Servicing Blaise instrument needs in a multi-mode environment: Some technical examples

Richard Frey, Kathleen O'Reagan, and Nona Brown
Westat
International Blaise Users Conference, April 2012, London, UK

Summary: *As response rates have dropped and budgetary constraints limit telephone surveys, many projects today are operating in multi-mode project environments that offer more than one way for respondents to submit survey answers. This paper looks at how Blaise 4.8 capabilities and features work with multi-mode projects requiring fast setup and processing. It includes a technical discussion of issues encountered, what worked well and lessons learned in using Blaise as the solution. We will discuss the use of a single database and how to handle mode differences such as question types and text.*

Introduction

In a perfect world, specifications would be written that are universal across modes. That said, some projects are not planned as multimodal and subsequently need to be modified to accommodate the differences between modes. The goal is to attain a seamless integration of all modes, in this instance Computer Assisted Telephone Interviewing (CATI) mode, Computer Assisted Web Interviewing (CAWI) mode, and paper surveys, resulting in a unified system. We chose to make use of a single SQL database for all modes to minimize issues associated with moving files back and forth between databases when the mode of interviewing changes. The management system will still keep track of which mode currently owns an interview. A case in data collection in one mode can create work activity in another mode or close out work in another mode. The decision to use one database requires planning but this is much less effort than would be required to maintain and synchronize a different data model and database for each separate mode.

Multi-Mode Management System

While a Multi-Mode Management System (MMMS) is not the focus of this paper there should be a mention of the role it plays in a multimodal environment.

The role of an MMMS is to provide a single interface that project systems can use to load cases and manage the complexity of interfacing with each individual mode. Figure 1 depicts an example of an MMMS that can:

- Control the activation of cases across multiple modes.
- Create new cases based on the completion status of prior cases, e.g., if you get a completion for a web survey, schedule a follow-up call 1 month later.
- Provide a user interface to permit managers to query case status and manually activate/deactivate cases or change modes.
- Implement a service layer so that updates between MMM and modes can occur in real-time.
- Allow for new modes to be brought online more easily.

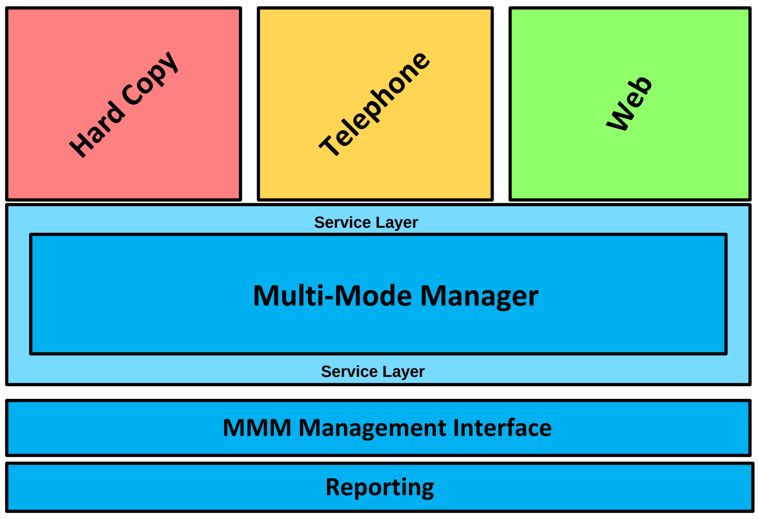


Figure 1. Example of Multi-Mode Manager

Using a Single SQL Database

In many multi-mode environments a mode is usually designed with a myopic view. Each mode has a data collection instrument, a database and a management system all different than other modes. As data collection proceeds, synchronization between modes becomes more and more difficult and time consuming. Programs have to be written such as to determine what mode a case is currently assigned, where it came from and completion status. Eventually the data will have to be merged for analysis, which again, necessitates writing programs to merge the mode data.

We chose to use Blaise for our multi-mode solution. Integrating the diverse modes using Blaise is relatively straight forward if you keep a few things in mind. To have one database you first need the instrument structure to be the same across all modes. In other words, instruments for each mode will have the same fields as any other mode. What will make each modes instrument unique will be the rules of the main datamodel and blocks. Conditional Defines are used to direct prepare directives to establish the rules of the datamodel for a mode.

Once the datamodel structure is finalized, a Blaise OLE DB Interface (boi) file can be created along with the tables in the SQL Database. The boi file is then used by each mode to read, write and update the database. However, each mode will only store data based on the rules of its instrument.

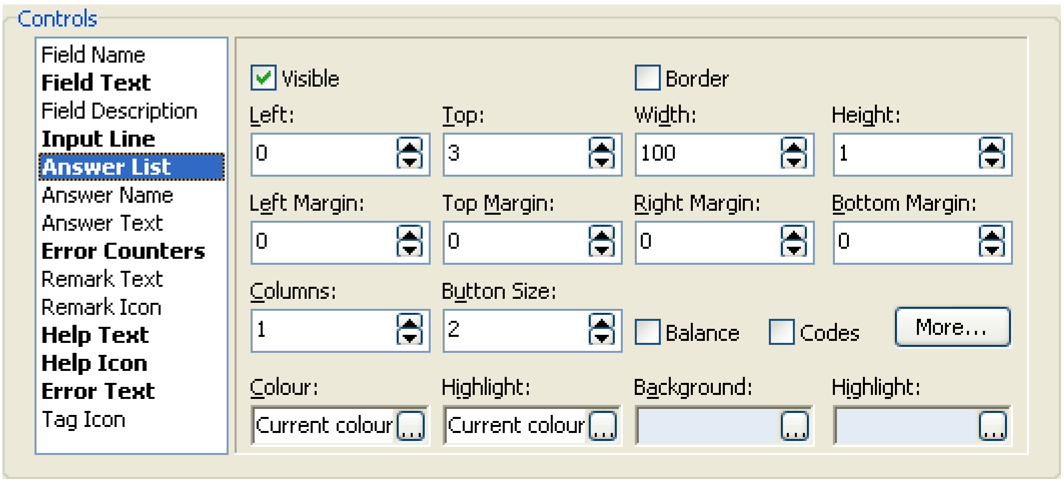
Customizing the Mode Library

Each collection mode will have its own customized Mode Library, also referred to as the modelib. Within a multi-mode environment there are times when you might want to combine the layout sets of a modes modelib. In doing so you increase the number of layout pages generated when the instrument is prepared. For each layout set in the modelib, the prepare process produces the individual pages with screen layout information needed for the DEP. The total number of pages is limited to 16,384 pages, easily reached with a large datamodel containing many arrays.

When using separate mode libraries the appropriate library must be used when preparing an instrument for a collection mode. Choosing the proper mode library is done at prepare time using prepare directives, specifically the {$MODELIB} prepare directive. This directive is used at the very beginning of the instrument as shown below.

```
{$IFDEF WebLayout}
    {$MESSAGE 'Preparing Web Version of our Instrument'}
    {$MODELIB C:\SampleInstrument\Config\Web_Inst.bml}
{$ELSE}
    {$MESSAGE 'Preparing DEP/CATI Version of our Instrument'}
    {$MODELIB C:\SampleInstrument\Config\CATI_Inst.bml}
{$ENDIF}
```

For the CAWI mode, one of the design decisions required the display of all enumerations and sets in one column, and indentations five positions from the left side of the question text. In addition, we increased the size of the radio buttons and further separated the radio button from the category text. These global changes were made to the properties of the Answer List default FIELDPANE in the Internet layout set of the CAWI modelib. The property changes are shown in Figure 2, the Answer List control.



Controls

Field Name
Field Text
Field Description
Input Line
Answer List
Answer Name
Answer Text
Error Counters
Remark Text
Remark Icon
Help Text
Help Icon
Error Text
Tag Icon

[✓ Visible] Border

Left:	Top:	Width:	Height:
0	3	100	1

Left Margin:	Top Margin:	Right Margin:	Bottom Margin:
0	0	0	0

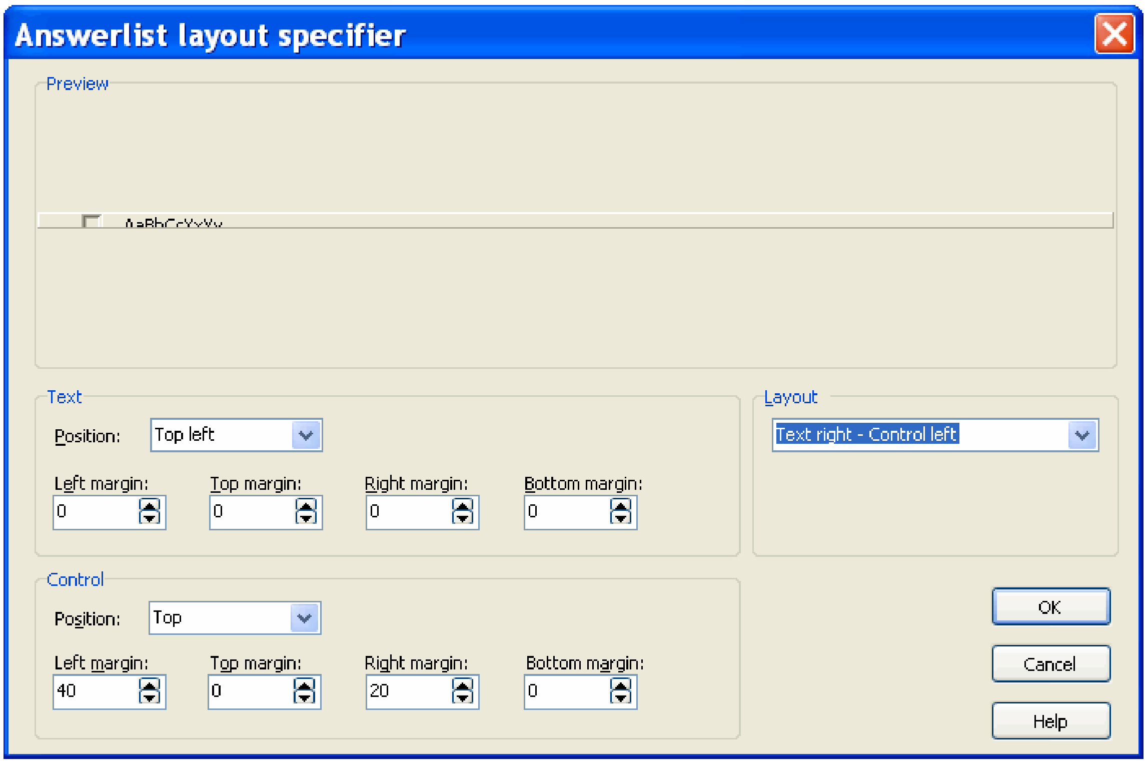
Columns:	Button Size:
1	2

Balance Codes More...

Colour:	Highlight:	Background:	Highlight:
Current colour	Current colour		

Figure .2 Example of Answer List Control

The radio button and text positioning were set using the Answer List Layout Specifier which is launched by selecting the More… button on the Answer List controls property panel as shown in Figure 3.


Answerlist layout specifier

Preview

AaBbCcYyYy

Text

Position: Top left

Left margin: 0

Top margin: 0

Right margin: 0

Bottom margin: 0

Layout

Text right - Control left

Control

Position: Top

Left margin: 40

Top margin: 0

Right margin: 20

Bottom margin: 0

Figure 3. Answer List Layout Specifier

Lastly, the color scheme of the project's web site had to be incorporated into the layout of the survey to provide a consistent look and feel across pages of the projects web site. Using the color scheme tool the default project colors were easily set for each controls color related properties.

Question Layout Decisions

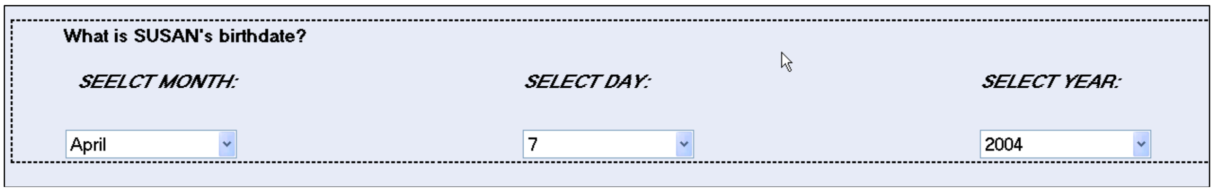
One of the important decisions when designing the screen layouts is determining how the survey questions will be displayed in CATI mode or grouped on a page in CAWI mode. When examining the survey, you'll see the majority of questions will display across modes without any or minimal changes. The remaining will need some modifications, especially for the overall look and feel of the web page including the header, content area and footer. Modifications will include the formatting of questions and question groups such as multi-column, group tables, or other specifies as part of an enumeration.

A multi-column group type is useful for positioning several questions next to each other. Examples of applications include quantity-unit questions where the respondent has to specify both a quantity and the unit, or date of birth questions which include month, day and year.

Figure 4, shows a date of birth question using the multi-column format. Typically there is lead in text for the month question and then instructions for the day and year responses. In a multi-mode environment, prepare directives determine the text to be used. The following shows the month, day and year field definitions using prepare directives:

```
ChildBirthMonth
{$IFNDEF WebLayout}
"What is ^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's
birthdate?
    @/@/@/@I[ENTER MONTH.]@I"
{$ELSE}
    "What is
^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's birthdate?
    @/@/@IENTER MONTH@I"
{$ENDIF}
    : TMonth,DK,RF
ChildBirthDay
{$IFNDEF WebLayout}
  "@E[What is
^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's
birthdate?]
    @/@/@/MONTH: ^ChildBirthMonth@E@|@|@|@I[ENTER DAY.]@I"
{$ELSE}
  "@/@/@IENTER DAY@I"
{$ENDIF}
    : TDay,DK,RF {TI1_31}
ChildBirthYear
{$IFNDEF WebLayout}
"@E[What is
^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's
birthdate?]
    @/@/@/MONTH:^ChildBirthMonth  @|@|@|DAY:
    ^ChildBirthDay @E@|@|@I[ENTER YEAR.]@I"
{$ELSE}
"@/@/@IENTER YEAR@I"
{$ENDIF}
    : TYear,DK,RF {TI1990_2012}
```

An alternative would be to first duplicate the group type questions, then separate the two sets using prepare directives and modify one set for Web display only.


What is SUSAN's birthdate?

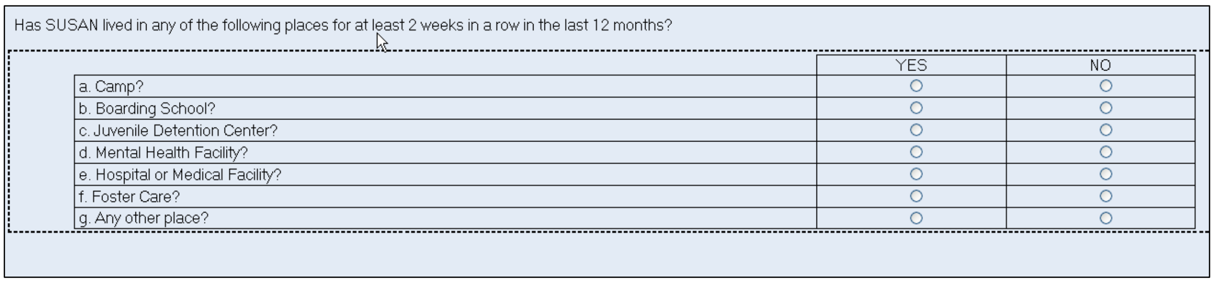
SEELCT MONTH:	SELECT DAY:	SELECT YEAR:
April	7	2004

Figure 4. Alternative way to display the fields

Additional layout items for the example include setting the month, day and year types to display as dropdowns and using the internet FIELDPANE Dropdown. The survey specifications include the dropdown view for all modes.

Another group type, group table, has a lead in question followed by several enumeration questions all having the same answer options. This group type is useful for positioning questions below each other in table form such as a series of Yes/No questions.

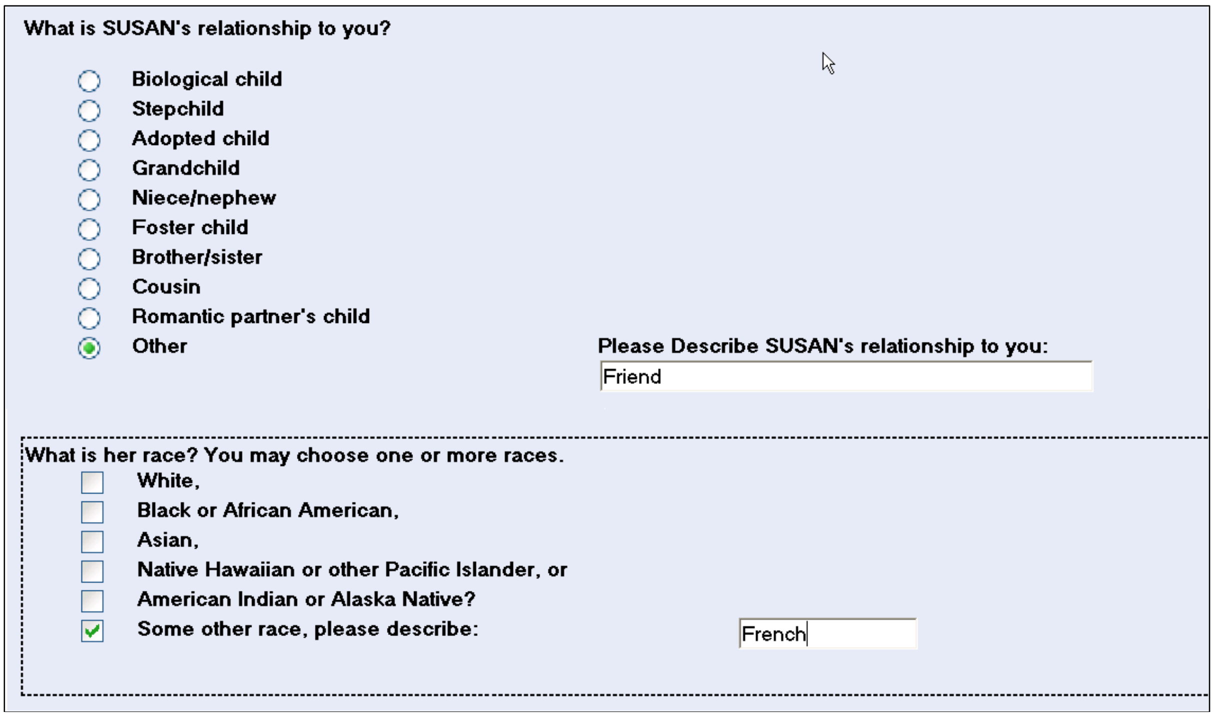
The multi-mode changes for this group are the same as the multi-column. The Blaise fields used in the layout as shown in Figure 5, were duplicated and differentiated for the web by prepare directives. An AUXFIELD was created to hold the text for the lead in question and given the SHOW method. The SHOW method causes the web page to skip the AUXFIELD text and give the first row in the table the focus.


Has SUSAN lived in any of the following places for at least 2 weeks in a row in the last 12 months?

	YES	NO
a. Camp?	○	○
b. Boarding School?	○	○
c. Juvenile Detention Center?	○	○
d. Mental Health Facility?	○	○
e. Hospital or Medical Facility?	○	○
f. Foster Care?	○	○
g. Any other place?	○	○

Figure 5. Example of web question using Show method

The final group type we'll look at is the Other Specify. This group type uses a category of an enumeration to associate the input field for the other specify. It is relatively simple to implement and the only decision is whether the text used to 'Please describe:' is contained in the enumeration category text or as the question text of the other specify field. Figure 6 shows both ways of formatting the other specify.


What is SUSAN's relationship to you?

- ○ Biological child
- ○ Stepchild
- ○ Adopted child
- ○ Grandchild
- ○ Niece/nephew
- ○ Foster child
- ○ Brother/sister
- ○ Cousin
- ○ Romantic partner's child
- ● Other

Please Describe SUSAN's relationship to you: Friend

What is her race? You may choose one or more races.

- ☐ White,
- ☐ Black or African American,
- ☐ Asian,
- ☐ Native Hawaiian or other Pacific Islander, or
- ☐ American Indian or Alaska Native?
- ✓ Some other race, please describe: French

Figure 6. Ways of formatting Other Specify

Interviewing Instructions, Differences between CATI and CAWI Modes

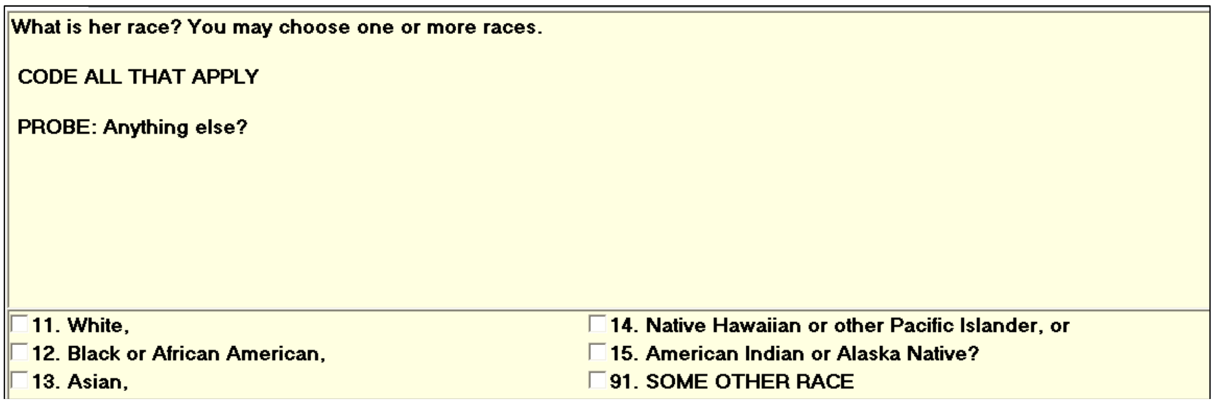
In addition to layouts, there are a number of other differences that should be taken into consideration when programming for a multi-mode survey. One difference that needs to be accounted for between CAWI and CATI mode is interviewing instructions. Standard CATI interviewing instructions may not be appropriate for a CAWI. Case formatting of text in CATI indicates whether the text is read to the respondent. If the text is in all upper case, the text is not read to the respondent and if in upper/lower case, the text is read to the respondent.

The simplest way to accommodate the interviewing instructions and upper/lower case differences between interview modes is through a combination of prepare directives and display fields.

An example would be a code all screen that typically has two interviewer instructions. One states ‘CODE ALL THAT APPLY’ and the other states ‘PROBE: Anything else?’ While the first instruction may be appropriate for a CAWI, the second is not. We accommodated the second interviewer instruction by creating a display field and using prepare directives set the field to the appropriate value as shown below.

```
{$IFNDEF WebLayout}
  dispProbe := “PROBE: Anything else?”
{$ELSE}
  dispProbe := EMPTY
{$ENDIF}
```

Figure 7 depicts a CATI code all screen and Figure 8 shows the same question displayed on the web.



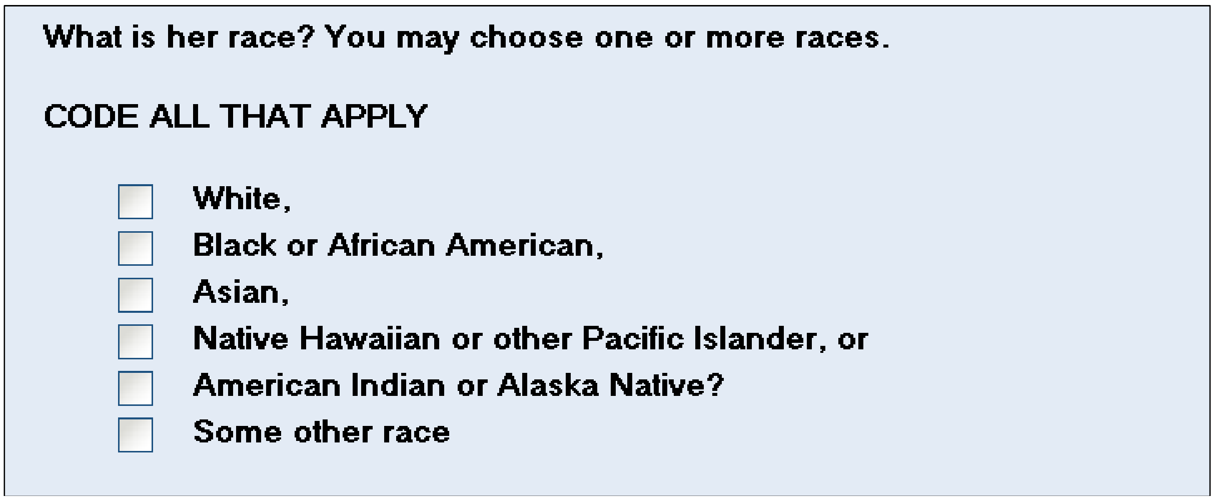
What is her race? You may choose one or more races.

CODE ALL THAT APPLY

PROBE: Anything else?

☐ 11. White,
☐ 12. Black or African American,
☐ 13. Asian,
☐ 14. Native Hawaiian or other Pacific Islander, or
☐ 15. American Indian or Alaska Native?
☐ 91. SOME OTHER RACE

Figure 7. Code all screen example in CATI



What is her race? You may choose one or more races.

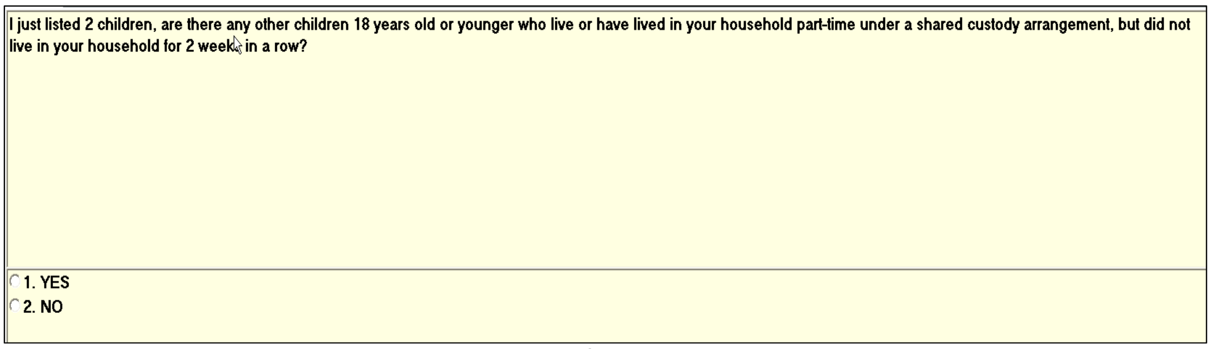
CODE ALL THAT APPLY

☐ White,
☐ Black or African American,
☐ Asian,
☐ Native Hawaiian or other Pacific Islander, or
☐ American Indian or Alaska Native?
☐ Some other race

Figure 8. Example of code all screen on web

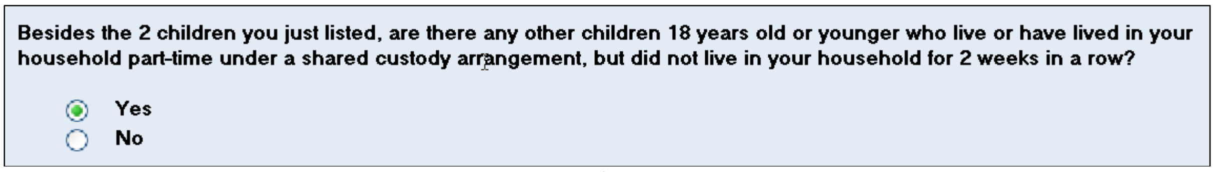
Although the first person is not used frequently in CATI interviews, there might be a question that contains wording such as “I just listed 2 children…” as depicted in Figure 9. The corresponding web text might have “Besides the 2 children you just listed…”, as shown in Figure 10. Just as interviewing instructions were handled in the code all screen above, so too would the question text displays be handled using prepare directives as shown below.

```
{$IFNDEF WebLayout}
  AuxScrText := "I just listed 2 children,"
{$ELSE}
  AuxScrText := "Besides the 2 children you just listed,"
{$ENDIF}
```


I just listed 2 children, are there any other children 18 years old or younger who live or have lived in your household part-time under a shared custody arrangement, but did not live in your household for 2 weeks in a row?

1. YES
2. NO

Figure 9. Example of CATI question phrasing


Besides the 2 children you just listed, are there any other children 18 years old or younger who live or have lived in your household part-time under a shared custody arrangement, but did not live in your household for 2 weeks in a row?

[✓ Yes] No

Figure 10. Example of web question phrasing

As part of interviewer training, the interviewers are taught not to read to the respondent anything that is in all capital letters. This includes enumerated type questions such as YesNo whose answer categories are all capitalized. Some enumerated type questions end in an ellipsis and these types have their answer categories in mixed case. We wanted to make the web survey look as uniform as possible. To that end, when we were close to the end of CATI development we made a copy of the Types file, renamed it, changed all the lettering to mixed case, and used prepare directives to include the new version of the file for the CAWI interview.

```
{$IFDEF WebLayout}
  INCLUDE "AdultWEB_Types.inc"
{$ELSE}
  INCLUDE "Adult_Types.inc"
{$ENDIF}
```

Whereas interviewers are trained to put in a DK response and a comment when a Hard Check occurs and the respondent insists that the data inconsistency is correct, CAWI respondents have no such training. In order to prevent the Web respondent from becoming frustrated and ending the interview early, we have found it advisable to use Soft Edits, where possible, instead of Hard Edits.

In addition, we found there were edits used in CATI that were not appropriate for CAWI. It is important to remember when adding or removing an edit with prepare directives, a RESERVECHECK needs to be added or removed depending on the collection mode. The following prepare directive code shows how to do this.

```
{$IFDEF WebLayout}
  RESERVECHECK
{$ELSE}
  CHECK
  IF (MisReportedChild = RESPONSE) THEN
    (ChildFName <> EMPTY)
       "@REMPTY LINE CANNOT BE MARKED AS MISREPORTED.@R"
  ENDIF
{$ENDIF}
```

Using Help in the Different Modes

The requirements for the project had no Help specified for the CATI and very little for the CAWI. The help requirements for CAWI were to add web links to a contact page, a list of study definitions and list of helpful numbers as shown in Figure 11. These were to be available for the respondent throughout the entire interview.

Figure 11. Example of web help options

Conclusion

Using a central repository for the storage of data from different collection modes worked well. The repository is not an assembly of different databases each bound to a specific collection mode. Rather it contains one database shared among the different modes. Blaise capabilities and features support rapid multi-mode survey development and implementation.

The first step to a successful implementation of a multi-mode survey using Blaise is to understand how Blaise instruments can be programed to operate in different collection modes and share one SQL database. What worked well was the establishment of a standard data structure that included common and unique fields from each collection mode. The structure was then translated into SQL tables using the Blaise Datalink component.

In addition to a establishing a standard data structure, prepare directives allowed each mode's instrument to have a look and feel appropriate for the collection platform, while not compromising the integrity of the data.

In taking on an endeavor like a multi-mode survey planning is important. The primary items encountered in this case were formatting fields and question text, and identifying critical fields that would be used to refresh text e.g. certain fill strings.

A major lesson learned was make sure the survey specifications take into account the differing multi-mode characteristics. For example, text for a CATI mode should have corresponding text for a web mode. However, projects are not always planned to be multi-mode, and may become multi-mode after the initial programming has been completed. In these instances projects may spend additional time and resources modifying the specifications to accommodate the differences between modes.