistics Netherlands and Utrecht University established a joint research program on ICT innovation in primary data collection under the name WIN (short for Waarneem-Innovatie Netwerk).

Simultaneously, Statistics Netherlands organized a number of internal brainstorms to identify surveys at risk and/or surveys containing topics that satisfy one or more of the problematic measurement properties. From these the following topics were identified: travel/mobility, ICT use, budget expenditure, time use, health, living/housing conditions and working conditions. These were later pooled to four WIN subprojects: 1) travel, 2) time and media use, 3) budget expenditure and 4) health and life style.

For each of the four subprojects, research is on-going but in different stages. Statistics Netherlands has developed an open-source cross-platform travel app (the CBS verplaatsingen app) that can be used for different surveys. It uses GPS-GSM-Wi-Fi and motion sensors to detect stops and travel, which can be used in different surveys. It is developed using Xamarin in order to be able to link to the Blaise app. In November a large-scale field test is planned exploring a variety of incentive strategies and stop detection rules. The app performs active sensor data, i.e. asks respondents for checks and additional information on purposes of the travels. The same app serves as a starting point for an online event-based time use diary that inserts geo-location data. This app will most likely be further developed within a joint research project with the Dutch Social and Cultural Planning Office and in collaboration with a Eurostat taskforce on the HETUS (Harmonized European Time Use Survey). For budget expenditure another app is explored that can scan receipts, can assist respondents using geo-tagging shopping and other spending area locations that they have visited and asks for consent to link to big data sources. This project is submitted as a proposal within a Eurostat call. Health and life style are the most diverse of the four topics and most likely demand for supplementary wearables such as activity trackers or smartwatches. Here, a first case study has been initiated to explore the fitness of a combination of smartphones and wearables for measuring type of activity and calorie usage. This research is done with several other institutes and is oriented at determining accuracy, cost and user-friendliness trade-offs of different options.

These explorations and research projects are not at all unique, not even at National Statistical Offices. Sessions are frequently included in the program of large survey methodology conferences and in 2019 a dedicated workshop, MASS, will take place at Mannheim university. However, to date, experience about the implementation in official surveys is still very thin. Also, apps are rarely developed open source, having in mind that multiple users may work on the code and benefit.

## Consent to provide sensordata

A obvious potential obstacle is privacy. What about person’s willingness to provide mobile device sensordata? In order to explore, the consent to sensordata requests, two studies have been conducted within WIN and a third is currently being prepared.

The first study was fielded in the LISS-panel of CentERdata in 2017 and asked for consent to five sensor measurements. Consent rates (taken from Struminskaya et al 2018a) are shown in figure 1.

Figure 1: Consent rates to five sensordata measurements: sharing location, making a photo of one’s house, making a video of surroundings of one’s house, making a selfie photo and wearing a fit bracelet.



The second study was fielded at Statistics Netherlands with the help of the Blaise team and combined consent questions with browser-initiated sensor measurements in a Blaise 5 questionnaire. Whenever a respondent gave consent he/she was asked to also provide the sensor data. The theme of the survey was general (Struminskaya et al 2018b). The sample consisted of former respondents to a range of surveys. The consent rate to location data was 67%, whereas the consent rates to taking photos and short movies was much lower and hovered around 15%.

A possible reason for the relatively low consent rates to the camera measurements is the artificial nature of the general survey. For this reason, a substantive survey on housing conditions will be fielded in which sensor measurements are more logical and remove much of the respondent effort.

## Surveys with sensors?

Summarizing, a new type of survey seems to emerge, one that combines questions with measurements. It is yet unclear how far respondents are willing to go and what are effective data collection strategies, but it seems inevitable that they will be implemented for at least part of the traditional survey topics.

What more can be expected? Experience sampling where respondents are asked to provide their emotions and/or satisfaction depending on decision rules employing sensor data. New incentives to respondents in the form of personal feedback. Multi-person apps that communicate whenever persons are close to each other. Sensors that react to external stimuli such as NFC and Bluetooth. Sensor measurements that help linkage to big data, such as supermarket scanner or loyalty card data. These are just a few examples; varying in utility and practical feasibility, but they all hold in common that they are technically feasible.



# Transforming Survey Paradata

*Laura Yoder, Andrew Piskorowski, and Mark Simonson  
University of Michigan, Survey Research Center*

## 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.

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 valuable in understanding interviewer behavior. With Blaise 5, we have been able to widen the collection of paradata to include 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 the results of a utility to automatically parse these sources of paradata from Blaise 4 and Blaise 5 into usable tables for analysis, reporting and quality control. The data from each version can be stored together and used in conjunction with other systems like time keeping, expense, survey data, and sample management. This paper will identify and demonstrate:

- Parsing the Blaise audit data (from 4.8 or 5.x versions) into relation tables (or CSV files)
- Examples of how to use these data for quality control, reporting purposes and as a backup copy of survey results
- Calculating useful paradata measures from these data (e.g., time on/time between page, last question seen/answered)
- Aggregating the data at various levels (e.g., page-level, session-level, respondent-level)

In addition, reporting tools such as SSRS (SQL Server Reporting Services), Excel, and Power BI 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 or analysis tools. The transformation techniques and standard paradata, reporting can be implemented by any user of Blaise 5 paradata to enhance the use of these data.

## 2. Introduction

The UM Survey Research Center and other groups have previously written a number of papers and presentations about Blaise audit information and survey paradata more broadly. The focus of this paper will be a new tool developed by the University of Michigan Survey Research Center (UM-SRO Audit Parser +) that can parse both Blaise 4 and Blaise 5 formats and output the information into a standard relational database format that can be integrated into the broader survey system environment.

Although the format and information contained in Blaise 4 and 5 audit trail files is quite distinct, there is enough similarity that we have been able to design a set of tables that can capture information from both systems. In general (but not always), Blaise 4 offers a subset of information available in Blaise 5. Therefore, some fields (columns) in the paradata tables may be empty for a Blaise 4 project. One major difference is the

introduction of “Page” events to Blaise 5 - a page can have one or more fields displayed and has its own set of events. For Blaise 4 projects, there will be no Page level information.

Based on this information, the basic structure of our paradata information are as follows, with one to many relationships (parent to child) moving left to right:

**Blaise 5 structure**

Case level (CaseSummary table) → Audit Session(s) (ADTSession) → Audit Pages (ADTPage) → Audit Fields (ADTField)

**Blaise 4 structure**

Case level (CaseSummary table) → Audit Session(s) (ADTSession) → Audit Fields (ADTField)

**3. Blaise Paradata Native Formats - a brief overview**

Blaise 4 stores audit information in a delimited text file and Blaise 5 uses SQLite or SQL Server format. Due to this difference, the parser program needs to be flexible enough to read and write to a variety of formats. In addition, while there is a structure to the information in the Blaise audit files, effective parsing requires code that can easily manipulate and extract string information.

Below are small snippets of raw audit data used to provide context make apparent why a parser application is needed.

**Table 1. Blaise 4 audit data**

"5/2/2018 12:31:49:982 PM", "Enter Form:1", "Key:PCT1011	"
"5/2/2018 12:31:49:982 PM", "Metafile name:C:\BlProj\BFY_PROD\InstrumentMain\Storage\2018-04-12,14,00,00\BFY_Bas	
"5/2/2018 12:31:49:982 PM", "Metafile timestamp:Thursday, April 12, 2018 1:59:46 PM"	
"5/2/2018 12:31:49:982 PM", "WinUserName:tumsml1"	
"5/2/2018 12:31:49:983 PM", "DictionaryVersionInfo:0.0.0.0"	
"5/2/2018 12:31:50:008 PM", "Enter Field:VerifyR", "Status:Normal", "Value:"	
"5/2/2018 12:31:52:585 PM", "(KEY:)l[ENTR]"	
"5/2/2018 12:31:52:701 PM", "Action:Store Field Data", "Field:VerifyR"	
"5/2/2018 12:31:52:702 PM", "Leave Field:VerifyR", "Cause:Next Field", "Status:Normal", "Value:1"	
"5/2/2018 12:31:52:840 PM", "Enter Field:VolStatement", "Status:Normal", "Value:"	
"5/2/2018 12:31:52:966 PM", "(KEY:)l[ENTR]"	
"5/2/2018 12:31:53:027 PM", "Action:Store Field Data", "Field:VolStatement"	
"5/2/2018 12:31:53:028 PM", "Leave Field:VolStatement", "Cause:Next Field", "Status:Normal", "Value:1"	
"5/2/2018 12:31:53:052 PM", "Enter Field:xRecordIwConsent", "Status:Normal", "Value:"	
"5/2/2018 12:31:53:293 PM", "(KEY:)l[ENTR]"	
"5/2/2018 12:31:53:408 PM", "Action:Store Field Data", "Field:xRecordIwConsent"	
"5/2/2018 12:31:53:411 PM", "Leave Field:xRecordIwConsent", "Cause:Next Field", "Status:Normal", "Value:1"	
"5/2/2018 12:31:53:502 PM", "Enter Field:Section_B.CIIntro", "Status:Normal", "Value:"	
"5/2/2018 12:31:53:665 PM", "(KEY:)l[ENTR]"	
"5/2/2018 12:31:53:761 PM", "Action:Store Field Data", "Field:Section_B.CIIntro"	
"5/2/2018 12:31:53:764 PM", "Leave Field:Section_B.CIIntro", "Cause:Next Field", "Status:Normal", "Value:1"	
"5/2/2018 12:31:53:813 PM", "Enter Field:Section_B.SexOfChild", "Status:Normal", "Value:"	
"5/2/2018 12:31:54:042 PM", "(KEY:)l[ENTR]"	
"5/2/2018 12:31:54:123 PM", "Action:Store Field Data", "Field:Section_B.SexOfChild"	
"5/2/2018 12:31:54:124 PM", "Leave Field:Section_B.SexOfChild", "Cause:Next Field", "Status:Normal", "Value:1"	
"5/2/2018 12:31:54:162 PM", "Enter Field:Section_B.ChildNameF", "Status:Normal", "Value:"	
"5/2/2018 12:31:54:391 PM", "(KEY:)l[ENTR]"	
"5/2/2018 12:31:54:536 PM", "Action:Store Field Data", "Field:Section_B.ChildNameF"	
"5/2/2018 12:31:54:538 PM", "Leave Field:Section_B.ChildNameF", "Cause:Next Field", "Status:Normal", "Value:1"	
"5/2/2018 12:31:54:561 PM", "Enter Field:Section_B.ChildNameM", "Status:Normal", "Value:"	
"5/2/2018 12:31:54:751 PM", "(KEY:)l[ENTR]"	
"5/2/2018 12:31:54:867 PM", "Action:Store Field Data", "Field:Section_B.ChildNameM"	

**Table 2. Blaise 5 audit data**

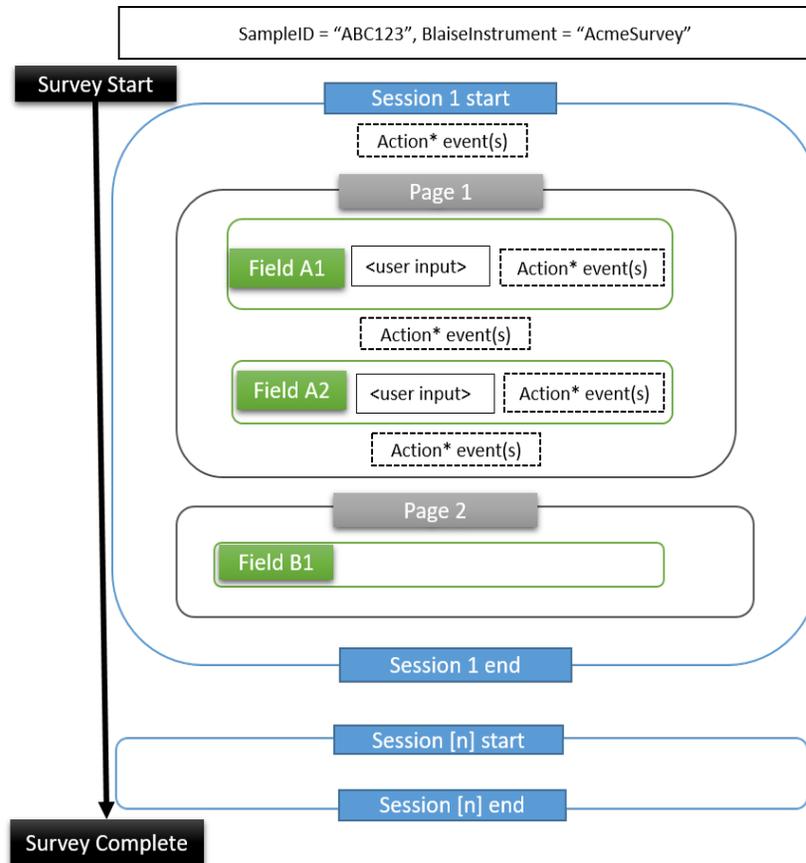
timestamp	tsadjusted	eventorder	content
04/20/2018 15:04:27.863	4/20/2018 15:04:27	60	<LeaveFieldEvent FieldName="SecA.ContinuuInterview.A013_Continue" Value="1" AnswerStatus="Response" />
04/20/2018 15:04:27.863	4/20/2018 15:04:27	70	<ActionEvent Action="NextField0" />
04/20/2018 15:04:28.888	4/20/2018 15:04:28	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="9" />
04/20/2018 15:04:28.888	4/20/2018 15:04:28	20	<EnterFieldEvent FieldName="SecA.Relations.A167_A028_RlnNHome" AnswerStatus="Empty" />
04/20/2018 15:04:49.971	4/20/2018 15:04:49	40	<KeyboardEvent KeyStrokes="1" />
04/20/2018 15:04:50.435	4/20/2018 15:04:50	60	<LeaveFieldEvent FieldName="SecA.Relations.A167_A028_RlnNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:04:50.435	4/20/2018 15:04:50	70	<ActionEvent Action="NextField0" />
04/20/2018 15:04:51.698	4/20/2018 15:04:51	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="9" />
04/20/2018 15:04:51.698	4/20/2018 15:04:51	20	<EnterFieldEvent FieldName="SecA.Relations.A167_A028_RlnNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:05:05.544	4/20/2018 15:05:05	30	<ToggleVisibilityEvent ControlName="ua_3ag" />
04/20/2018 15:05:08.720	4/20/2018 15:05:08	30	<ToggleVisibilityEvent ControlName="ua_3ab" />
04/20/2018 15:05:40.246	4/20/2018 15:05:40	30	<ToggleVisibilityEvent ControlName="ua_3ab" />
04/20/2018 15:05:42.603	4/20/2018 15:05:42	60	<LeaveFieldEvent FieldName="SecA.Relations.A167_A028_RlnNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:05:42.603	4/20/2018 15:05:42	70	<ActionEvent Action="NextField0" />
04/20/2018 15:05:43.619	4/20/2018 15:05:43	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="9" />
04/20/2018 15:05:43.619	4/20/2018 15:05:43	20	<EnterFieldEvent FieldName="SecA.Relations.A167_A028_RlnNHome" Value="1" AnswerStatus="Response" />
04/20/2018 15:05:50.644	4/20/2018 15:05:50	40	<KeyboardEvent KeyStrokes="[BACK]5" />
04/20/2018 15:05:53.008	4/20/2018 15:05:53	60	<LeaveFieldEvent FieldName="SecA.Relations.A167_A028_RlnNHome" Value="5" AnswerStatus="Response" />
04/20/2018 15:05:53.008	4/20/2018 15:05:53	70	<ActionEvent Action="NextField0" />
04/20/2018 15:05:53.734	4/20/2018 15:05:53	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="12" />
04/20/2018 15:05:53.734	4/20/2018 15:05:53	20	<EnterFieldEvent FieldName="SecA.Relations.A026_Rmarried" AnswerStatus="Empty" />
04/20/2018 15:06:10.000	4/20/2018 15:06:10	40	<KeyboardEvent KeyStrokes="5" />
04/20/2018 15:06:10.488	4/20/2018 15:06:10	60	<LeaveFieldEvent FieldName="SecA.Relations.A026_Rmarried" Value="5" AnswerStatus="Response" />
04/20/2018 15:06:10.488	4/20/2018 15:06:10	70	<ActionEvent Action="NextField0" />
04/20/2018 15:06:11.059	4/20/2018 15:06:11	10	<UpdatePageEvent LayoutSetName="HRS_Iwer" PageIndex="12" />
04/20/2018 15:06:11.059	4/20/2018 15:06:11	20	<EnterFieldEvent FieldName="SecA.Relations.A027_Rpartnerd" AnswerStatus="Empty" />
04/20/2018 15:06:15.928	4/20/2018 15:06:15	40	<KeyboardEvent KeyStrokes="5" />
04/20/2018 15:06:16.335	4/20/2018 15:06:16	60	<LeaveFieldEvent FieldName="SecA.Relations.A027_Rpartnerd" Value="5" AnswerStatus="Response" />

While the audit data contains a wealth of detail, it looks a bit messy and is not easily be used “as is”. If only the information were structured in a manner we could use for reporting and other purposes!

So, rather than go into detail about the native formats (that is what the Parser Application is for, after all), we will outline the structure and key components that are being parsed and how these relate to the output data that is stored in relational tables.

## Audit data structure conceptualized

---



One key takeaway from the figure above is that there are **one or more sessions for each respondent survey**. The audit data provides information about the session itself at the start and end of the session, such as timestamps, device used, browser type and whether the session was suspended or the survey was completed (except when the session is closed unexpectedly).

Once a session starts (assuming it is not shut down immediately), typically a page is displayed and then a field or set of fields within the page are displayed (in Blaise 4, there is no “page” event, but the field concept is similar). The audit information captures the page-level information and then the first field that is ready for input (where the input cursor resides). If the user enters some data (via keystrokes or selecting from a control like a radio button or drop-down), the input information is stored. Once a user moves to the next field or page, additional information is captured.

In the diagram, we have a reference to “**Action\* events**”. Note that Blaise 5 captures most everything as some type of “event”, and within this event you have various key-value pairs. For example: `KeyboardEvent KeyStrokes="1"` is a keyboard event with a key-value pair of `keystrokes="1"`. There also something called an “ActionEvent” - for example, `ActionEvent Action="NextField()`. The Action events are quite common and our output database has

columns to generically capture these events. But there are many other possible events (for example, GoToUriEvent, ToggleVisibilityEvent, SuppressSignalEvent, etc.). For simplicity, the diagram uses “Action\*” as a way to refer to both Action and all these other type of events that only occur in certain circumstances.

## **4. The Audit Parser Application and Output Database Structure**

To combat the proliferation of parsing tools and the resulting difficulties in consistently using the paradata information for reports and applications, UM SRC decided to build a generic parser application that can process all the different types of Blaise audit formats and output to a standard set of relational database tables. The result is that instead of focusing resources on the different parsing routines and understanding the unique set of information created by these routines, the focus would shift to understanding and querying a standard relational database created by the parser application.

### **4.1 How it works**

The UM-SRO Audit Parser application was designed to do the following:

- Understand both Blaise 4.8 and Blaise 5 audit trail formats.
- Be able to read and write to and from text based delimited files, SQL Server tables and SQL Anywhere tables.
- Allow a “manual” mode, where a user selects a case to parse and can view the results.
- Allow a “batch” mode, where the program runs on a scheduled basis and processes multiple cases.
- Do all this with a minimum number of application dependencies, so the program can run on a user’s PC or on a Server.

In addition, the application supports two basic approaches to finding and parsing audit information. One approach is to leverage Survey Management System (SMS) information about case status and use this information to find and process the Blaise audit trail data. The other approach is “self discovery” - based on an instrument id and file/database location, process all audit data that are found. In both cases, we need a mechanism to track what has already been processed and what still needs to be processed.

While the “parsing” part of the application is somewhat complex, the key to this application is really the output data model. There are trade-offs between capturing all the detail provided by the audit data and having a database that is easy to use, efficient and can be queried by various stakeholder groups, applications and reporting systems.

The resulting database model (implemented on SQL Server but generic enough for most any relational database system) mirrors the basic structure of the audit data itself. There are the four tables make up the core of the data model.

### **Study Paradata Database tables (output from Parser application)**

---

- **CaseSummary** - sample line level information with **one record for each case** for a given Blaise survey instrument (within the audit data the case id is called “key” or “KeyValue”, in CaseSummary it is “SampleId”).
- **ADTSession** - Blaise audit data session level data for each case. There is at least one session for each survey. The session has a unique number for each case which stored in the “BlaiseSession” column.
- **ADTPage** - Blaise audit data information at the “Page” level (Blaise 5 only). For Blaise 4, there will not be a record in this table.
- **ADTField** - Blaise audit data at the field level for each case. This is the most complex set of information. While some information (e.g., navigation action) falls outside the field event itself, in our model we decided to collapse everything between the “Enter Field” event and the next field’s “Enter Field” event into **one record**.

The tables are “relational”, that is, there is a **parent-child relationship** between them that allows you to use SQL code to join information into a single result set. The tables are ordered above to reflect the one-to-many relationship, from CaseSummary to ADTSession (for each case, one or more sessions), to ADTPage (for each session, one or more pages) to ADTField (for each page, one or more fields).

## Audit data structure relationship to output database tables

---

In this section, we'll discuss how raw audit information is parsed and transformed into data that are part of a relational database. The relational database is the creation of UM SRC. It is modeled on the Blaise audit data and some naming conventions are similar but, to be clear, the tables and field names used are UM SRC conventions.

The structure of the current set of tables comes from common elements found in various other audit data tables created previously at UM SRC, an analysis of the Blaise 5 audit structure, and the common reporting and use case needs we have identified.

We also note that because Blaise uses "FieldName" to reference the unique question item name, this may cause a little confusion at times. Therefore, we will reference **database "fields"** as "**columns**" or "**column names**". These table names and column names that are part of the output tables are usually referenced in the form of **tableName.columnName**.

So let's take our most potentially confusing concept, the Blaise "FieldName", and try to sort out how we store it in the database as an example. The Blaise "fieldname" value from the audit trail (for our example, let's say we have a field named "Gender") is parsed and stored in a table called ADTField. The column name where it is stored is also called "fieldname". The resulting table reference is **ADTField.FieldName** and for this specific value, "Gender", we can make the notation **ADTField.FieldName = "Gender"**.

### 4.2 General Information About the Survey

The most general information about a given case and survey are stored at the Case-level and Session-level. Here we summarize some of the key information that is available.

#### 4.2.1 Case Level

The **CaseSummary** table contains essentially **three types of information** in addition to the SampleId for the respondent:

1. Audit data processing and summary information.
2. Sample Management System (SMS) information.
3. Interviewer Quality Control (QC) information.

The parser can work **without** any SMS or Interviewer information, but we have added these for additional integration capabilities and QC reporting.

## Case-level identity information

---

**CaseSummary.SampleId:** The respondent identifier, usually named “Key” or “KeyValue” in the Audit data.

**CaseSummary.InstrumentName:** The Blaise instrument name for the survey.

**CaseSummary.InstrumentId:** The Blaise InstrumentId GUID key (Blaise 5).

## Case-level Audit data information

---

**CaseSummary.ADTisComplete:** True if the survey is complete based on the audit data information.

**CaseSummary.ADTSessions:** Number of sessions in the Audit data.

**CaseSummary.ADTCompleteDT:** Timestamp of when survey was completed based on the audit data.

**CaseSummary.ADTLanguage:** Primary language of the survey based on Audit data information.

## SMS source data

---

**CaseSummary.SMSisComplete:** True if the survey is complete based on the SMS information.

**CaseSummary.SMSSessions:** Number of sessions based on the SMS.

**CaseSummary.SMSCompleteDT:** Timestamp of when survey was completed based on the SMS.

**CaseSummary.SMSLanguage:** Primary language of the survey based on the SMS.

**Additional audit summary** information (which comes from Sessions or Fields) has also been added to the CaseSummary table to allow a quick query of only this table (no joins needed) to get often requested information.

## Audit summary information

---

**CaseSummary.ADTSurveyTime:** Total time (in minutes) for a survey based on the audit data (will equal the sum of all the session duration times).

**CaseSummary.SurveyQVisited:** Number of **unique** questions (fields) visited by the respondent (each field name is only counted once).

**CaseSummary.SurveyQAnswered:** Number of questions with an answer value provided (based on the last visit to the field).

Interviewer quality control information is also available for some projects. At UM, this information feeds into the Interviewer QC application.

## Interviewer QC information

---

**CaseSummary.SMSIwer / CaseSummary.ADTIwer:** Interviewer id from the SMS system and/or audit data (when available).

**CaseSummary.InterviewOrder:** For a given interviewer, what is the order of the interview (based on completed date and time).

**CaseSummary.MediaFiles:** Number of media files found for this interview (e.g., video recordings or audio recordings).

### 4.2.2 Session Level

The **ADTSession** table contains session level information for each case based on the Blaise audit data session records. At minimum, a survey must have at least one session. If a survey is restarted, there may be more than one session. The session has an indicator (“completed”) to indicate if the survey has been completed in the given session.

Because Blaise 5 has multimode capability, the session information also helps us determine the mode of the interview, in addition to the type of device being used, the device height and width and other information.

### Key Session level columns

---

**ADTSession.BlaiseSession:** Session number (unique for each SampleId).

**ADTSession.Completed:** True if the survey is completed in this session (last session).

**ADTSession.DeviceType:** Type of device derived from information in the session (PC, Tablet, SmartPhone).

**ADTSession.Mode:** Mode (CATI, Web, CAPI) used for session data. For a mixed mode project, sessions may have different modes. Currently, the parser derives the mode from various other fields.

**ADTSession.LayoutSetName:** Name of the page LayoutSetName (from the first page of the session).

**ADTSession.OSName:** Operating System Name (e.g., Windows, Mac, Android, etc. which is derived from the audit OS string).

**ADTSession.Browser:** Name of browser, if applicable.

**ADTSession.DisplayWidth:** User device display width .

**ADTSession.DisplayHeight:** User device display height.

**ADTSession.LastField:** Name of the last field entered in the session (parser stores last fieldname found for the session in this column).

**ADTSession.LastFieldAnswered:** Name of the last field with an answer value in the session (parser finds this information and stores it in this session column).

### 4.3 Timings

One of the most important pieces of information the audit data provides are timestamps at various levels of detail which allow us to calculate timings (durations). This timestamp information and duration information (usually in minutes) is parsed, calculated, and stored in the various tables.

At the highest level, we have Session timing information. Each session has a begin and end timestamp and we calculate the Session duration as the difference between these. If a survey only has one session, the Session duration equals the total survey time. For multiple sessions, however, the total survey time is the sum of the Session durations.

Here is a summary of **Session and Page level timing** information. There are timestamp fields available for Session and Page, but the calculated Duration time (done by the Parser application based on the timestamp fields) is usually what is needed for reporting.

#### Case, Session and Page level duration fields

---

**CaseSummary.ADTSurveyTime:** Total survey time in minutes (calculated by Parser from Session durations).

**ADTSession.SessionDuration:** Time (in minutes) spent in a given session.

**ADTSession.CumulativeDuration:** Cumulative time (in minutes) for a given session and sum of previous sessions.

**ADTPage.PageDuration:** Time (in minutes) spent on a given page. The parser calculates this using the audit timestamp associated with a given page UpdatePageEvent and then uses the **next** page UpdatePageEvent timestamp as the end time.

Note that the final session ADTSession.CumulativeDuration will equal CaseSummary.ADTSurveyTime.

**Field level timings** can get a little more complex. This section describes some different types of timing information available at the field level.

### Key timing columns at the Blaise field level

---

**ADTField.EnterTS:** Timestamp associated with the Enter Field event (when the cursor moves into the field).

**ADTField.LeaveTS:** Timestamp associated with the Leave Field event (when the cursor moves out of the field).

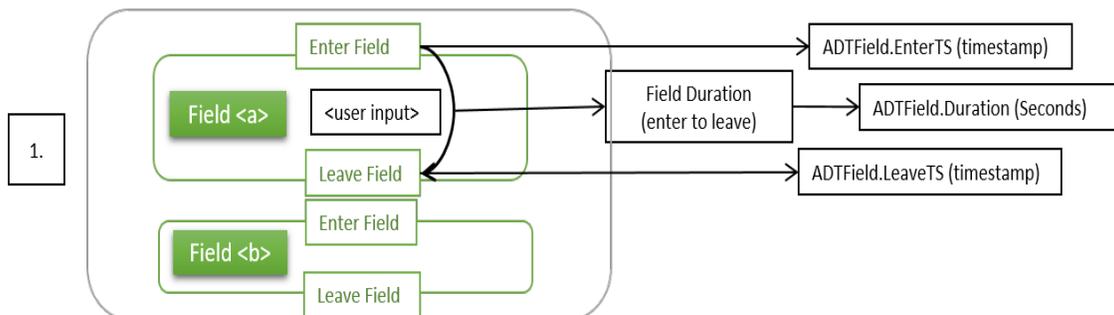
**ADTField.KeystrokeTS:** Timestamp at beginning of a keystroke entry for the field.

**ADTField.NextFieldBegTS:** The EnterTS of the **next** field (if it exists).

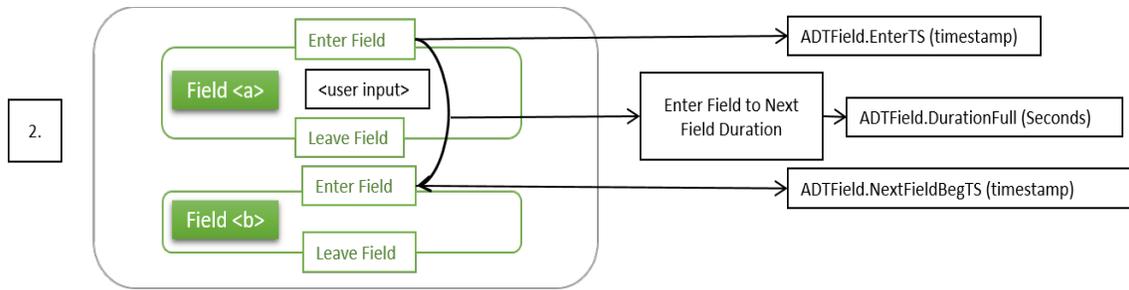
**ADTField.Duration:** Duration in seconds between Enter and Leave.

**ADTField.DurationFull:** Duration in seconds between Enter and the next Field Enter.

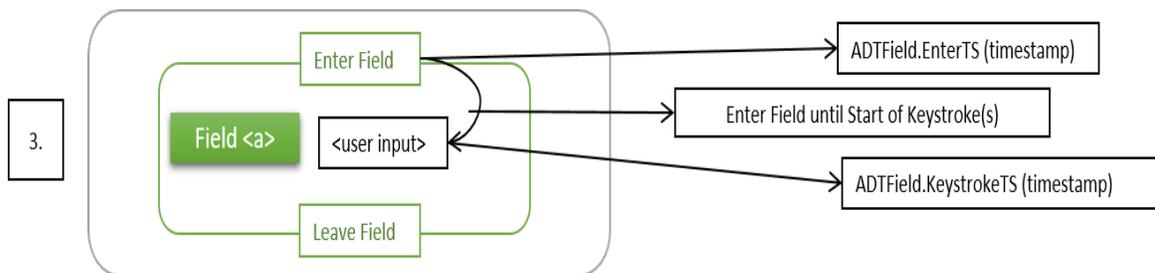
1. The duration between Enter and Leave field timestamps is a measure that only calculates the time within the field. Once the cursor leaves the field, the timing stops. Note that the sum of all these timings will likely be **less** than the total time for the page.



2. The duration between Enter event for a given field and the Enter for the **next** field captures the entire time on the field plus any events that may occur **after** the leave field event. For example, some complex conditional logic may have to execute or there can be screen displays that require a little extra time to render. We have found this is the **most consistent measure of “field” time** (as some versions of Blaise or modes do not always have accurate Leave Field times currently). Note that measure 1 should always be **less than or equal to** this measure.



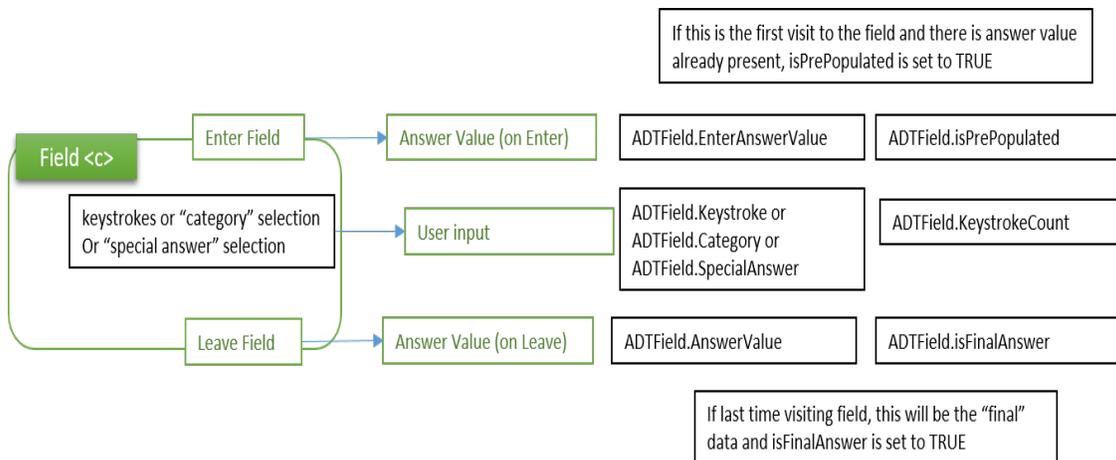
3. The time between EnterTS and KeystrokeTS is typically the time it takes to **read the item before starting to answer**. This information could be combined with length of the question text (not in the audit data) to get an idea of interviewer speaking speed or respondent reading speed (web survey).



#### 4.4 Answering the Questions

Depending on the type of question, there is typically **keystroke** information available or what Blaise terms “**category**” information (selections from a drop-down list or radio button). There are also “**special answers**” entered via special “hot” keys or menu items, such as Ctrl-D for “Don’t Know”.

In addition to whatever action is taken to indicate an answer, the audit trail also stores any answer information that is already in the field before the user edits the data. The information is stored in “ADTField.**EnterAnswerValue**” and may be there because of a “preload” data or because the question is being revisited and was previously answered.



After the user is done editing the answer, the “final” data are stored in the column ADTField.AnswerValue, unless they fall into the category of a Special Answer, in which case they are stored in the column ADTField.SpecialAnswer (as in the example above, Ctrl-D will result in ADTField.SpecialAnswer = “Don’t Know”).

Because **a field can be visited more than once**, you need to know the last time a field was edited to get the final answer that will be recorded in the main survey database. For this, we add a flag in the column ADTField.**isFinalAnswer**. By querying only the fields marked as ADTField.isFinalAnswer = True you can obtain the actual survey data that should match what is in the Blaise data file.

#### A summary of fields related to **answering a question item**

---

**ADTField.EnterAnswerValue:** Any data associated with the field **before** user input.

**ADTField.IsPrepopulated:** True if this is the first visit to the field and EnterAnswerValue is not empty.

**ADTField.Keystroke:** Keystroke values entered by the user (if applicable).

**ADTField.KeystrokeCount:** The number of keystroke values typed from ADTField.Keystroke as calculated by the Parser.

**ADTField.Category:** Category values selected (radio or drop-down type question), stored like “5” or “5-1-2”.

**ADTField.AnswerValue:** The actual answer value for the question when the user leaves the field.

**ADTField.SpecialAnswer:** For “Special Answer” type of responses (like using Ctrl-D to select “Don’t Know”), the answer value is stored here.

**ADTField.isFinalAnswer:** True if this is the **last** ADTField.AnswerValue for a given field (final data for a field). This is calculated by the Parser application.

#### 4.4.1 Additional Information About “Keystroke” and “Category” Data

The **keystroke** field captures all the keystrokes recorded by the audit data. Keystroke information may be as simple as:

```
ADTField.Keystroke = “1”
```

Or look more like:

```
ADTField.Keystroke = “R says DK because of A[BACK]memoryt[BACK]”
```

The column “**KeyStrokeCount**” sums up the number of keystrokes for this field (counting special keys like “[BACK]” as one keystroke. For the second example above, it will be:

```
ADTField.KeyStrokeCount = 31
```

The column “**AnswerValue**” is what Blaise records as the answer for this question. For the second example above:

```
ADTField.AnswerValue = “R says DK because of memory”
```

For questions that already have data (either because the respondent returned to this question or it was “preloaded”), the information is stored in “**EnterAnswerValue**”. If there is an EnterAnswerValue and the corresponding AnswerValue is different, it indicates the value was changed.

The column “**Category**” is like Keystroke, except it stores the selections made from a control. Multiple selections are separated by a hyphen (-), so selecting answers 1 and 2 looks like this:

```
ADTField.Category: 1-2
```

If the question allows **multiple responses**, the AnswerValue would look like this:

```
ADTField.AnswerValue: 1-2
```

If only a **single selection** is allowed, the last selection is what will be the answer:

```
ADTField.AnswerValue: 2
```

## 4.5 User Navigation

Capturing how the user navigates through the survey is one of the unique things available in the audit trail information. It is important to understand that there is typically both **forward and reverse navigation** possible. So, a given page (identified with a Blaise PageIndex number) and a given field (identified with a Blaise FieldName) can be visited one or more time.

### User navigation columns (all calculated or derived by the Parser application)

---

**ADTPage.UserPageOrder:** Order of pages as navigated by the user. Is unique across the entire survey.

**ADTPage.PageVisitNumber:** Visit number for same page. (If same page visited twice, 2nd time will have a value of 2).

**ADTField.UserFieldOrder:** Order of fields as navigated by the user. Is unique across the entire survey.

**ADTField.FieldVisitNumber:** Visit number for same field. (If same field visited twice, 2nd time will have a value of 2).

**ADTField.LeaveFieldNav:** A value of 1 indicates next field and 2 indicates next page, -1 indicates previous field and -2 indicates previous page.

In the database, we capture the overall order of navigation through **pages** with **ADTPage.UserPageOrder** and through **fields** with **ADTField.UserFieldOrder**. UserPageOrder and UserFieldOrder continue across sessions and are unique for a respondent. By ordering results by UserPageOrder and UserFieldOrder, the exact sequence of events for a survey can be tracked.

Note that ADTPage.PageIndex and ADTField.FieldName, however, may **not be unique**. In the tables, we have added **ADTPage.PageVisitNumber** and **ADTField.FieldVisitNumber** to indicate the visit number. Every page/field in the audit data will have a VisitNumber = 1. If the page/field is visited again, the parser increments the VisitNumber and saves another record in the database tables.

The direction of navigation after leaving a field is captured in **ADTField.LeaveFieldNav**. A positive number indicates forward navigation and a negative number indicates backward navigation. A value of 1 indicates next field and 2 indicates next page, -1 indicates previous field and -2 indicates previous page.

## Basic example of how the database tracks a user navigating through survey fields and answering questions

Using these concepts, here is a basic example. For the example, we will use two questions with Blaise FieldNames of “Gender” and “Age”. Let’s say Gender is selected from a pick list (Female = 1 and Male = 2) and Age is entered as a number.

The user will go to Gender and select the answer for Female (1), go to the Age question and enter “25”, and then return to the Gender item and change the answer to Male (2). How does this look in our Paradata field table (ADTField)?

The record in ADTField for the first visit to Gender looks something like this:

```
ADTField.UserFieldOrder = 1
ADTField.Fieldname = “Gender”
ADTField.FieldVisitNumber = 1,
ADTField.Category=“1”
ADTField.AnswerValue=“1”
ADTField.isFinalAnswer = False
ADTField.LeaveFieldNav = 1
```

The record in ADTField for the next field, Age, where the user enters an answer and then returns to the previous field (Gender) looks something like this:

```
ADTField.UserFieldOrder = 2
ADTField.Fieldname = “Age”
ADTField.FieldVisitNumber = 1,
ADTField.Keystroke=’26[Back]5”
ADTField.AnswerValue=“25”
ADTField.isFinalAnswer = True
ADTField.LeaveFieldNav = -1
```

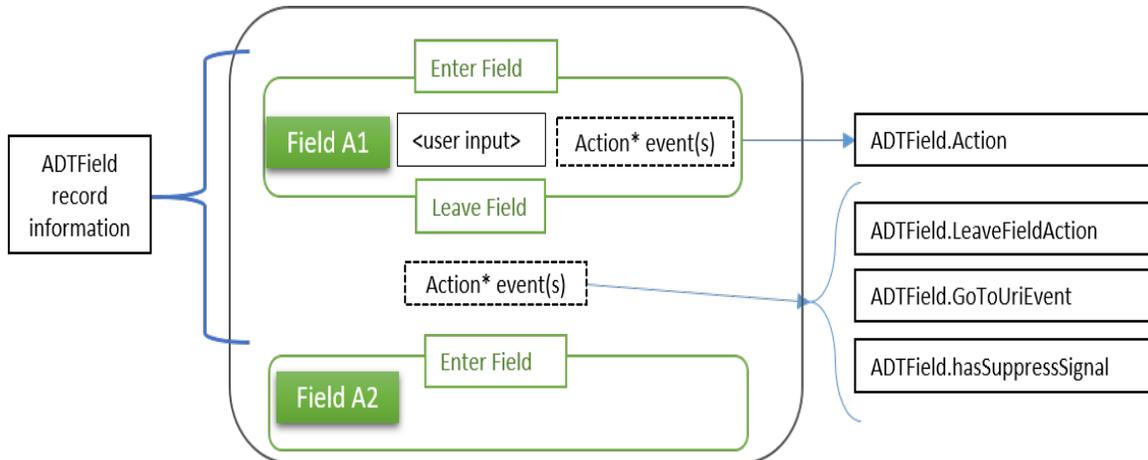
And, finally, returning to Gender for the final time, changing the answer and navigating to the next page:

```
ADTField.UserFieldOrder = 3
ADTField.Fieldname = “Gender”
ADTField.FieldVisitNumber = 2,
ADTField.InitialAnswerValue=“1”
ADTField.Category = “2”
ADTField.AnswerValue = “2”
ADTField.isFinalAnswer = True
ADTField.LeaveFieldNav = 2
```

## 4.6 Action Events Within and Between Fields

There are many different types of “actions” that Blaise records and these may or may not be of interest to users of the data. To simplify things, there are two “action” columns for each record, one for actions while within the field and the other for actions after leaving the field. There are also some special columns to store other types of events.

As mentioned earlier, a single record for a Blaise Field includes all the audit information between the field’s Enter and Leave events, as well as any events that occur before the next field Enter event. Every field visit, therefore, has one record in the ADTField table and this record contains all the information from the time the user enters the field until the time the user enters the next field (or ends the survey).



**“Action” and other Blaise events (within a field and after “Leave Field”)**

**ADTField.Action:** Name of Blaise action or other events that occur before the “Leave Field” event (e.g., AssignField({Expression State.ActiveFieldName}# 'DontKnow').

**ADTField.LeaveFieldAction:** Name of Blaise action that occurs after leaving field (e.g., “NextField()”) or other types of events that occur before entering the next field.

**ADTField.hasSuppressSignal:** True if LeaveFieldAction has a “SuppressSignal” event (e.g., “SuppressSignal(Secl.check\_1\_)”)

**ADTField.GoToUriEvent:** Launching of other programs (e.g., “C:\Blaise\Manipula.exe” or “C:\TechSmith\Camtasia Studio 7\CamRecorder.exe”).

## 4.7 Special ADTField Records: REMARKS

Remarks are stored as records in the ADTField table and are distinguished from field data using the column `ADTField.isRemark = TRUE` to filter the records. This is different from the `ADTField.hasRemark` column, which indicates a field that has an associated remark (but is not the remark itself). The fieldname for a remark starts with the pound sign (#), so a remark for Age would be named `#Age`.

The remark itself is stored in the `ADTfield.AnswerValue` column.. Therefore, another use case for this data is to easily obtain all the user remarks entered into the instrument.

### Remark related columns

---

**ADTField.FieldName:** For remarks associated with a given field, the fieldname starts with # (e.g., `#Age`).

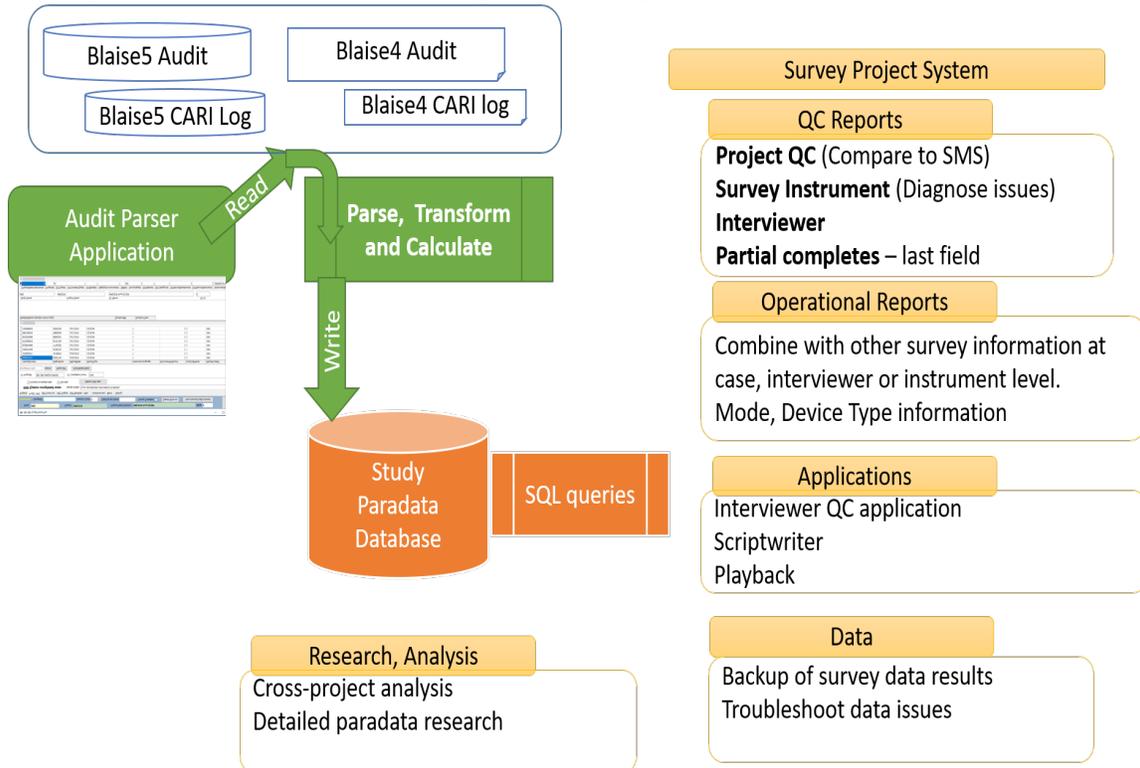
**ADTField.isRemark:** True indicates this record is a **remark** (not an answer field).

**ADTField.hasRemark:** True indicates the given field has a remark. In this example, the record for `ADTField.FieldName = "Age"` will also have `ADTField.hasRemark = True`.

## 5. Paradata As Part of a Survey Project System

With the Audit parser application converting raw audit information into tables, the Study Paradata database can be integrated as one part of the overall survey project system.

### Study Paradata database as part of larger survey environment



The image above summarizes some of the ways UM SRC is integrating the audit data into the overall survey system via the Study Paradata database. In addition to reporting, which will touch on in the next section, note that the Paradata database is also used by various **applications**. In the past, these applications built in their own unique parsing routines, but now they can leverage the Paradata database instead of having to support parsing code (this issue was most apparent when moving from Blaise 4 to Blaise 5 as the parsing code for version 4 would no longer work for version 5). Here are some other possible use case examples for this data.

#### Use case examples for Study Paradata tables

- **Backup of survey data.** A query that filters on ADTField.isFinalAnswer = True and includes ADTField.AnswerValue and ADTField.Fieldname will produce a set of data results.
- **Backup of remarks.** Interviewer remarks can be exported with a simple query for one case or the entire instrument.
- **Quality control.** Compare SMS information about survey complete status and survey completed date and time with the Audit data.

- **Intervene on potential issues** proactively. For example, query for large numbers or outliers such as number of sessions, survey time, field times or other anomalies that may indicate a problem with the instrument, internet connectivity, user issues, etc.
- Talk about “paradata” at your next social event and impress your friends.

## 6. Queries and Reporting - answering key questions

With the audit data parsed in a standard format, it allows for the development of many different types of reports. The storage of the data in relational tables (in SQL) facilitates easy access to the data from a variety of applications, including SAS, Excel, SPSS, R, and Python. For our data manager group, the advent of the parsed audit data saved us approximately 200 lines of SAS code used to create reports around survey timings.

Even without a reporting system or SQL queries, the Paradata information can be accessed using common tools like Excel Power Query and then displayed in data tables, charts, pivot tables, etc.

Note that as a **best practice**, at UM we recommend using database views and parameterized stored procedures for most end-user access to the data. This not only can further simplify access for the end user, but avoids user mistakes in creating custom query joins or filtering data, ensures that data is consistent and results are replicated no matter who uses the information.

Below are some examples of reports that use the Paradata tables. These are provided to help “bring to life” how the paradata information can be used for survey projects.

A simple, but effective report showing interview length by mode (for the same instrument).

---

### Iw Length by mode

	WEB	TEL	FTF	FTF-E
IW average	119.43	103.18	103.90	145.57
IW median	106.55	103.36	100.69	143.82

The following report about number of sessions tells us how many attempts were needed to complete the survey. For those cases that required several sessions, the data was looked at in more detail to understand why so many sessions were required (which can be due to instrument issues, connectivity issues, etc.).

---

## Sessions\* - Completes

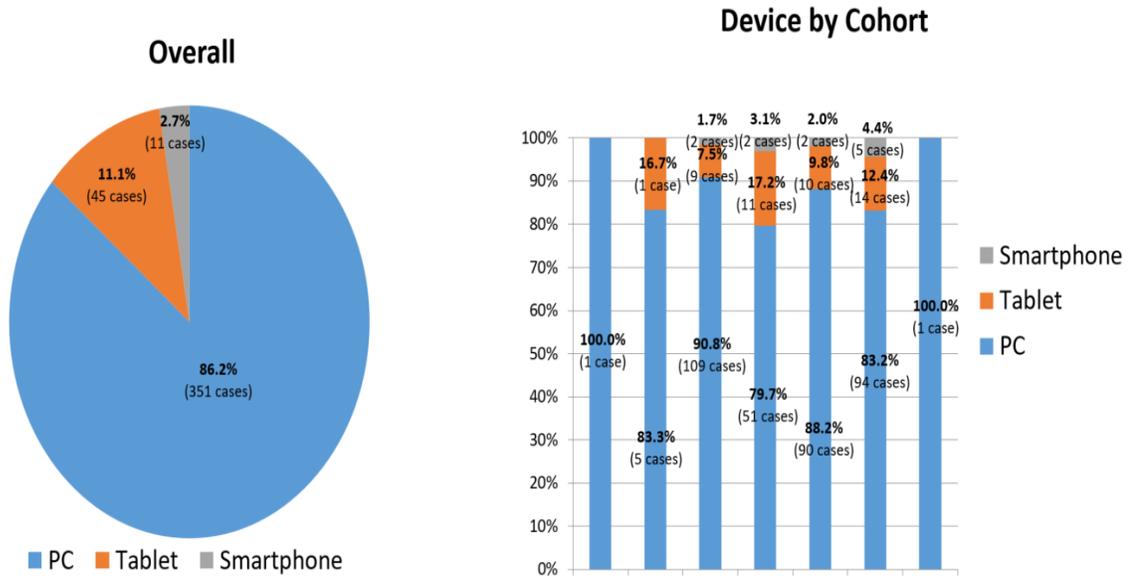
Sessions to Complete	N	%
1	173	42.5
2	114	28.0
3	63	15.5
4	18	4.4
5	17	4.2
6	6	1.5
7	4	1.0
8	5	1.2
9	3	0.7
10	1	0.2
12	2	0.5
27	1	0.2
<b>Total</b>	<b>407</b>	<b>100</b>

\*From Blaise audit data

A report about the type of devices used for a web survey combined with survey Cohort information (although it could just as well be demographics like age, gender, or any other variable of interest). This report is based on the **last Session device** used. A variation of this report would be to look at those who have multiple sessions and switched devices or comparing devices used for incomplete surveys. In this manner, we may find if respondents are having more difficulty answering the survey with certain types of devices.

---

# Devices Used\* – Completes (N=407)



An Excel table created from the Paradata database. The difference between Total Question visits and Distinct Questions tells us how many questions were **revisited**. The difference between distinct questions and answered questions tells us how many of the question fields were left unanswered.

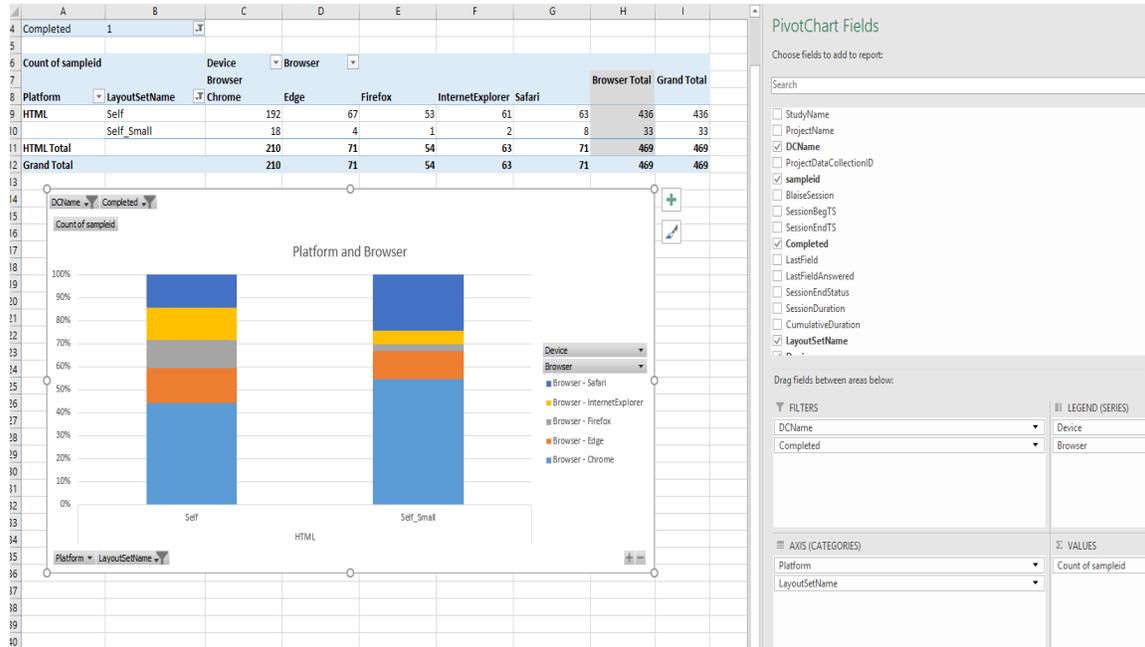
We can supplement a report like this with drill down information such as the question fields that had the highest number of revisits or highest percentage of being unanswered.

Item	Web	CATI	DCAPI
Average Survey Time	119.0	109.6	128.2
Average Total Question Visits	396.2	469.0	510.8
Average Distinct Questions	395.0	455.5	510.3
Average Answered Questions	360.5	437.0	493.3
Average Revisited Questions	1.2	13.5	0.6
Percent Revisited	0.3%	2.9%	0.1%
Average Unanswered Questions	34.6	18.5	17.0
Percent Unanswered	8.7%	4.1%	3.3%

Analyzing “no answer” distribution for each survey question in Excel. While this information can come directly from the survey data, using the audit data we can confirm how many answered or not based on only those who actually **visited** the question. And, with the Paradata database, it is operationally easy to make this part of a regular reporting process while the survey is in the field.

isFinalData	1		
Count	Column Label		
Row Labels	Answer	No Answer	Grand Total
Section_A.Q1	325	2	327
Section_A.Q1_EXP2b	330	1	331
Section_A.Q10	654	10	664
Section_A.Q11	654	10	664
Section_A.Q12	652	12	664
Section_A.Q13	652	12	664
Section_A.Q13b_Dec	460	154	614
Section_A.Q13b_Decr	365	139	504
Section_A.Q14	649	15	664
Section_A.Q14b_Dec	442	181	623
Section_A.Q14b_Decr	354	148	502
Section_A.Q15	655	8	663
Section_A.Q16	652	12	664
Section_A.Q16a_Dec	227	74	301
Section_A.Q16a_Decr	170	46	216
Section_A.Q17	645	18	663
Section_A.Q17a	538	115	653
Section_A.Q18	643	20	663
Section_A.Q18a	503	150	653
Section_A.Q19	647	17	664
Section_A.Q19a	494	158	652
Section_A.Q1a	276	48	324
Section_A.Q1a_EXP2b	272	58	330
Section_A.Q1e_EXP2b	156	2	158
Section_A.Q2	652	7	659
Section_A.Q20	642	22	664
Section_A.Q20a	484	162	646
Section_A.Q21	651	13	664
Section_A.Q21a_Dec	371	25	396

An example of an Excel pivot table and chart that can be used to look at the data in a number of different ways. Having Audit data in the database makes it easy to create pivot views of the information.



A simple Excel data table, using CaseSummary data, that can be filtered. For example, you can find the survey that has 27 sessions!

	H	J	L	M	N	O	W	X	AB	AC
1	SampleID	SurveyTime	SurveyQTotal	SurveyQVisite	SurveyQAnswer	SurveyQNonRespon	ADTisComplete	ADTLanguage	ADTCompleteDT	ADTSession
16	0103970010	218.8277	415	415	387	4	TRUE	ENG	6/25/2018 12:05	3
190	0137530010	196.9514	467	466	393	0	TRUE	ENG	6/18/2018 16:51	6
192	0138022020	183.1878	403	389	315	8	TRUE	ENG	6/13/2018 16:41	10
845	0474581040	173.474	425	423	400	2	TRUE	ENG	6/27/2018 0:11	3
437	0585891020	110.2411	349	348	326	0	TRUE	ENG	7/3/2018 10:27	2
797	1351610010	131.1268	425	422	380	1	TRUE	ENG	7/20/2018 20:05	4
007	1462340020	99.4442	342	341	321	1	TRUE	ENG	7/10/2018 5:30	2
277	5002117020	118.2784	389	387	341	1	TRUE	ENG	8/9/2018 17:26	5
406	5004360020	70.0545	434	433	403	2	TRUE	ENG	6/25/2018 19:56	2
163	5019680020	429.8955	479	466	434	0	TRUE	ENG	7/8/2018 19:13	27
369	5024660010	113.0084	376	375	348	1	TRUE	ENG	8/7/2018 17:49	3
370	5024680010	184.743	551	548	522	2	TRUE	ENG	7/23/2018 14:21	5
626	5218600011	137.3753	296	294	270	0	TRUE	ENG	7/24/2018 15:52	3
679	5224060011	104.1734	303	299	269	0	TRUE	ENG	7/5/2018 20:36	3
715	5227510010	56.5474	390	389	362	1	TRUE	ENG	7/10/2018 20:26	3
806	5238771010	91.0415	488	486	450	0	TRUE	ENG	8/18/2018 12:44	2
149	5270780010	133.087	563	559	528	1	TRUE	ENG	8/15/2018 15:39	8
311	5288990010	150.9883	353	351	324	5	TRUE	ENG	7/4/2018 12:20	4
643	5343810010	111.761	289	287	207	2	TRUE	ENG	7/26/2018 15:17	5
757	5360780010	154.0652	415	412	372	2	TRUE	ENG	8/17/2018 23:10	3

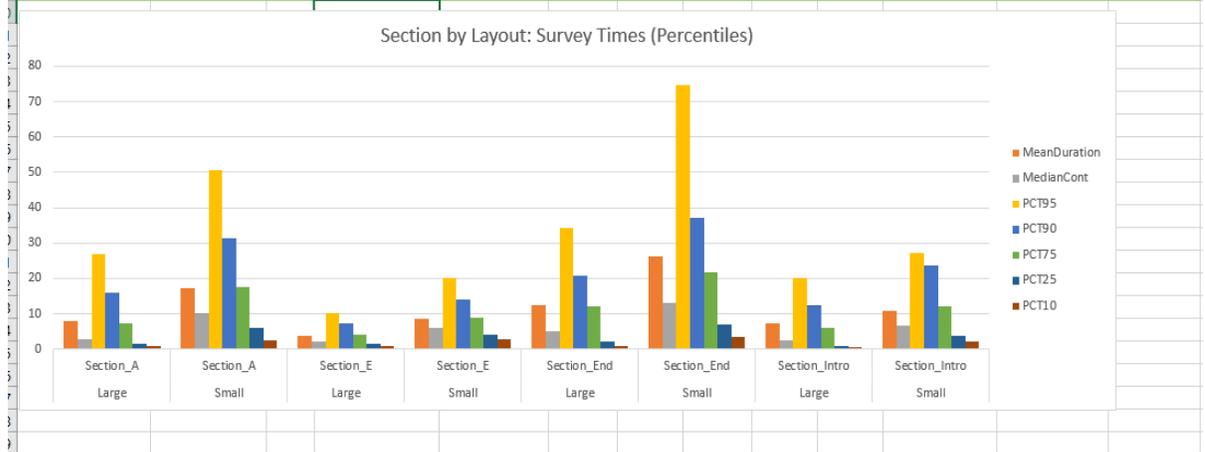
Using additional data from our SMS and combining on SampleId, we can look at interview length by interviewer experience and preferred mode of data collection.

Combined with data from sample management system

Iw_PrefMode	Cohort	Language	IwerExper	N_iwlength	Mean_iwlength	Median_iwlength	StDev_iwlength	Min_iwlength	Max_iwlength
FTF				1157	103.3127111	100.0245	31.00072803	40.5742	396.2678
FTF-E				5872	144.6993268	143.08615	38.87581184	33.2633	437.483
TEL				3006	103.0796926	103.10075	32.98908942	2.0548	313.1581
Web				704	118.0538178	105.63825	54.79536045	25.331	532.7911
FTF			New hire	363	104.3972427	101.8254	30.66333789	42.7838	222.4892
FTF			Not project experienced iwer	40	99.212265	97.70635	27.989136	60.8847	193.0174
FTF			Project experienced iwer	11	98.55875455	99.8347	26.59229221	64.2933	144.7301
FTF			Project experienced iwer last wave	743	103.0739848	99.4256	31.40009962	40.5742	396.2678
FTF-E			New hire	1706	146.5972143	145.23665	40.35017472	33.2633	437.483
FTF-E			Not project experienced iwer	159	151.0236157	144.8047	40.31134317	51.8265	303.4432
FTF-E			Project experienced iwer	38	158.1895632	159.10075	38.0422538	54.882	229.1455
FTF-E			Project experienced iwer last wave	3969	143.5010434	142.1164	38.10540098	36.5119	420.8289
TEL			New hire	1428	103.0207601	102.9225	33.2059803	2.6759	242.0938
TEL			Not project experienced iwer	177	111.3743633	108.8599	33.88610293	6.3381	245.7088
TEL			Other/unknown	38	96.54570526	92.74745	21.01411819	60.593	165.1183
TEL			Project experienced iwer	72	100.1392181	102.7332	31.12927658	4.7271	221.9102
TEL			Project experienced iwer last wave	1291	102.363972	103.1851	32.87861015	2.0548	313.1581
Web			Other/unknown	7	106.9418143	102.5983	18.61962267	86.5101	141.3257
Web			Web - no iwer	697	118.1654161	105.6486	55.03168503	25.331	532.7911

In Excel, we aggregated field level data to get timing information (including percentiles) by Section and LayoutSetName (small indicates it is for a mobile device, large is for a PC or tablet... Median is the 50th percentile).

LayoutSetName	SectionName	Cnt	MeanDuration	MedianCont	PCT95	PCT90	PCT75	PCT25	PCT10	MaxDuration	MinDuration	StdDev
Large	Section_A	31273	8.025459	2.685	26.8754	16.137	7.273	1.546	0.93	1151.909	0.002	25.35901134
Small	Section_A	2848	17.365277	10.2845	50.52005	31.2734	17.5655	6.08725	2.4133	956.84	0.002	33.91680664
Large	Section_E	7570	3.932094	2.305	10.29765	7.3137	4.18925	1.606	1.012	723.83	0.013	11.10421431
Small	Section_E	659	8.585547	5.987	20.2243	14.1896	8.906	4.073	2.9066	136.932	0.694	11.62375825
Large	Section_End	1742	12.332176	5.099	34.336	20.8937	12.17825	2.257	0.8937	1165.037	0.073	40.80959944
Small	Section_End	144	26.307958	12.953	74.6092	37.2273	21.8055	7.10275	3.464	498.41	1.987	57.60272442
Large	Section_Intro	2258	7.446145	2.404	20.1581	12.4866	5.97625	0.867	0.5584	667.002	0.126	27.88038701
Small	Section_Intro	153	10.953535	6.539	27.0712	23.607	12.269	3.693	2.2472	96.416	1.103	13.72840386





# A Different Approach to Blaise Remarks

*Peter Stegehuis, Westat*

## Introduction

Interviewer comments, or remarks in Blaise, are a useful way for field interviewers to record information that may be necessary for the data editing/cleaning stage. Remarks may be made when interviewers are not sure what to do with information provided by the respondent. Also, during long and complicated interviews it may not always be feasible to back up to the question for which the answer may need to be changed, so a comment might be the only way to relate important details.

Triaging all the Blaise remarks can be a very time-consuming and costly task. A Blaise remark is just a text string, so there is no structure or categorization possible. Reading every single remark is the only way to determine whether or not the remark contains actionable information. Prioritizing - or only dealing with - certain categories of comments before reading all remarks is virtually impossible, as categorization cannot happen without reading the remarks in the first place.

This paper shows a solution we have implemented in Blaise 4.8, using the integrated functionality of the DEP and Manipula. This approach retains the very useful elements of Blaise remarks: the interviewer can add or edit a remark at any question, and a paperclip icon shows the existence of a remark. But it adds a new element: the interviewer has to select a category before adding the comment itself. The comments get stored in their own data file, with the comment category as a separate variable. This new variable can subsequently be used as needed during the data editing stage.

## Why make remarks?

Ideally, a respondent hears an interviewer's question, remembers all relevant information instantly and answer correctly all the time. Alas, during interviews in real life however it happens frequently that additional and potential answer-altering information gets divulged well after the question has already been answered.

In some cases the interviewer cannot back up to the question for which the answer should now be changed. For instance if an interview starts with eligibility questions to determine who in the household will get the follow-up questions, it is common to disallow backing up into that section. If one of the answers in that section needs to be changed based on information that becomes available after passing that 'wall' then a Blaise remark may be the best option the interviewer has.

In other instances it may simply take too much time for the interviewer to back up multiple screens and change an answer. The risk is always there that a respondent will take the opportunity of this pause in active interviewing to end the cooperation, and with it stop the whole interview. A remark at the current question is much more easily made, with less risk for an untimely end of the interview.

A third reason for adding a remark, especially during a long interview, can be when it is hard to check what has been entered earlier in the interview.

Here is a screenshot of a standard Blaise remark popup:

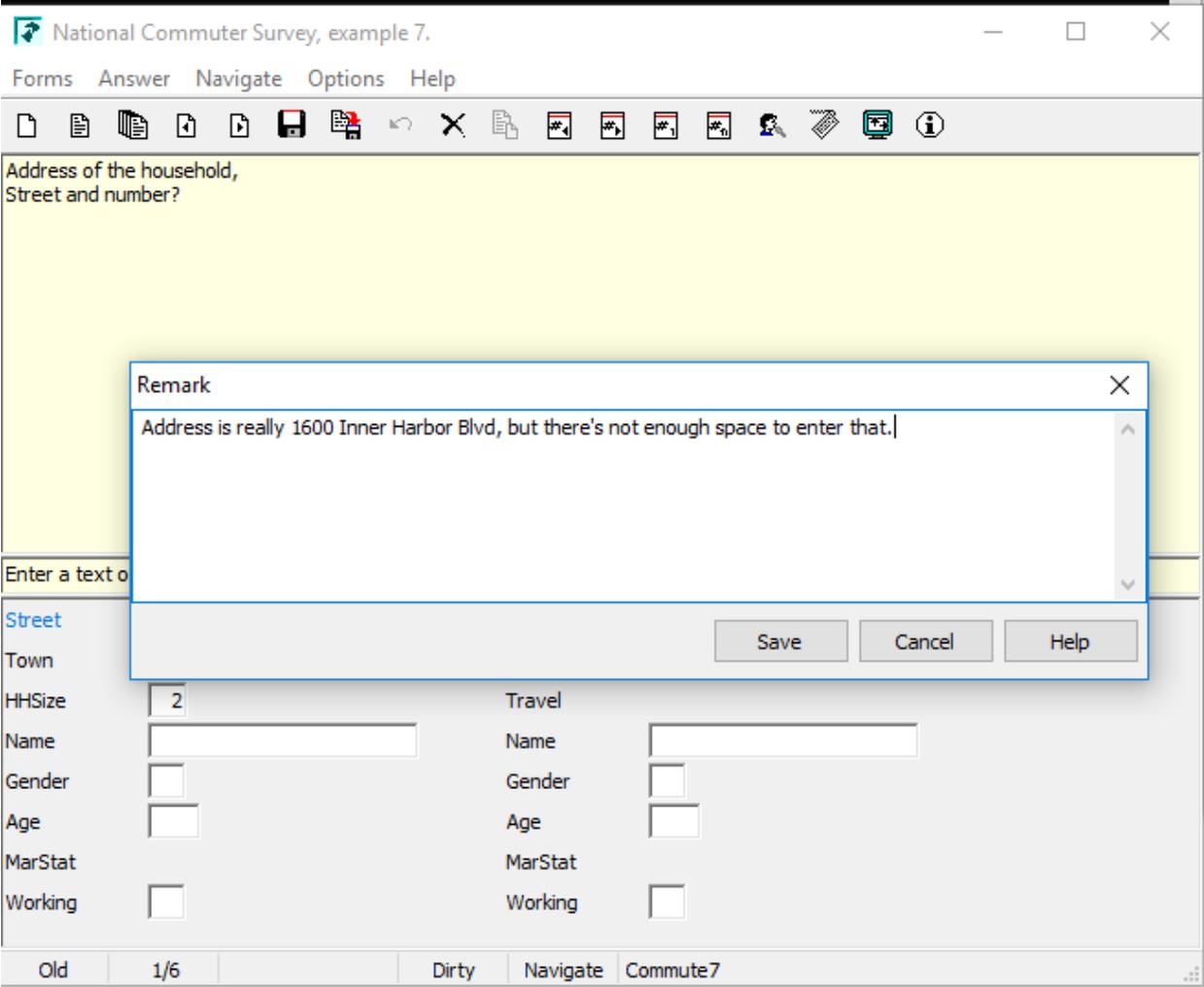


Figure 1. An example of a remark in the Data Entry Program in Blaise 4.8

### What is good about remarks?

Remarks can help improve data quality in the phase the data will go through after interviewing, often called the data editing or the data cleaning process. If a remark contains enough specific information the intended data change can be made by the home office staff. Otherwise one or more answers may have to be set to DontKnow, or the remark may have to be ignored.

Blaise has a simple and effective way of showing an interviewer where remarks were made during the interview, with the paperclip icon as shown in Figure 2.

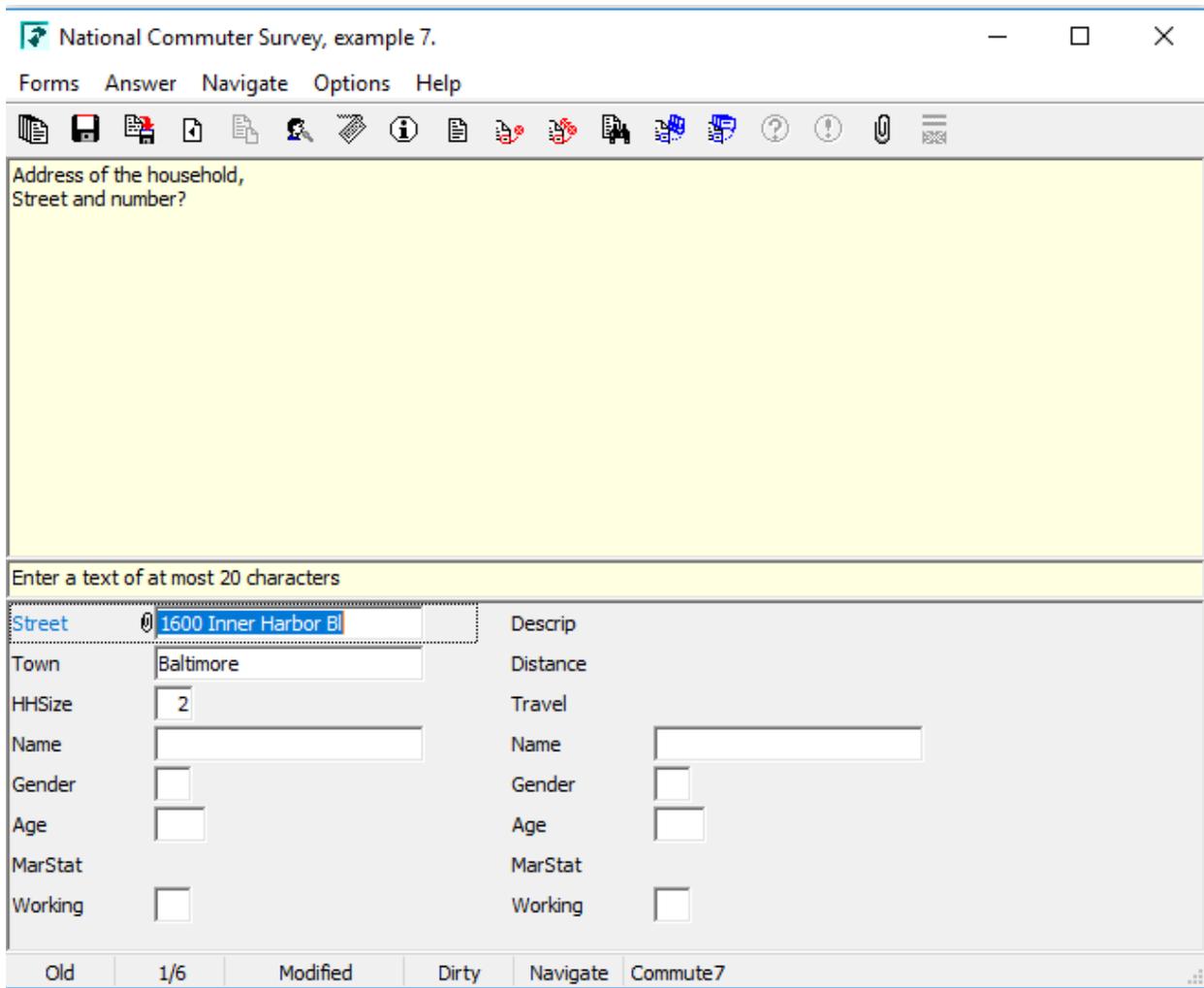


Figure 2. The resulting paperclip at the question where the remark was made

Blaise also allows for easy export of all remarks, per case or for all cases at once, to a separate text file.

## Problems with remarks

The problems that arise from interviewers resorting to remarks are twofold:

- The resulting data quality is not likely to be as good as if the change had been made by the interviewer, with the respondent still available
- Going through all the remarks for all interviews in the home office is time-consuming and expensive

Especially if interviewers are resorting to remarks to not break the flow of an interview, the number of remarks to work through for data editing staff can be quite high. There is no easy way to know which

remarks are actionable – contain enough information to make the intended changes to the data – or which ones are more important to data quality compared to others. As remarks are just text strings you really have no option to read all of them before you can prioritize. Depending on constraints in the data delivery schedule, staff availability and overall budget handling all remarks appropriately can become a problem.

One possible course of action is to try to make the interview instrument as user-friendly as possible for the interviewer, in order to reduce the need for making remarks in the first place. We have implemented attempts at doing exactly that, for instance by adding menu options that give access to review screens, and that may even allow quick access to a certain section and automatically return to the question from which the menu item was called.

The focus of this paper is however finding ways to make the handling of remarks at the home office more manageable. In order to make categorization possible before reading all remarks we will show how to replace the standard Blaise remark popup with a Manipula dialog box that can be tailored to the needs of the project at hand.

This starts with the DEP menu option for remarks, which will have to be ‘diverted’ to a Manipula procedure.

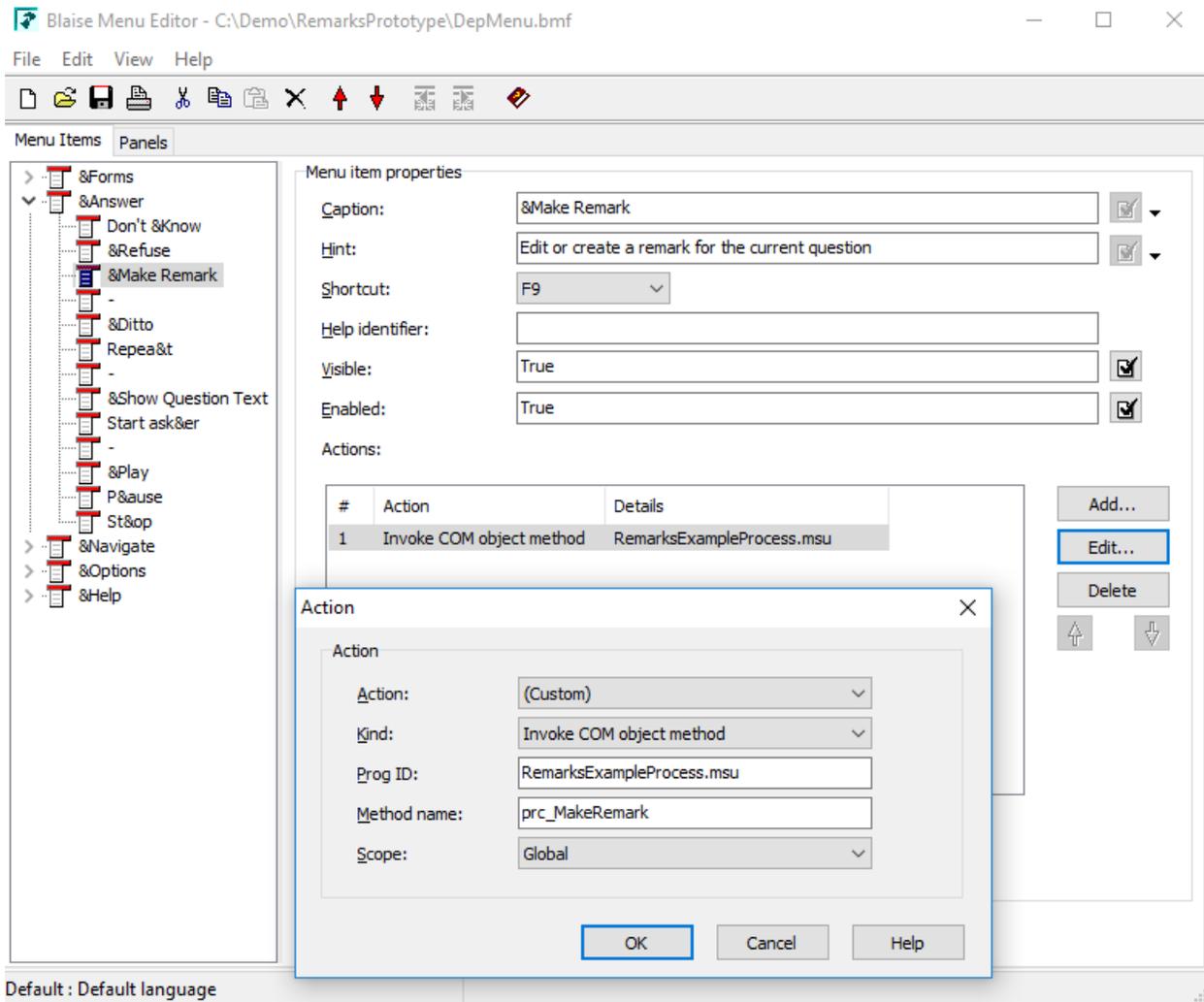


Figure 3. A screenshot of the DEP Menu Editor and the menu item to replace the standard remark popup with a customized version.

We tend to use the custom Action Kind ‘Invoke COM object method’ with the Scope set to Global, as the screenshot above shows, instead of selecting the probably more straightforward ‘Start Manipula’ option in the ‘Kind’ dropdown menu. This is done on purpose: using ‘Start Manipula’ will initialize/load the Manipula setup every time it is called. Using the method shown above the setup will be loaded once, at the start of the interview, and kept in memory. For a small Manipula process the speed difference may not be very noticeable, but for more extensive ones it really makes a difference in how fast the procedure gets executed. The lag at the very start of the interview may become fairly long if there are many setup to load, but a pause before the interview starts is a lot more manageable than several pauses later on, during the interview.

The next screenshot shows an alternative remark popup for the same remark as shown in Figure 1. The category at the top has to be selected before the remark text box becomes accessible.

National Commuter Survey, example 7.

Forms Answer Navigate Options Help

Address of the household,  
Street and number?

Enter a text of at most 20 characters

Street 1600 Inner Harbor Bl

Town Baltimore

HHSize 2

Name

Gender

Age

MarStat

Working

Old 1/6 Modified

Remark related to:

- Person name
- Mode Of Transport
- Work related
- Other

Remark:

Address is really 1600 Inner Harbor Blvd, but there's not enough space to enter that.

Figure 4. The same remark as in Figure 1, now with a custom remark popup that includes a category

Here is another example of a remark in the same interview:

The screenshot shows a web-based survey form. At the top, the title is "National Commuter Survey, example 7." Below the title is a menu with "Forms", "Answer", "Navigate", "Options", and "Help". A toolbar contains various icons for file operations and navigation. The main question is "How do you travel to your work?". Below the question is a list of seven options with checkboxes:

- 1. Do not travel, work at home
- 2. Public bus, tram or metro
- 3. Train
- 4. Car or motor cycle
- 5. Bicycle
- 6. Walk
- 7. Other means of transportation

Below the options is a text prompt: "Enter at most 3 values". This is followed by a form with several input fields:

Street	<input type="text" value="1600 Inner Harbor Bl"/>	Descrip	<input type="text" value="Sales clerk"/>
Town	<input type="text" value="Baltimore"/>	Distance	<input type="text" value="5"/>
HHSize	<input type="text" value="2"/>	Travel	<input type="text" value="1-2-3"/>
Name	<input type="text" value="Amber"/>	Name	<input type="text"/>

Below the form is a section titled "Remark related to:" with four radio button options:

- Person name
- Mode Of Transport
- Work related
- Other

Below the radio buttons is a section titled "Remark:" with a text box containing the text: "Can only answer three, but use all at one time or another (not every day)."

Figure 5. Another example of a categorized remark

And here is a screenshot of the Data Viewer, to show the external storage of the remarks data:

CaseID	Remark	FullFieldName	LocalFieldName	Field	RemarkCategory	Remark	DateRemarkCreated	TimeRemarkCreated
5	1	Street	Street		Other	Address is really 1600 Inner	9/25/2018	5:03PM
6	2	Work[1].Travel[1]	Travel		ModeOfTransport	Can only answer three, but us	9/25/2018	5:19PM

Figure 6. Data Viewer screenshot for the external remarks data file.

Note that the remark field doesn't show the full text here, in order to make the screenshot fit the page without resorting to an unreadable small font size. In the data file the remark is stored in its entirety however.

The interaction between DEP, DEP menu and external data model and data file is all done in the Maniplus process that is referenced in the menu file. It uses some of the extremely powerful capabilities of Manipula in Blaise 4.8:

- having access to the metadata and the data of the current interview, in particular the name of the current question in the interview
- the ability to read from and write to separate data files
- display dialogs that can use external data
- access the internal Blaise remarks (still used for the 'paperclip functionality')

Following in Figure 7. are a few pretty basic lines from the setup code showing the use of a temporary file and access to remarks and metadata information within the Maniplus code:

```

PROCESS RemarksExampleProcess

SETTINGS
  MessageFile = 'AnyRemarkErrors.txt'

USES
  Commute7
  RemarksStorage

UPDATEFILE theRemarks : RemarksStorage
  SETTINGS
    OPEN = NO
    MAKENEWFILE = NO

TEMPORARYFILE DepSession: Commute7
  SETTINGS
    INTERCHANGE = SHARED

{...}

{The procedure called from the DEP menu;}

PROCEDURE prc_MakeRemark
  AuxFullFieldName:= ACTIVEFIELD
  prc_GetRemark
  dlg_MakeRemarkDialog
  CASE OKCancelButton OF
    OK : DepSession.PUTREMARK(ACTIVEFIELD, TRIM(AuxRemark))
      prc_WriteToRemarksFile
  ENDCASE
  prc_ReleaseRemarksFile
  SETALIENROUTERACTION (BLRAEFAULT)
ENDPROCEDURE {prc_MakeRemark}

{And a few other lines showing the use of meta information}
AuxLocalBlock := DepSession.GETFIELDINFO(ACTIVEFIELD, 'BLOCKNAME')
AuxFileName:= DepSession.GETMETAINFO('DICTIONARYNAME') + '_Remarks.bdb'
theRemarks.OPEN(AuxFileName)

```

Figure 7. Several code snippets from the Maniplus setup

The complete setup used for this example is a relatively short one, perhaps 150 lines, a good part of which are taken up by the definition of the dialog box. The main point of including some of the lines here is to show that these kinds of changes and additions to standard Blaise DEP functionality are actually very easy to make.

## **Conclusion**

Blaise 4.8 offers ways to use Manipula as a separate but integrated layer of control for the Data Entry Program, opening doors for functionality that was not available before. This functionality can be used to great effect for the interview itself, as has been shown in previous IBUC papers. The small example in this paper shows how it can be used to also positively affect the post-interview stage of data editing.

The hope is that in a current or future version of Blaise 5 changes like these – if necessary at all – can be made as effectively.

# ScriptWriter – The compilation of Script Generation

*Max Malhotra*

*University of Michigan Survey Research Center*

## 1. Introduction to ScriptWriter

Using Blaise 5 for interviewer-administrated surveys pose unique technical challenges. One of these challenges is training interviewers in a clear and concise manner where they can learn the unique aspects of each data model in the shortest time possible. What we as an organization have observed is that round robin implementation of data model scripts is one effective mechanism for training interviewers. By utilizing the round robin technique, each interviewer participates in a different role and can verbalize questions and possible response options to simulate a genuine interviewer administrated session and ultimately build interviewer confidence.

This can be technically challenging as some data models are rapidly changing during the development stage. Scripts can consist of hundreds of pages and there currently is no out of the box solution provided for script generation. As a result, we built a utility in-house to handle Blaise 5 script generation. Here we will talk about some of the technical components behind the development of ScriptWriter which utilizes a number of Blaise APIs such as Meta, DataRecord, DataLink, and SessionData.

## 2. Background

ScriptWriter is a vital utility for our organization and the lifeblood of how Computer Assisted Telephone Interviews (CATI) and Computer Assisted Personal Interviewing (CAPI) training sessions are administered. Prior to ScriptWriter application, it was an arduous process which consisted of screen captures of the data models and countless hand edits in the attempt to simulate a script. This is especially problematic as scripts can consist of hundreds of pages.

The first version of ScriptWriter was created in VB.NET by Youhong Liu for Blaise 4.8. For Blaise 5, Max Malhotra wrote a new version of ScriptWriter in C# (.NET). The Blaise 5 version of ScriptWriter uses an Audit Trail Parser Utility developed for the organization by Mark Simonson. This utility parses out the audit trail data and returns the corresponding question order to the ScriptWriter application.

## 3. Overview

ScriptWriter B5 is a C# .NET program that facilitates the creation of interviewer training scripts. The application allows the user to enter training annotations while progressing

through a stand-alone version of the Blaise instrument. ScriptWriter produces .HTML output of the training annotations combined with the survey question text and the responses that the user has entered. The output includes only the user's path through the survey instrument, versus all survey items, which is ideal for producing an interviewer training script. If necessary, the .HTML output can easily be imported into Microsoft Word for further editing and saved as a Word document. However, the .HTML output is pre-formatted and "ready to go" with minimal editing required.

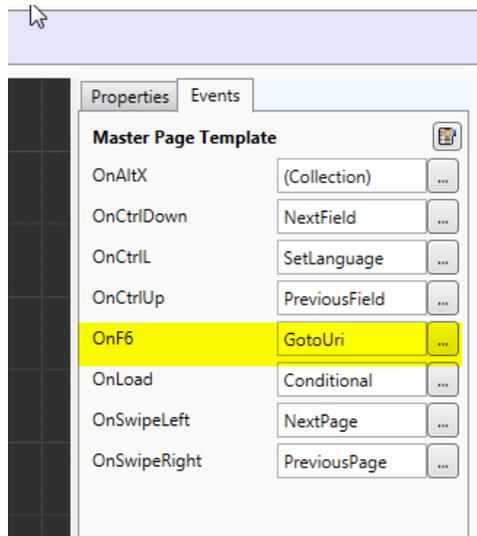
#### **4. Prerequisites**

- The application is designed to work in conjunction with the Data Entry Program (DEP) thus the appropriate DEP .dll(s) need to be available.
- .NET Framework installed.
- ScriptWriter is a database driven application and requires some initial back-end configuration for project identification.
- Any project using ScriptWriter requires modifications to the Blaise Resource Database (.blrd) in order to launch an external ScriptWriter Notes application.
- Proper network user access needs to be granted to person launching DEP or generating the scripts.
- Depending on how the Data Model was constructed some CSS modifications may be required.

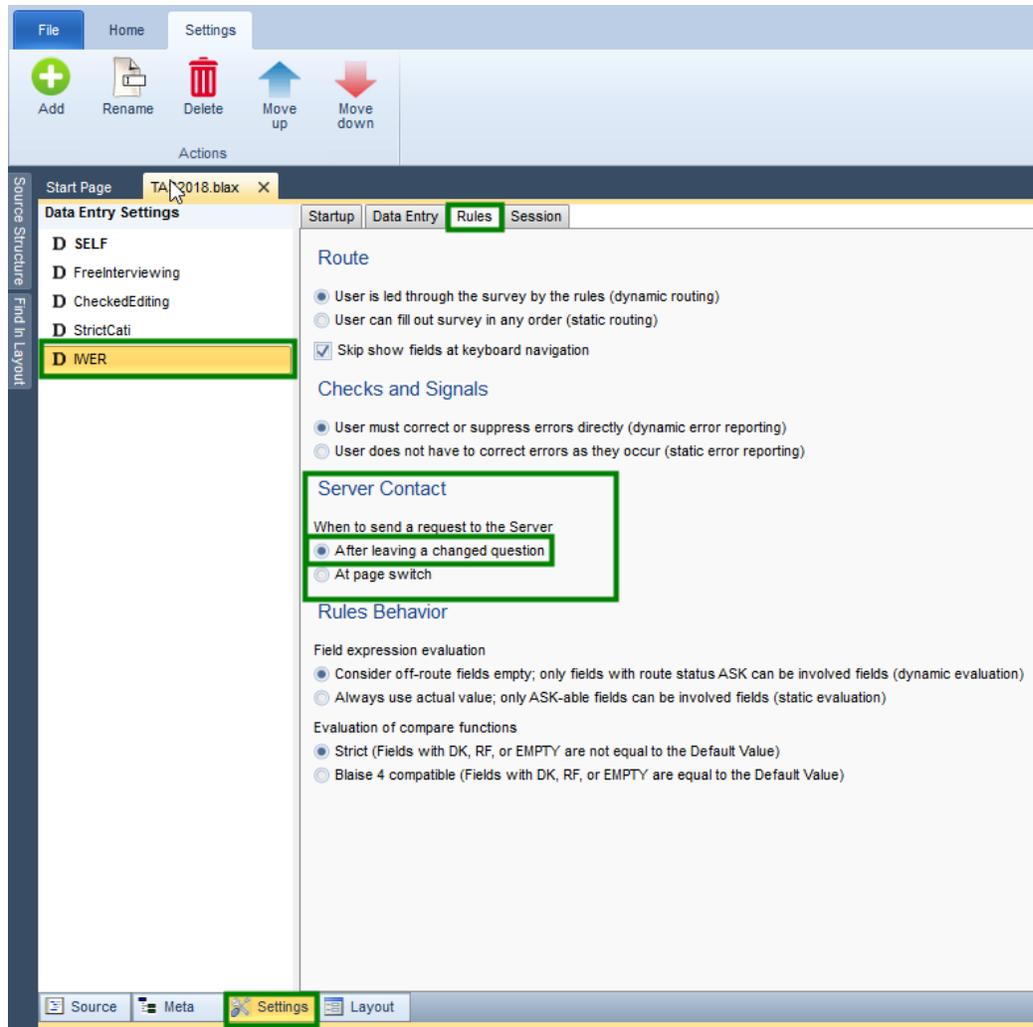
#### **5. ScriptWriter Setup**

Before using ScriptWriter, there is a small amount of set-up that is required on the part of the Blaise programmer. Also, the user(s) will need to install the application on their computer or work from where the application is housed on the network and obtain access rights to their specific project directory.

1. Open Microsoft SQL Server and go to Server Name: {Enter the DB server Name} Database Name: ScriptWriter Table Name: dbo.tProjects and add in an appropriate row for the Data Model and Server in question (make sure to fill out all pertinent information).
2. Modify the B5DESW.exe.config to the app in order to point to your appropriate Audit Trail Server.
3. Inside the Blaise Resource Database (.blrd) select your Master Page Template and add an OnF6 event.



- a.
  - b. Select the OnF6 event and add a GotoUri event passing in the following information (your Primary Key name may not be called SampleID as shown below so please remember to compensate accordingly. Or, if you are using SampleID, then you can map it to the Field Reference you are utilizing).
    - i. Argument : SampleID.ValueAsText + ' ' + State.ActiveFieldName + ' ' + State.InstrumentId + ' ' + State.LanguageName
    - ii. URI: {Network/Local Folder Path for ScriptWriter Notes Application}\B5DESW.exe
    - iii. Above we are passing arguments from the data model to the called external application.
4. Go to your DM Settings and make sure for whichever Data Entry Setting you plan to use, that the Server Contact is set to: "After leaving a changed question".
- a. If this is not done and you have multiple questions on a page, then ScriptWriter will only update the notes for the first question. This is because server contact has not been made to initiate the field being updated for the remaining questions.
  - b. Remember that your Data Entry Setting name may differ per DM and this is not an issue as long as you provide the correct name to the tProject table.



c.

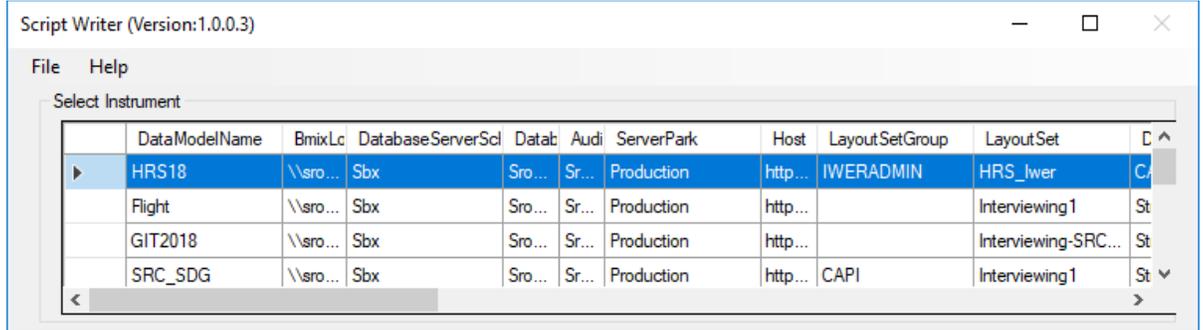
5. Load the appropriate preloaded lines.
6. Deploy the newly modified Data Model (DM) to the appropriate Blaise Server.
7. Launch the ScriptWriter Application => Select your DM from the list provided in the Data Grid View => Provide a Key Value => Click Script Gen to launch DEP => Press F6 to bring up the ScriptWriter Notes Application => Add Appropriate Notes => Save and Quit the instrument => Click Generate to output a new script for the questions you answered and the notes that you entered.

## 6. ScriptWriter Explained

### 6.1 User Interface (UI)

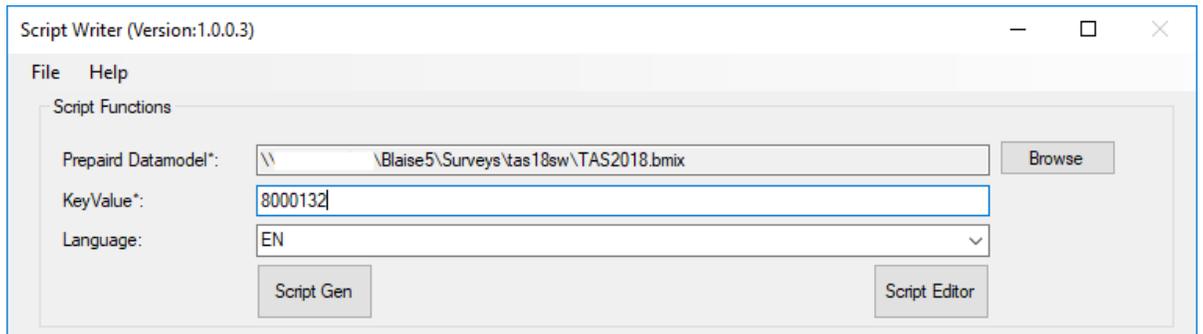
The premise behind the UI design of ScriptWriter was to make the application minimally intrusive to the end user, thus most work is performed behind the scenes in the code base. This way we are able to keep the number of user clicks to an absolute minimum.

## 6.2 ScriptWriter – Select Instrument



1. On this screen, you will choose the instrument and server set pairing you wish to work with.

## 6.3 ScriptWriter – Script Functions



1. Script Gen: On this screen, simply enter your KeyValue and Language. If you click on Script Gen it will launch DEP for that KeyValue in the language selected.
2. Script Editor: Alternatively, if you have already launched DEP before and created a script using DEP and the ScriptWriter Notes Application, then you can simply enter your KeyValue, select the Language and then click on Script Editor. This will allow you to edit your script or generate the script again in either the same or a different language based on what the data model (DM) provides.
  - a. NOTE: The language selected shall determine how the question text and answer categories appear. However, the notes you entered shall still appear in the same language as you entered them.

## 4 ScriptWriter – Script Gen

FES

UNIVERSITY OF MICHIGAN MEDICAL CENTER  
UNIVERSITY OF MICHIGAN

Respondent: ANNA -- MARTIN

◆ Verify that you are speaking to the correct TAS Respondent before proceeding:  
Before I get started, I would like to make sure I am speaking with the right person. You are ANNA - MARTIN? If No, ASK: Do you ever go by that name?

◆ If R never goes by the preloaded name, suspend the interview and seek out the correct Respondent

◆ Minor name corrections can be made in the contact information update screens at the end of the interview

1. Yes  
 5. No

Sample ID: 8000132 | Version Date: 9-5-2018 | Version Time: 10:35:00 | Current Date: 9-18-2018 | Current Time: 1:28:6 | V.1.6 | Section\_Intro.Q1

1. Clicking on Script Gen will launch the DEP window and put you at the appropriate location in the instrument.

## 6.5 ScriptWriter – Notes Application

Script Writer

Notes

Active Field Name: [Section\\_Intro.Q1](#)

Notes1

R: Yes, I am.

Notes2

IWER: Great, thank you for that conformation.

Notes3

Trainer: Iwer's should provide neutral feedback 30%-50% of the time and at the end of sections as learned in GIT. Please remind them if you notice they are not doing this. Throughout the interviews, interviewers should verify spellings of names, dollar amounts, dates, numbers, etc. provided by the INF or R. Please remind them if you notice they are not doing this. For this mock interview, you will have a DDU

Notes4

Trainer: READ TO IWERS: The goal of this mock interview is a) move through an interview more quickly - similar to a real iw and b) to give you another chance at practice and to see how the physical measures/biomarkers fit into the interview. You should all have your physical measures and biomarkers equipment bag. During this mock iw, we will be pairing up to complete the physical measures but will not

Notes5

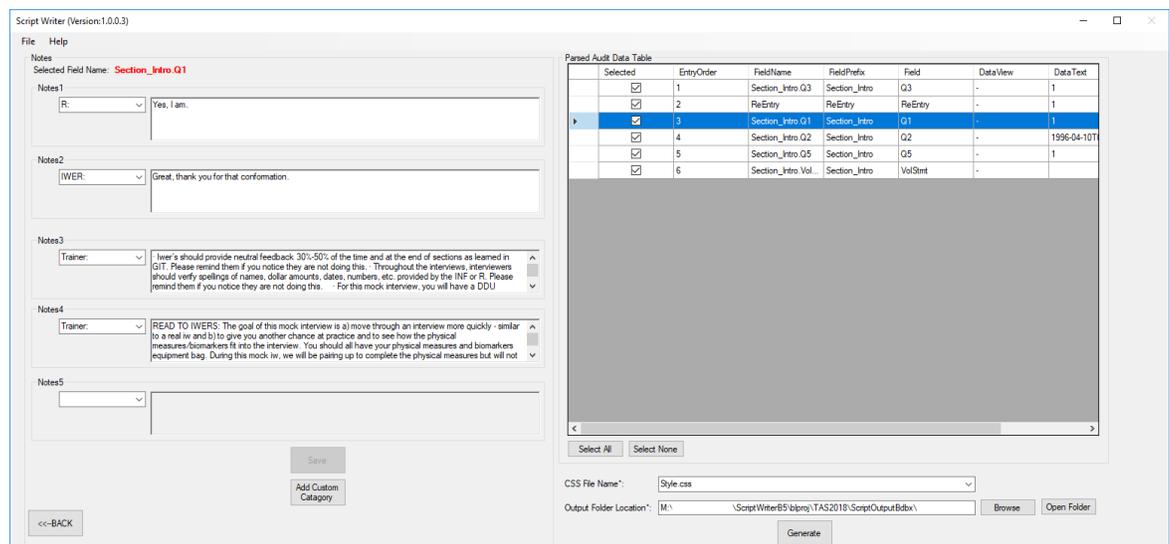
DDU Operator:

Save

Add Custom Catagory

1. After you are inside the DEP and inside the Data Model (DM), you can press the F6 button at any time to bring up the ScriptWriter Notes Application. This is where you will populate your script on a per question level.
2. You can always add additional Custom Categories at any time you feel appropriate for a question in the script. Simply click on the “Add Custom Category” button and your new category will show up in the notes drop down.

## 6.6 ScriptWriter – Script Editor



1. Upon either exiting DEP or clicking on the Script Editor button, the user will end up in the Script Editor window. The user can choose to modify the script by selecting the question and editing the note. Once the user is satisfied with the script the user can click on the “Generate” button to output the script.
2. Inside this window, you can choose what questions you wish to display in the generated output and are able to exclude any question you do not wish shown in the output.

## 6.7 ScriptWriter – Generated Script

**Verify R Section\_Intro.Q1**

We would like to make sure that this interview has reached the right person. Is your name ANNA.MARTIN, or do you ever go by that name?

1. Yes  
5. No

R:	Yes, I am.
IWER:	Great, thank you for that conformation.
Trainer:	Iver's should provide neutral feedback 30%-50% of the time and at the end of sections as learned in GIT. Please remind them if you notice they are not doing this. Throughout the interviews, interviewers should verify spellings of names, dollar amounts, dates, numbers, etc. provided by the INF or R. Please remind them if you notice they are not doing this. For this mock interview, you will have a DDU operator following along in Blaise and SurveyTrak. Remind Iwers to have their interviewer packet and spiral bound quick start cards with them as they will be referring to them during the mock iv. Iwers should have stop watches out for Section D. Iwers will need their PM/Bio bags for Section I.
Trainer:	READ TO IVERS. The goal of this mock interview is a) move through an interview more quickly - similar to a real iv and b) to give you another chance at practice and to see how the physical measures/biomarkers fit into the interview. You should all have your physical measures and biomarkers equipment bag. During this mock iv, we will be pairing up to complete the physical measures but will not re-practice the blood spot or saliva collection.

Value: 1

---

**Q2 Request Birth Date Section\_Intro.Q2**

What is your date of birth?

R:	4/10/1996
IWER:	To confirm you said, your birth date is 4/10/1996?

Value: 1996-04-10T00:00:00

---

**Q5 R has RB Section\_Intro.Q5**

- ◆ VERIFY that R has the Response Booklet with (him/her) for the interview.  
Do you have your Response Booklet with you (that we mailed to you with the letter asking for your participation)? We will be using this booklet throughout the interview. On certain questions, I will ask you to refer to a page in the booklet for help in answering the question. (Would you like to go find it?)
- ◆ IF R doesn't have the Respondent/Response Booklet with (him/her) or cannot find it, PROBE. If you're able to connect to the internet you could view the booklet online. The website is <http://fes.isr.umich.edu/TA2018-RB/index.html>
- ◆ RECORD whether R has the Respondent/Response Booklet, is viewing it online, or the interview is being done without the booklet.

1. R has Response Booklet with (him/her)

1. After clicking on the generate button you can view your generated output.
  - a. Behind the scenes, this is where the ScriptWriter application does the bulk of the heavy lifting. It utilizes Blaise APIs such as Meta, DataRecord, DataLink, and SessionData to construct the proper output based on audit trail order and question and answer categories as well as specific script text on a per question level and correlate that information based on the questions the user wishes to display.
2. As an added benefit, the use of the training scripts sometimes helps to flush out the question and answer categories inside the data model (DM).
3. If additional languages are available for the data model (DM), the user can simply select the desired language from the language drop down and generate a script in that chosen language. The question and answer categories will translate to the chosen language provided that option is available in the data model (DM). Otherwise, the default language question and answer categories will be shown for that question.

## 7. ScriptWriter Application Usage – End Users

### 7.1 Intended Audience of the Application

The target audience/end user of this application is for personnel who either administer training (trainer) or receive training (interviewer) pertaining to Computer Assisted Telephone

Interviews (CATI) and Computer Assisted Personal Interviewing (CAPI). In the case of our organization, this means both in-house -- our Survey Services Lab (SSL) and field personnel.

## **7.2 Administration of Training**

The output of the script that is generated by ScriptWriter is used as the bases of the training material and the script(s) are administered in a round robin style of training.

First, let us define what constitutes a round-robin training session. In our context, this essentially means creating a circle where the interviewers and trainer are part of a pattern and participate in role-playing. The person playing the role of the respondent can be part of the circle or inside the circle. In essence, the interviewer reads a question from the script then either the trainer or another interviewer plays the role of a respondent and respond to the question asked, this is a part of the same generated script as well.

It is completely study dependent in terms of what questions and respondent responses are part of the script. The ScriptWriter Application allows the user to easily choose what questions do or do not make it into the script. This allows the trainer who creates the script the ability to generate a controlled set of responses that cover the important parts of the survey.

A round robin training session is conducted for most studies and consists of typically between 2-20 interviewers per trainer This number is based on factors such as Data Model Size, length of study, trainer and interviewer availability, script length and question complexity.

These scripts help immensely to increase the confidence level of the interviewers. It also gives these individuals a familiarity with the question structure and the type of study the data model consists of prior to ever making a call or showing up on a respondent's doorstep for an appointment.

## **8. Conclusion**

Interviewer training scripts play an integral role in Computer Assisted Interviewing (CAI) studies. They are used in interviewer training to provide interviewers with practice through a Blaise questionnaire, while trainers highlight important features. Scripts are also used to assess interviewer questionnaire administration, ensure the quality of data, and offer a means to standardize training elements. Standardization is crucial to the proper training and certification of interviewers, especially across training teams with multiple training sessions.

The ScriptWriter Tool was developed to facilitate the script development process. The goal was to create a user-friendly application with the flexibility to serve the needs of many projects and diverse Blaise applications. In almost all cases, using ScriptWriter should result in significant time and resource savings in the production of interviewer training scripts, as compared with alternative methods. Since the question components are pulled directly from the Blaise instrument, the resulting script is accurate and contains the components necessary to ensure the proper training and certification of interviewers.

In closing, ScriptWriter serves as an important tool in our tool belt for training individuals in conducting Computer Assisted Telephone Interviews (CATI) and Computer Assisted Personal Interviewing (CAPI). Our organization continues to put the ScriptWriter application through its paces by utilizing new data models as they become available and we continue to try to improve upon it as new feature requests are made.

In the future, we wish to develop additional functionality that allows for the scripts themselves to be either sent to an external source or translated by a human or through automated means and easily be brought back and used by the application. This would essentially build a translation service for script components other than question and answer categories which are already capable of being translated.

## **References**

Blaise 5 Help. (2018). '<http://help.blaise.com>'

University of Michigan Survey Research Center. (2018). 'ScriptWriter B5 Setup Instructions', *p 1-4*

University of Michigan Survey Research Center. (2009). 'Scriptwriter Manual', *p 1, 24*

# New Features in Blaise Colectica Questionnaires

*Jeremy Iverson, Dan Smith, Colectica*

Blaise Colectica Questionnaires allows survey researchers to build surveys faster using an intuitive user interface, to leverage the DDI metadata standard, and to generate rich documentation and reports. The software improves transparency into the data capture process.

The third release, launching in October 2018, includes support for grids, rosters, fills and dynamic text, and text formatting. Also featured are collaboration improvements and flowchart export.

The software stores questionnaire specifications using the open DDI and GSIM standards, and can connect to metadata repositories and question banks powered by Colectica software. Data descriptions can be linked with source questions, creating harmonized data and showing data lineages.

Surveys designed with this tool can be fielded using Blaise 5 on the desktop, on the Web, and on mobile devices. The tool converts the DDI metadata into a Blaise project and source code. Changes to surveys made with the tool can be published and executed within the Blaise environment, allowing rapid iteration while developing surveys.

## Benefits

### Faster Survey Development Iterations

Blaise Colectica Questionnaires offers an intuitive survey design surface and questionnaire palette, allowing survey designers to build questionnaires without learning a domain specific language.

### Question Banks Powered by Open Standards

Questions, blocks, and logic can be created within the program or reused from question bank powered by DDI. Reusing standardized questions assists in creating more comparable data.

### Multiple Outputs

Surveys designed with Blaise Colectica Questionnaires are stored as a specification in DDI format. From this single survey specification, multiple outputs can be created automatically:

- Blaise 5 source code
- Paper form
- PDF specification
- Publish to Colectica Repository and Colectica Portal
- DDI metadata

Custom formats can also be built, including custom reports, delimited formats, and additional survey systems.

## **New Features in October 2018**

### **Grids**

A question grid is a series of questions that share the same response options, possibly with header text.

In Blaise, a grid can be specified using a `GROUP .. ENDGROUP` block with `FIELDS` that share the same response type. A `LAYOUT` with a `OptionsButtons` template displays this on the screen as a grid.

In Blaise Colectica Questionnaires, a question grid can be specified by providing the header text, response options, and the text of the questions. This is stored as a standardized question grid in DDI format. When targeting Blaise 5, the appropriate `GROUP` and `LAYOUT` statements are generated. See Figure 1 for example source code.

### **Rosters**

A question roster is a series of questions asked about one or more members of a list, for example members of a household.

In Blaise, a roster can be specified with a `FOR` loop that asks a `BLOCK` multiple times.

In Blaise Colectica Questionnaires, a roster can be specified by providing the following information:

- The maximum number of roster entries allowed
- A value (possibly from a previously asked numeric question) indicating the number of items in the roster
- The sequence of questions to be asked

This information is stored as a DDI Loop construct that iterates over several `QuestionItems`..

Blaise Colectica Questionnaires then generates the appropriate code, including the necessary `BLOCKS`, `FOR` loop, and table `LAYOUT`. See Figure 2 for example source code.

### **Dynamic Text**

The ability to describe fills or dynamic text has been added to Blaise Colectica Questionnaires in this release. This new feature allows a question to use fills and dynamic text, and still be able to be used across different surveys.

When a question is described in DDI, it is globally uniquely identified and made available for use by reference in many different surveys. However, in each survey, a text fill described by a question will likely use a different source variable. To allow reuse of a single question across the contexts of different surveys, a DDI formatted question can describe input parameters. This is much like an input parameter for a block in Blaise. For example, the question can describe a text input parameter named `personName`. When describing the text fill in DDI, this `personName` input parameter is used as the replacement token in the text. All tokens used for fills are input parameter names designated by the individual question description.

To connect an input on a question, the Blaise Colectica Questionnaires tool presents the user with a list of variables present within the survey that are of a compatible data type. This list includes all variables that are in the current scope within the survey, or could be brought into scope. In DDI, a binding ties together a variable and a parameter. A binding is created to tie together the user-selected variable in the scope and the input parameter of the question. This allows a variable such as `name` or `firstName` to be paired with the `personName` input on our example question. Through this coupling, question definitions are portable across surveys while still allowing for usage of survey specific values in fills and dynamic text. The Blaise Colectica Questionnaires user interface hides this complexity from the user, enabling easy reuse across surveys of complex fill and dynamic text question definitions.

## **Text Formatting**

In DDI, Colectica uses the text-based Markdown format, a lightweight markup language, for creating rich text. This Markdown format is also used for describing question text. The Blaise Colectica Questionnaires tool now converts Markdown formatted text into the appropriate Blaise emphasis tags. The Blaise predefined tags of bold, italic, underline, hyperlink, image, headings, horizontal lines, line breaks, spaces, and tables are all supported.

## **Collaboration**

When used together with Colectica Repository and Colectica Portal, Blaise Colectica Questionnaires allows users to collaborate during survey development. Blaise Colectica Questionnaires allows users to comment on any item, including the overall survey, individual blocks, or specific questions. These comments are stored on the repository, so any user looking at a survey while connected to the repository will see what comments other users made.

Users can also see the change history of individual questions, blocks, or the entire survey. This functionality is similar to the Track Changes and Review functionality in programs like Microsoft Word. The goal is to make survey review and iteration faster and more auditable.

## **Reusable Questionnaires**

Blaise Colectica Questionnaires is designed to allow building reusable questionnaires, reusable sections of questionnaires, and reusable questions and response options. The goal is to enable standardization and reuse while keeping the survey author from the underlying complexity by using an intuitive user interface.

The survey author specifies questions, block sections, and logic guided by the visual interface. This guided and structured creation of the questionnaire, as oppose to using a Word document, allows the Blaise Colectica Questionnaires tool to record the structured definition using DDI, an open XML standard for describing questionnaires and survey data. All components of the survey are versioned and identified, allowing for both multi user collaboration and audit trail functionality.

When a survey author wishes to compose a new questionnaire, they can create new questions and components, or they can select questionnaire items from a institutional repository. Blaise Colectica Questionnaires will visually guide the author while creating new items to ensure that the structures are accurately captured using the DDI standard. The tool also allows searching for existing questionnaire items, such as blocks, questions, and response options, to reuse. Browsing a repository for additional items to reuse allows for comparable data collection across waves of a survey or across surveys.

Blaise Colectica Questionnaires enables publishing to a central repository, which can be thought of as an enhanced question bank, storing multiple types of survey components. When a survey author publishes new items to repository, the items are recorded centrally and made available for reuse. Changes to items also are published to the repository and this action creates new revision of changed items, creating an audit trail recording who changed what and when. Publishing to a central repository also enables other users and processes to view changes to a survey instrument as they happen and follow development more closely. The centralized search and audit trail of all questionnaire components can be used by survey authors, project leaders, or automated software agents applying business rules.

Blaise Colectica Questionnaires is able to publish a DDI based questionnaire specification to Blaise. To publish a survey specification to Blaise, the author simply has to click a button in the software and Blaise survey is generated without involving programmers. The software creates all types and fields, adds specified edit checks, blocks, and rules section, all based on the DDI-based specification. The source code generation also insert comments, denotes DDI identifiers and versions of components, and generates Blaise-to-DDI mapping files for use in data documentation processes after survey fielding and data collection.

After a survey is fielded and the data have been published, the reusable questionnaire items are still available in the central repository. A survey author starting on a new survey can reuse questionnaires, sections, and questions from prior surveys. The tool allows authors to import blocks and questions, and specify their required input parameters if any conditional routing or dynamic text relies on values outside the scope of the shared item. The survey author can also import and reuse defined types and standardized questions and sections across surveys.

The reuse of questionnaire components by Blaise Colectica Questionnaires allows tracking comparable data over time. An analyst can track data sets with similar types and question usage, see which data were captured by the same question in different questionnaires, and search and track defined type/codelist/classification usage. This centralization and versioning over time enables researchers to search the repository for likely comparable data.

## Future Developments

Blaise Colectica Questionnaires remains in active development. Ideas include:

- Prefill data
- Export flowchart images
- Question manager
- Generate interviewer manuals, generated based of additional question metadata
- Support for Blaise PARALLELS
- Sample surveys

Most ideas for new features come from users of the software. Ideas can be submitted to the Colectica or Blaise teams.

## Appendix A: Blaise Source Code Examples

Figure 1: Blaise source code used to represent a question grid.

```
GROUP grid1
  FIELDS
    x, y, z : String[5]
  ENDGROUP

LAYOUT
  AT grid1 TABLE TEMPLATE OptionButtons
```

Figure 2: Blaise source code used to represent a question roster.

```
BLOCK BPerson
  FIELDS
    name : String
    age : 0..120
    sex : (male "Male", female "Female")

ENDBLOCK

BLOCK BRoster
  FIELDS
    Person : ARRAY[1..10] OF BPerson
  LOCALS
    i : INTEGER
  RULES
    i := 3
```

```
FOR i:= 1 TO 10 DO
  Person[i].ASK
ENDDO
ENDBLOCK
```

```
FIELDS
  PersonRoster : BRoster
```

```
RULES
  PersonRoster.Ask
```

```
LAYOUT
  AT PersonRoster TABLE TEMPLATE Table
```