IBUC 2023
DURHAM, NC, USA

Conference Proceedings

20th International Blaise User Conference

October 24-26, 2023

RTI INTERNATIONAL

Preface

This document contains the papers presented at the 20th International Blaise Users Conference (IBUC) hosted by RTI International on October 24 to 26, 2023, in Durham, North Carolina, USA.

The Scientific Committee organized and planned the conference program, chaired by Gina-Qian Cheung, Independent Blaise Consultant. The members of the committee were:

- Gina Cheung (The chair of the Blaise Users Group)
- Jane Shepherd (Westat, USA)
- Tim Carati (Blaise, Statistics Netherlands)
- Roger Linssen (Blaise, Statistics Netherlands)
- Ramasubramanian Suresh (RTI International, USA)

RTI and the committee worked together to create this proceedings book for the benefit of the conference participants and others.

Table of Contents

Blaise 5 at Statistics Netherlands

Maarten Pouwels, Jeroen Schröder, Statistics Netherlands

Table of Contents

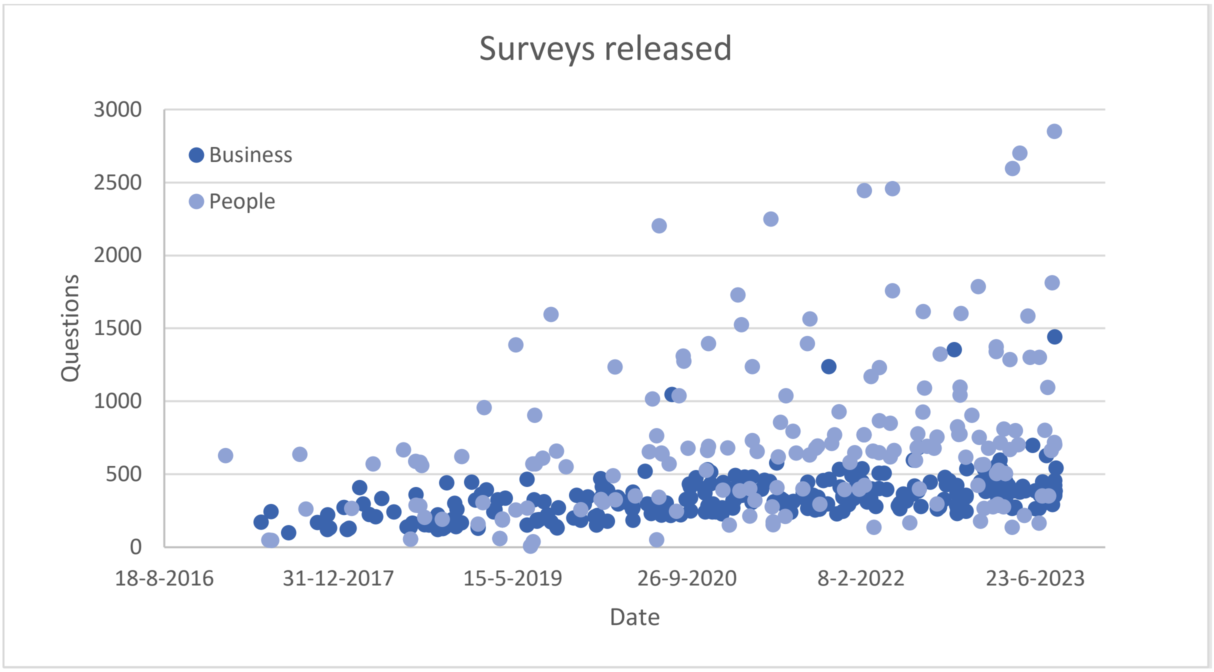
1. Overview Data Collection with Blaise 5

In 2014, Statistics Netherlands started a project (Phoenix) to renew the data collection process and upgrade the data collection systems in place. Blaise 5 is a massive part of this upgrade. As of 2023, all data collection is performed through the Phoenix system. In total, there are more than 130 surveys in use at Statistics Netherlands. Half of these statistics are targeting people and half are targeting businesses. Most of these surveys are repeated yearly, apart from some pilot surveys with no follow-up and some surveys which are repeated monthly, quarterly, or over a number of years. Most of the business surveys are based on Eurostat regulations and are therefore repeated yearly. One objective of the Phoenix project was replacing all Blaise 4 surveys with Blaise 5 surveys. Blaise 4 is no longer used for any surveys at Statistics Netherlands. This process went gradually. First, the CAWI surveys with the lowest complexity were migrated. These were "CAWI-only" surveys with standard question types. The surveys' lower complexity and smaller size, and the fact that they only made use of the CAWI mode, made them suitable candidates to start the migration process.

Starting in 2020, the master-detail surveys were migrated to Blaise 5. To replace the master-detail, system parallels were used instead of the detail surveys. In 2021, a start was made on moving CATI surveys into the Phoenix system. In 2022, the last surveys were implemented in the Phoenix system using the CMA app for CAPI surveys.

> back to Table of Contents

Figure 1: Number of Surveys Released per Year



2. Team Setup and Roles

At Statistics Netherlands, there is a department for social statistics, a department for business statistics, and a department for data collection. When a new survey is needed or an existing survey needs to be updated, the statistical department contacts data collection and a project is started to get the new survey up and running. A project team is formed with every member of the team having a specific specialized task regarding the different parts of the survey, such as sampling, communication, and building. The building of the survey is done by a survey designer and a survey builder.

2.1 Survey Designer

The survey designer, as the name suggests, is responsible for the design of the survey. The design consists of 2 documents. The first is a Word document that clarifies what questions will be asked and how they will be formulated. The formulation of the questions needs to be checked against several things. First, does everyone understand the questions that will be asked? Some words are considered too difficult for the general public, and other times, the formulation needs to be tactically accurate. It's possible the question has too many options. In this case, it might be easier to make it into 2 separate questions. Also, the text should not be ambiguous to avoid misconceptions. The answer options should be clear, and the respondent should not be forced into a statement they do not feel comfortable with. All questions must have a unique variable name so the statistical department can correspond the output to the questions asked.

The document is created in Word because it's a widely used text processor and most people are familiar with it. This allows the statistical department to also read the design document. Special macros were built to make the design easier to implement. This also makes it easier for questions to be standardized, which allows the builder to use tooling and convert the question automatically.

The second document contains the routing of the survey. This is created using Visio because it's more visual. This contains the routing for the survey and any checks that need to be implemented. When there are technical questions, the survey designer is advised by the builder about what is possible and

> back to Table of Contents

what possible advantages and disadvantages there are. When the design is finished and approved, it is sent to the survey builder to create the survey. When the development of the survey is completed, the designer conducts the first test to see if everything has been implemented correctly. When this process is concluded, the statistical department is also invited to test the survey.

2.2 Survey Builder

When a design has been checked and approved by the statistical department, it is sent to the builder. Using Blaise 5, the builder creates the survey as specified in the design. This includes creating the questions and implementing the routing and checks, but also testing and making sure everything is built using the latest standard.

When making a survey, as well as making sure the survey works as described, we also make sure that if a different builder must update the survey or make small adjustments, they can do so quickly and easily. By standardizing the process of creating a survey as much as possible, most surveys are easily managed and uniform. By using tooling created at Statistics Netherlands, a base project can be automatically generated. This ensures that when building a survey, the base design is standardized and can be used to build the rest of the survey. This also makes sure you have the latest version of all standard parts of a survey, like the welcome page and the receipt page shown when the survey has been filled in. We also have rules about code formatting to increase code readability. This includes rules for code indentation and IF THEN ELSE statements, but also variable naming of auxiliary fields and locals, etc.

3. Resource File

At Statistics Netherlands, 1 resource database is used for all surveys. Because different surveys have different needs, lots of parameters are used to make sure surveys can be adjusted as needed without altering the resource database. There is a difference in layout between the master pages for business surveys and people surveys. These differences are all mapped to different variables in the resource file, which can be selected as needed. Using the Project Builder, the builder can easily select the type of survey to be built. This automatically adjusts the settings as needed, reducing the amount of settings that have to be set manually, preventing mistakes.

The current resource database has 4 different resource sets. The default layout is the large CAWI layout for laptops and computers. It’s set at a screen width of 1024 pixels, regardless of available space. If the survey has an index, the width is 1280 pixels. If the screen is between 1024 and 1280 pixels, the index is not shown.

There is also a CAWI layout for smartphones. The smartphone layout contains all question types that are available to the regular CAWI layout, but to accommodate the smaller screen group, templates are displayed in a different manner. The questions are stacked on top of one another instead of beside each other. This does mean that there are mode differences between the smartphone and large resource set. Research is conducted to lessen the mode effects. For the option list template, different variants were tried out to make sure differences between modes are as small as possible. Special care is taken to make sure straight lining (filling in the questions all the same without reading the question) is less common and that the variant people find most pleasant to use is implemented.

Currently, the smartphone resource set is mainly used for people surveys and not business surveys. Work is underway to implement the smartphone layout for business surveys wherever possible. Research is being done to determine what businesses primarily use smartphones for their day-to-day activities, as opposed to using a computer, and whether surveys aimed at them are suitable for a smartphone layout. The biggest problem is displaying tables on a smartphone. Small tables with only

> back to Table of Contents

1 column should be possible, but tables with many rows and columns are difficult to implement on smartphones. Business surveys also tend to have a lot of text to provide additional information to respondents. For instance, in a survey on employment, additional information might be given on what is meant by employees and whether people hired through a temp agency also belong in this category. These texts are shown with help buttons in the large CAWI layout, but those buttons don't work on a smartphone. New solutions have to be found for this problem and others like it.

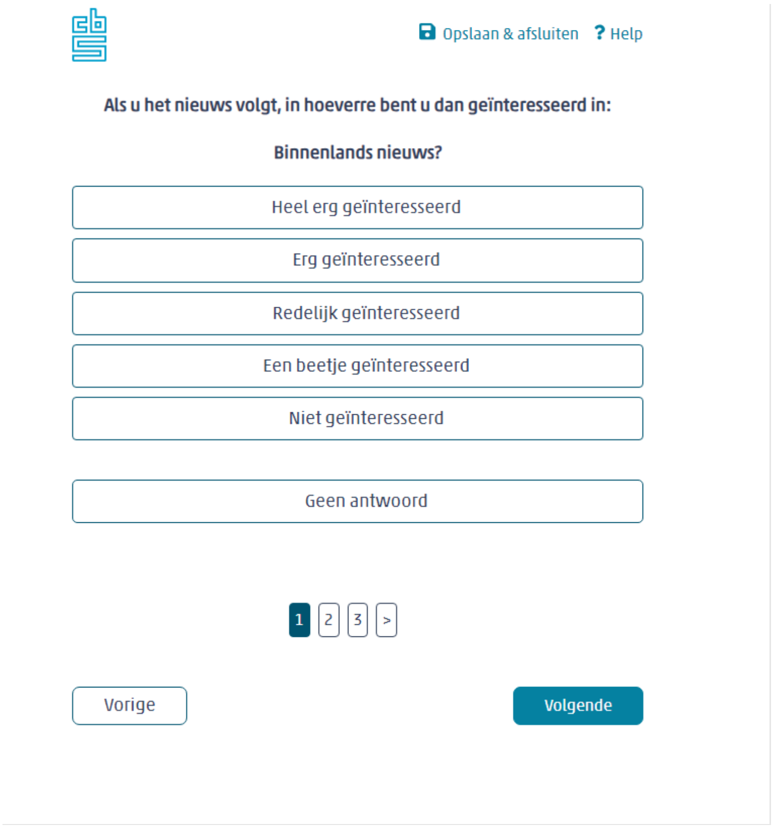
A different resource set is used for the printed version. Previously, the printed version was primarily used by the respondent to make a copy of their filled-out survey for their own documentation. Back then, the printed version was a part of the CAWI mode. The current resource set has moved the print to its own mode, which creates new possibilities for the printed version. An example of this would be that it's now possible to create a summary of the filled-out summary, as opposed to only being able to print out the full survey.

Finally, there is a CATI/CAPI resource set. The main difference between this resource set and the others is that in this resource set, more information on the respondent is available to the interviewer on screen. It is also possible to fill out this survey with minimal use of the mouse so an interviewer can fill out the survey quickly and accurately. One of the ways in which this is accomplished is by including a field in which the interviewer can enter a number corresponding to an answer for enumeration and set questions.

Currently, work is underway on redesigning the resource file. One of the goals of this new resource file is to modernize the look and feel of the surveys and to make the smartphone the focus of the design, rather than an afterthought. By making the smartphone the main focus of the design, problems like the different look for option lists between the smartphone and computer layout can be minimized. This, in turn, eliminates as many mode differences as possible. Research is underway to find the best way to ask matrix questions as conveniently as possible while eliminating mode differences.

> back to Table of Contents

Figure 2: Matrix Question in Carousel



Als u het nieuws volgt, in hoeverre bent u dan geïnteresseerd in:

Binnenlands nieuws?

Heel erg geïnteresseerd

Erg geïnteresseerd

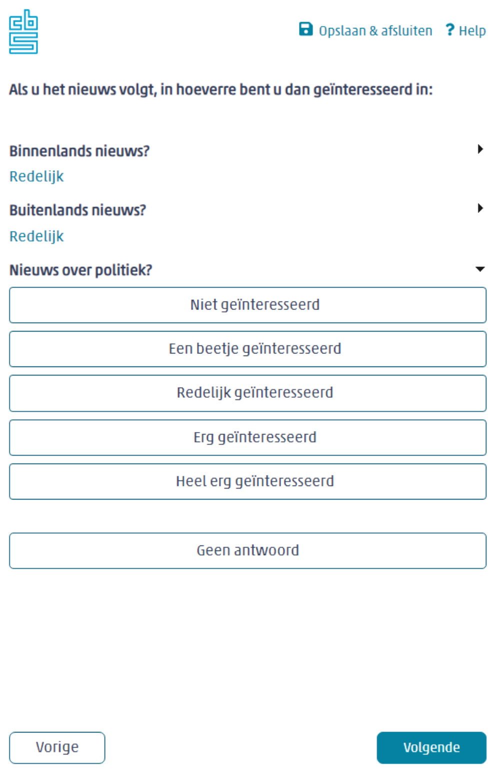
Redelijk geïnteresseerd

Een beetje geïnteresseerd

Niet geïnteresseerd

Geen antwoord

Figure 3: Matrix Question in Accordion Style



Als u het nieuws volgt, in hoeverre bent u dan geïnteresseerd in:

Binnenlands nieuws?
Redelijk

Buitenlands nieuws?
Redelijk

Nieuws over politiek?

Niet geïnteresseerd

Een beetje geïnteresseerd

Redelijk geïnteresseerd

Erg geïnteresseerd

Heel erg geïnteresseerd

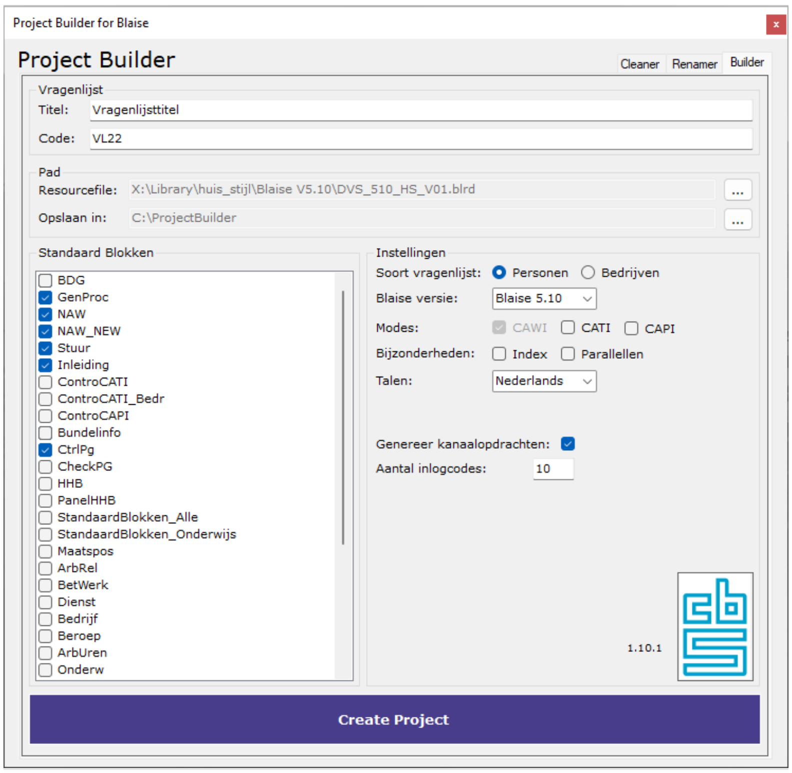
Geen antwoord

> back to Table of Contents

4. Project Builder

To ease the development of surveys at Statistics Netherlands, tools are used that are developed in house. One of these tools is the Project Builder.

Figure 4: Project Builder



Project Builder for Blaise

Project Builder

Vragenlijst

Titel: Vragenlijsttitel

Code: VL22

Pad

Resourcefile: X:\Library\huis_stijl\Blaise V5.10\DVS_510_HS_V01.blrd

Opslaan in: C:\ProjectBuilder

Standaard Blokken

- ☐ BDG
- ☑ GenProc
- ☑ NAW
- ☑ NAW_NEW
- ☑ Stuur
- ☑ Inleiding
- ☐ ControCATI
- ☐ ControCATI_Bedr
- ☐ ControCAPI
- ☐ Bundelinfo
- ☑ CtrlPg
- ☐ CheckPG
- ☐ HHB
- ☐ PanelHHB
- ☐ StandaardBlokken_Alle
- ☐ StandaardBlokken_Onderwijs
- ☐ Maatspos
- ☐ ArbRel
- ☐ BetWerk
- ☐ Dienst
- ☐ Bedrijf
- ☐ Beroep
- ☐ ArbUren
- ☐ Onderw

Instellingen

Soort vragenlijst: [✓ Personen] Bedrijven

Blaise versie: Blaise 5.10

Modes: ☑ CAWI ☐ CATI ☐ CAPI

Bijzonderheden: ☐ Index ☐ Parallellen

Talen: Nederlands

Genereer kanaalopdrachten: ☑

Aantal inlogcodes: 10

1.10.1

Because of the number of surveys created at Statistics Netherlands each year, the need arose for an automated way of setting up projects to speed up the building process. This included both the way in which teams were set up, but also the way in which surveys were built.

Before the creation of the Project Builder tool, most projects were set up using a step-by-step guide. As this was quite time consuming, an investment was made to see what parts of the process could be automated. As most projects use similar settings and make use of a number of standard blocks, the Project Builder tool was created to take care of these steps.

The Project Builder presents the user with a number of options for the survey to be created. This includes the project title and project code, what type of survey it will be, and what modes are to be used. It gives the option to show an index for the survey and asks whether the survey makes use of a master-detail type setup. It then gives the user the option to either make it single language or multi-language. And lastly, the user can create test cases.

After these options have been selected, a number of blocks on the left side of the screen are preselected based on those options. These can be edited depending on what's needed for the survey to be created.

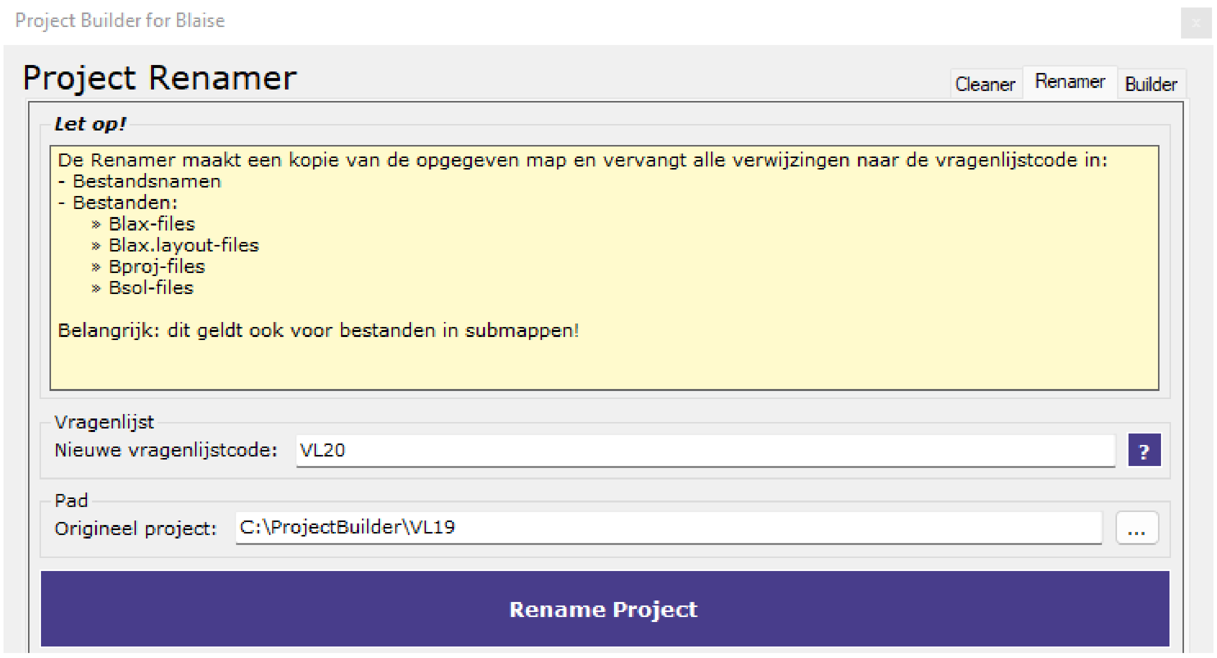
By clicking "Create Project," the project will be created based on all of these settings. The user can then use this to further implement the survey-specific questions and settings.

> back to Table of Contents

5. Project Renamer, Project Cleaner, and Project Verifier

There are a number of other tools used at Statistics Netherlands to make the process of creating surveys more streamlined. One of these tools is the Project Renamer.

Figure 5: Project Renamer



Project Renamer

Let op!

De Renamer maakt een kopie van de opgegeven map en vervangt alle verwijzingen naar de vragenlijstcode in:
- Bestandsnamen
- Bestanden:
  » Blax-files
  » Blax.layout-files
  » Bproj-files
  » Bsol-files

Belangrijk: dit geldt ook voor bestanden in submappen!

Vragenlijst
Nieuwe vragenlijstcode: VL20

Pad
Origineel project: C:\ProjectBuilder\VL19

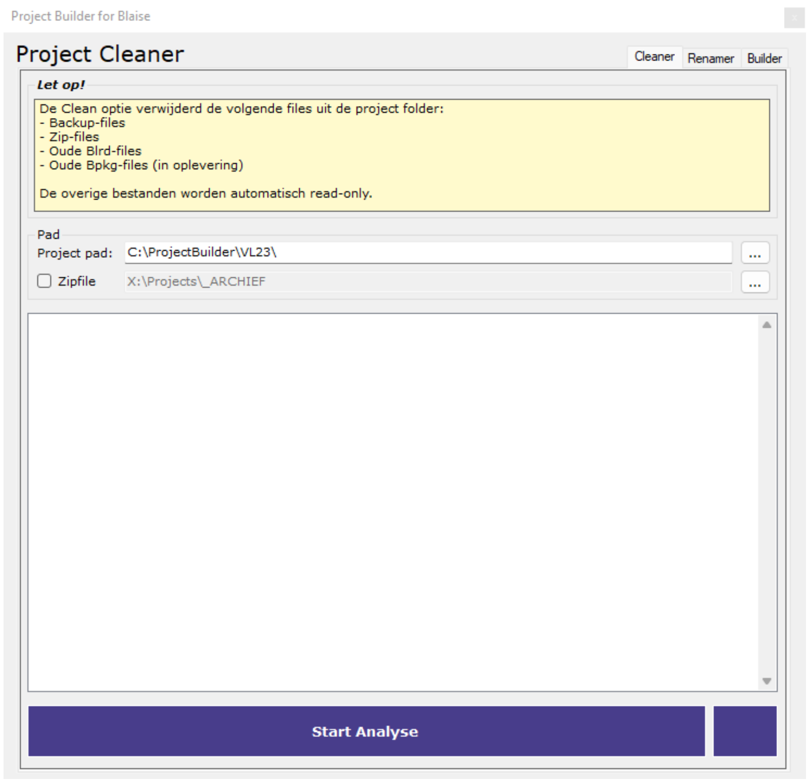
The Project Renamer gives the user the option to make a direct copy of a project with a different project code. This is convenient for surveys that are done yearly with minor edits between versions. It also allows for easy testing and bug fixing.

To use it, the user points the Project Renamer to the folder containing the original project and provides the new project code. The Project Renamer searches the selected folder and all containing folders for files having the old project code in its name. It also searches files to see if there are any references to the old project code inside the file.

There is also a Project Cleaner, which is used when a project is finished. It cleans up the project folder, removing all files which are not required to run the project. It also gives the option to create a backup, which is saved to an archive.

> back to Table of Contents

Figure 6: Project Cleaner


Project Builder for Blaise

Project Cleaner

Let op!

De Clean optie verwijderd de volgende files uit de project folder:
- Backup-files
- Zip-files
- Oude Blrd-files
- Oude Bpkg-files (in oplevering)

De overige bestanden worden automatisch read-only.

Pad

Project pad: C:\ProjectBuilder\VL23\

☐ Zipfile X:\Projects_ARCHIEF

Lastly, there is a Project Verifier. This tool is used when a survey is ready for testing. As a final check, the user can run the Project Verifier to see if there are any settings which were missed before testing. The Project Verifier will let the user know if there are any irregularities in the project.

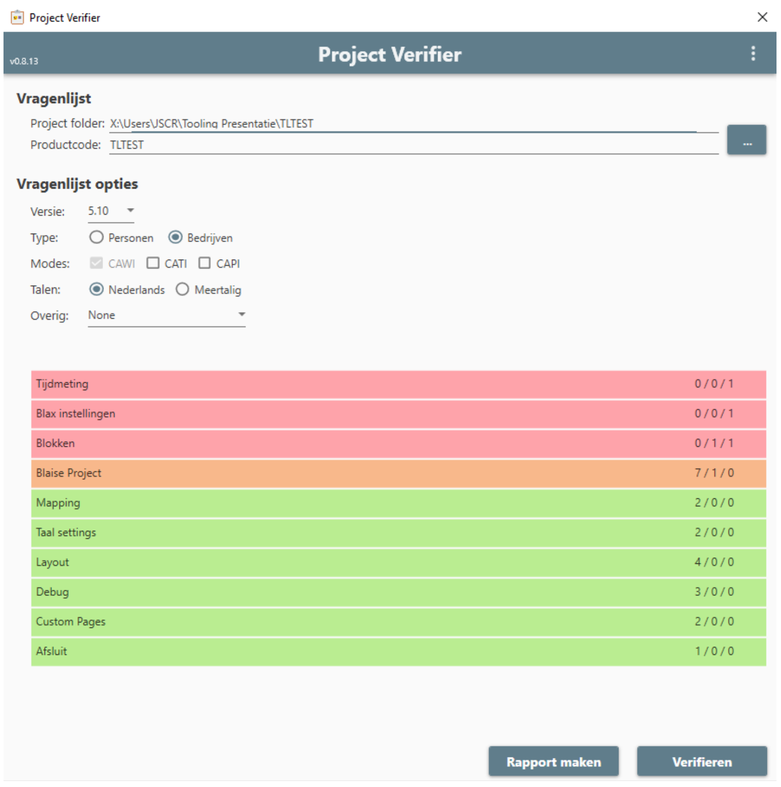
The Project Verifier has a number of options that let it know what is required of the project. This includes the Blaise version used, the type of project, the modes used, if it's multi-language, and if there is an index.

The Project Verifier uses these options to check the project. It returns a number of categories which can either be red, orange, or green, as seen in the screenshot below. In general, red means there is an error that needs to be resolved, orange is a warning, and green means all is well.

An example of a check that is performed is a debug check, which lets you know if there is any debug commentary left in the project. It also lets you know if there are any standard blocks that are normally added to a project of this type missing in the checked project.

> back to Table of Contents

Figure 7: Project Verifier



Project Verifier

v0.8.13

Project Verifier

Vragenlijst

Project folder: X:\Users\JSCR\Tooling Presentatie\TLTEST

Productcode: TLTEST

Vragenlijst opties

Versie: 5.10

Type: Personen [✓ Bedrijven]

Modes: [✓ CAWI] CATI CAPI

Talen: [✓ Nederlands] Meertalig

Overig: None

Tijdmeting	0 / 0 / 1
Blax instellingen	0 / 0 / 1
Blokken	0 / 1 / 1
Blaise Project	7 / 1 / 0
Mapping	2 / 0 / 0
Taal settings	2 / 0 / 0
Layout	4 / 0 / 0
Debug	3 / 0 / 0
Custom Pages	2 / 0 / 0
Afsluit	1 / 0 / 0

6. Design Converter

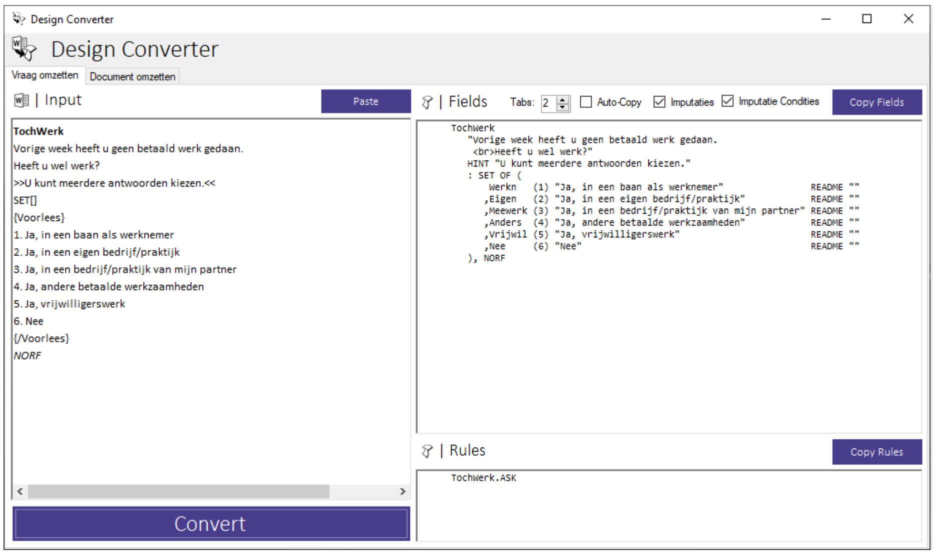
One of the most convenient tools is the Design Converter. The Design Converter allows the user to automatically convert (parts of) a design to Blaise code.

Because the designs used at Statistics Netherlands have been standardized, it is possible to detect different types of questions using this tool. The input side takes a question from the design. The Design Converter interprets the question and converts it to the corresponding Blaise code. There are a number of options to customize the output. This includes the number of tabs to be used when formatting the output, and automatically copying the output so the user can directly paste it to the Blaise project. It’s also possible to include the code for imputations.

Currently, this only works for single questions, but work is underway on a tool which will be able to interpret full designs.

> back to Table of Contents

Figure 8: Design Converter



Tabs: 2 [ ] Auto-Copy [✓] Imputaties [✓] Imputatie Condities

```
TochWerk
Vorige week heeft u geen betaald werk gedaan.
Heeft u wel werk?
>>U kunt meerdere antwoorden kiezen.<<
SET[]
{Voorlees}
1. Ja, in een baan als werknemer
2. Ja, in een eigen bedrijf/praktijk
3. Ja, in een bedrijf/praktijk van mijn partner
4. Ja, andere betaalde werkzaamheden
5. Ja, vrijwilligerswerk
6. Nee
{/Voorlees}
NORF
```

```
      TochWerk
         "Vorige week heeft u geen betaald werk gedaan.
          <br>Heeft u wel werk?"
         HINT "U kunt meerdere antwoorden kiezen."
         : SET OF (
             Werkn   (1) "Ja, in een baan als werknemer"              README ""
            ,Eigen   (2) "Ja, in een eigen bedrijf/praktijk"          README ""
            ,Meewerk (3) "Ja, in een bedrijf/praktijk van mijn partner" README ""
            ,Anders  (4) "Ja, andere betaalde werkzaamheden"          README ""
            ,Vrijwil (5) "Ja, vrijwilligerswerk"                      README ""
            ,Nee     (6) "Nee"                                        README ""
         ), NORF
```

```
      TochWerk.ASK
```

> back to Table of Contents

Survey Coordination in Blaise 5: A Case Study

Kenneth Rosbach, United States Department of Agriculture – NASS

1. Introduction

The National Agricultural Statistics Service (NASS) conducts two studies for agricultural producers involved in the bee and honey industry. The first is an annual survey that collects data on honey production, beehive inventory, and sales and economic data. After colony collapse disorder became an issue in the US, NASS started conducting a Quarterly Colony Loss survey collecting data on hive inventory, loss, and the reasons for loss.

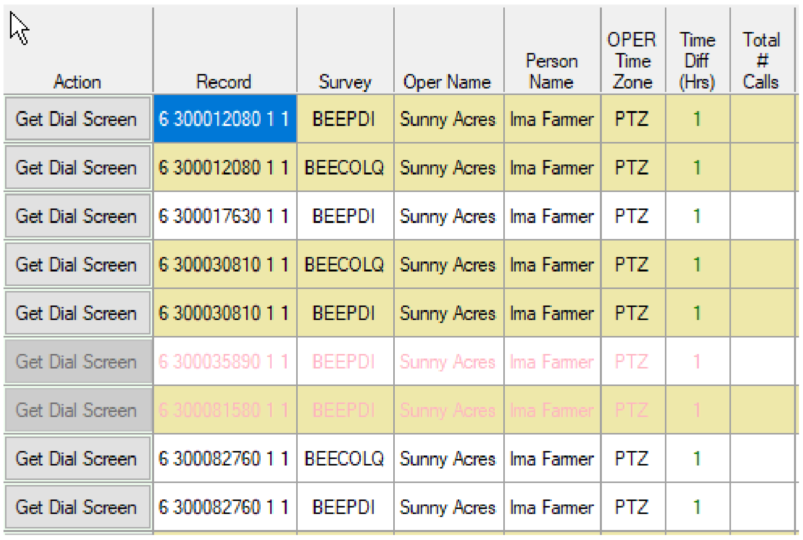
The data collection efforts for these two studies overlap in January. The annual survey typically has a sample of approximately 9,000 beekeepers, and the quarterly survey has a sample of approximately 3,500. As many as 1,500 to 2,500 potential respondents are sampled for both surveys.

2. Background

To reduce the burden on respondents sampled for multiple surveys, NASS prefers to complete all surveys for a respondent in a single contact. Historically, this required having a field enumerator collect data for all surveys on paper. This process required a significant amount of time and money, and it prevented us from being able to effectively use our call center enumerators, as they exclusively call surveys in CATI. With Blaise 4, we could only call on one survey at a time, and switching from one survey to the other was an arduous task. We then began to consider options to simplify this process for our call center enumerators.

Efforts to coordinate this overlap have gone through a few iterations. First, we would call one survey in Blaise 4, then collect the other survey on paper immediately after. Our second approach was to use a Windows program to list respondents in such a way that matching respondents were listed together. See Figure 1.

Figure 1. Original Interface for Matched Surveys



Action	Record	Survey	Oper Name	Person Name	OPER Time Zone	Time Diff (Hrs)	Total # Calls
Get Dial Screen	6 300012080 1 1	BEEPDI	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300012080 1 1	BEECOLQ	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300017630 1 1	BEEPDI	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300030810 1 1	BEECOLQ	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300030810 1 1	BEEPDI	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300035890 1 1	BEEPDI	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300081580 1 1	BEEPDI	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300082760 1 1	BEECOLQ	Sunny Acres	Ima Farmer	PTZ	1	
Get Dial Screen	6 300082760 1 1	BEEPDI	Sunny Acres	Ima Farmer	PTZ	1	

> back to Table of Contents

An enumerator would click on one line to launch the first survey in Blaise 4. Upon completion, they would route back to the listing page, and they could click on the next line to launch the matched survey in Blaise 4. Color coding was used to distinguish between records that had been completed and records remaining eligible for contact. This created a fair amount of overhead between surveys, both in terms of interview time and programming.

The first several questions of an interview contain several administrative items. We ask who we're talking to, verify contact information and business status, and complete other administrative items. While this was the best option we knew about at the time, the interview flow in switching between surveys was a concern.

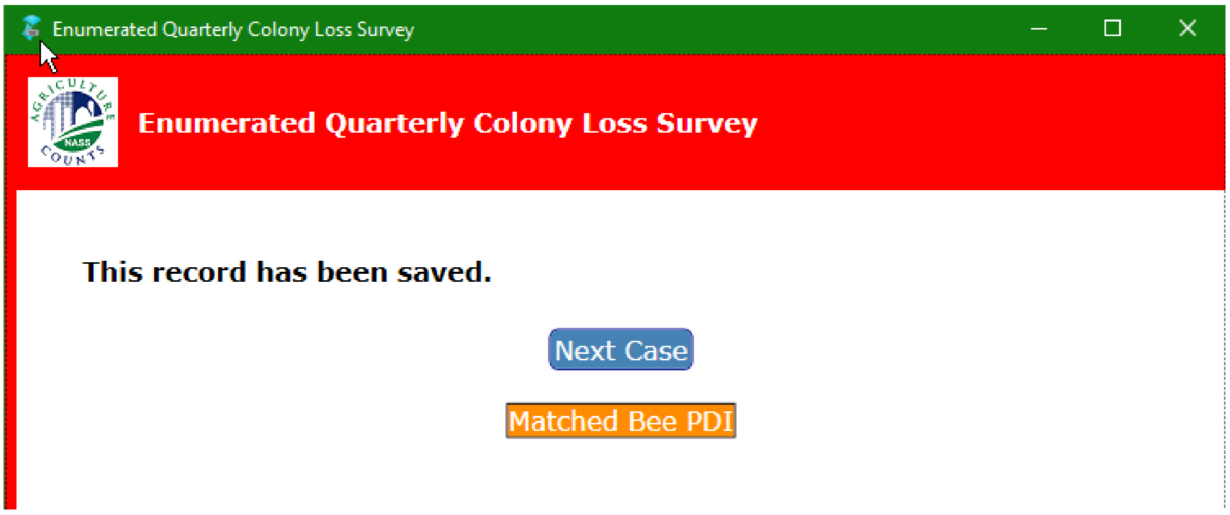
3. Where We Are Now

3.1 Interview Flow

Blaise 5 offers us the ability to launch another Blaise survey using controls in the Blaise Resource Database (blrd), or layout file. When we learned of this feature, we thought this would be a good option for us to improve the flow of a telephone interview when conducting multiple surveys.

We added a button to our receipt page at the end of an interview. See Figure 2. If the respondent was selected for both surveys, the button will be visible, and the enumerator will have the option to click into the matched Bee and Honey survey or select the next form in the current survey. Selecting "Matched Bee PDI" will start that interview at the dial screen.

Figure 2. Receipt Page with a Matched Survey


Enumerated Quarterly Colony Loss Survey
Enumerated Quarterly Colony Loss Survey
Next Case
Matched Bee PDI

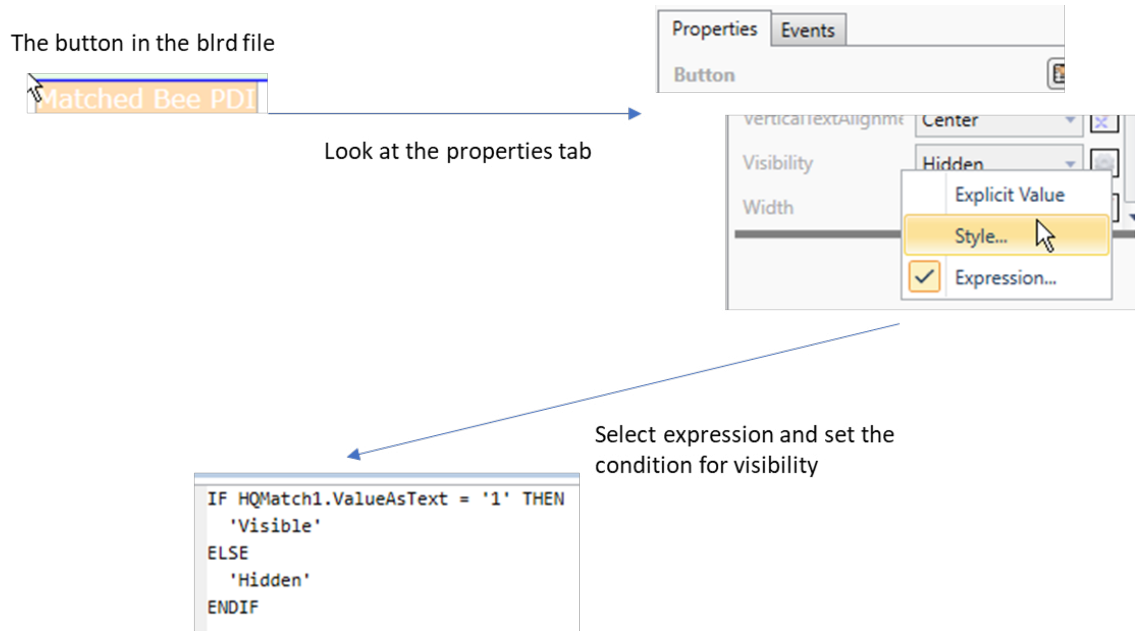
This record has been saved.

3.2 Nuts and Bolts—How We Make it Happen

To implement this button, we start in the Resource Editor and add the button. Our first consideration is that we need to hide the button if there is not a match. To do this, we use Manipula prior to the survey to code an indicator field in the database when a match across surveys is identified. Then in the blrd file, we reference the value of that field in an expression for the visibility property of the button. See Figure 3.

> back to Table of Contents

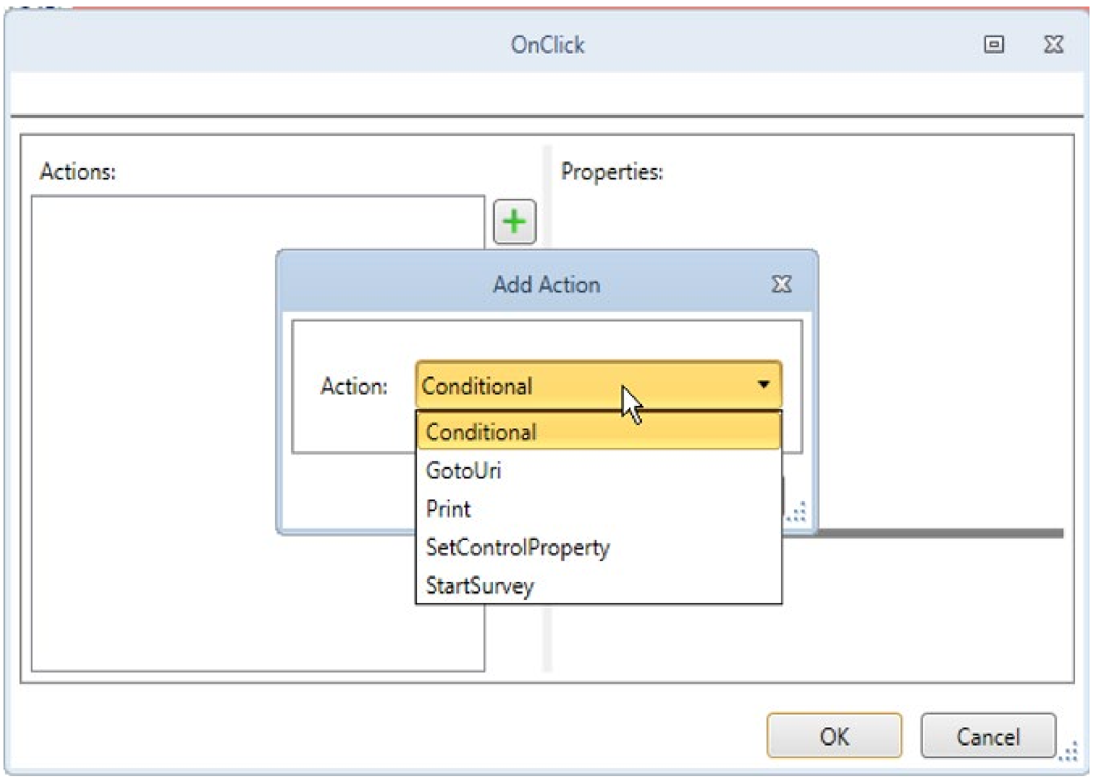
Figure 3. Setting Visibility Condition



```
IF HQMatch1.ValueAsText = '1' THEN
  'Visible'
ELSE
  'Hidden'
ENDIF
```

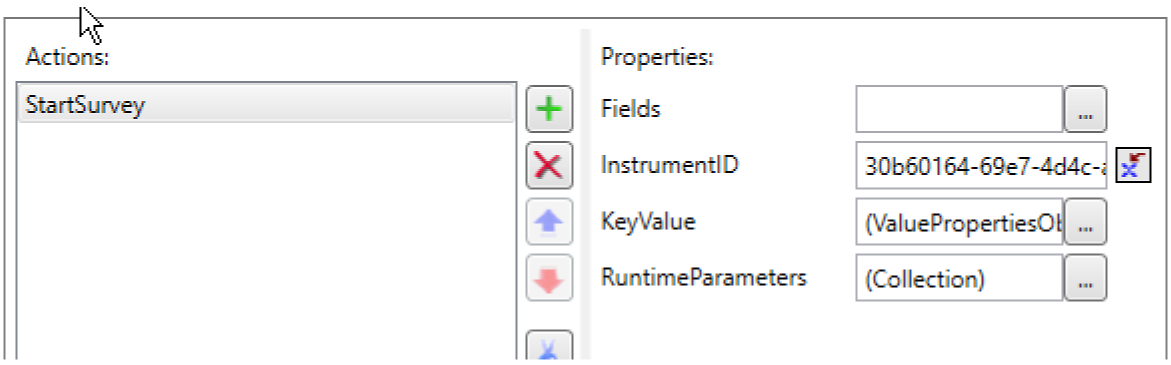
Now that we know the button will only appear when there is a match, the next task is to figure out how to launch the new survey when the enumerator clicks the button. In Blaise 5, this is fairly simple. In the blrd, we navigate to go to the events tab, go to the OnClick event, and select StartSurvey. See Figure 4.

Figure 4. Starting a Survey from Event Settings



> back to Table of Contents

Figure 5. Event Options

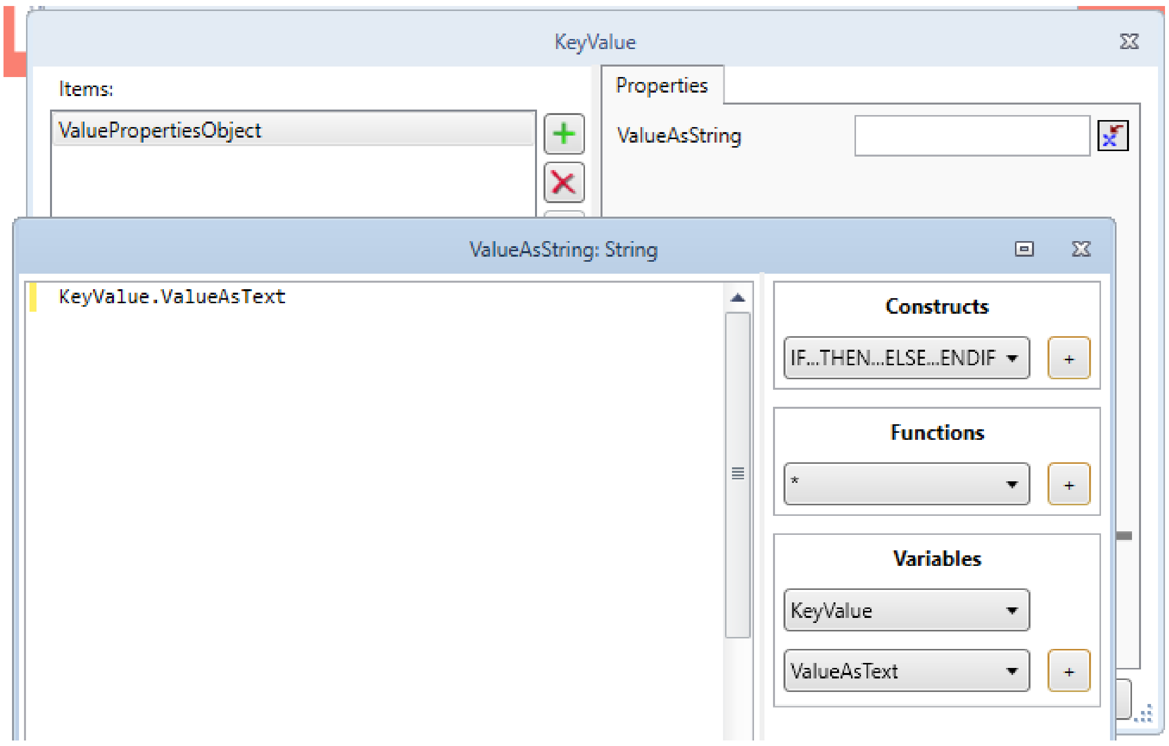


Fields	
InstrumentID	30b60164-69e7-4d4c-
KeyValue	(ValuePropertiesOb
RuntimeParameters	(Collection)

Once you have selected the StartSurvey action, several options will appear in the properties window. The most straightforward is the instrument ID. You can use an explicit value or an expression. For now, we use an explicit value and copy the GUID right into the blrd. See Figure 5. While this approach is easy to set up, it lacks the versatility required for ready use in other coordination efforts.

Next, we want to make sure that the record we're bringing up in the destination survey is the same person as in this survey. NASS uses the same four field primary key in all our surveys, so we know the key value of a record in this survey will be the same as the key value in the destination survey. So, we can click on KeyValue. Then click the green plus to add ValuePropertiesObject. This gives us a properties window, where we choose an expression. In the variables drop-down menu, we select KeyValue. In the option box below, we select ValueAsText. Click the plus button to generate the expression. See Figure 6.

Figure 6. Setting the Key Value

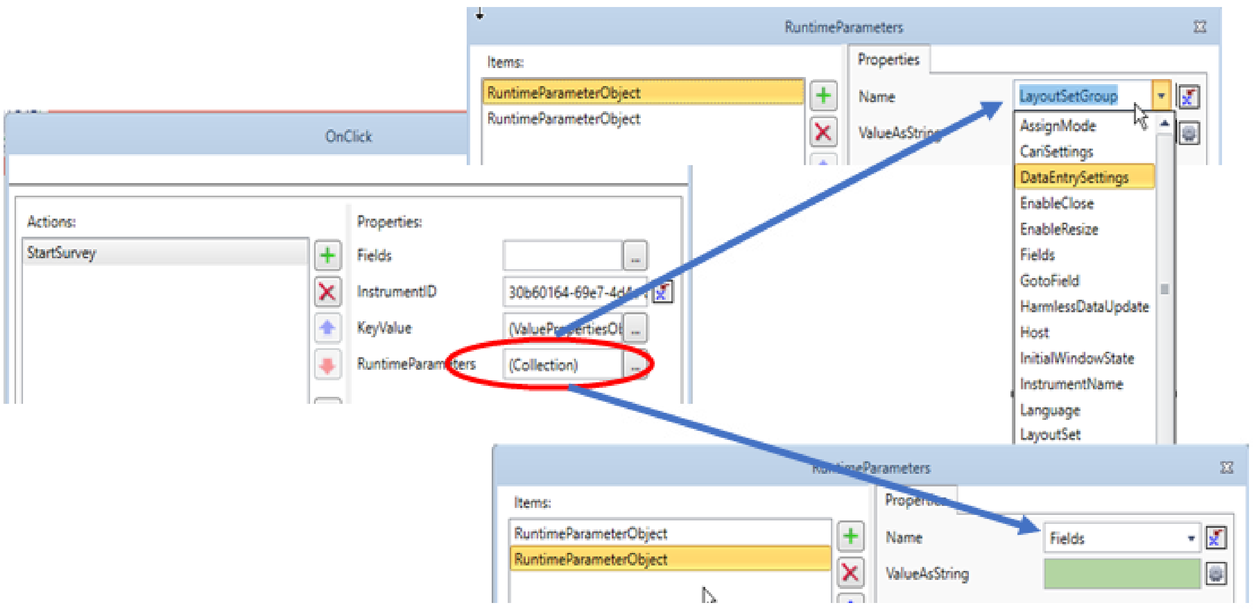


```
KeyValue.ValueAsText
```

> back to Table of Contents

Then we can use the RunTimeParameters field to set some parameters for the new form we're launching. Click the green plus to add a RuntimeParameterObject, then select what you need from the drop-down menu. In our case, we define a layout set and several field values. We then use an expression to define those values. See Figure 7.

Figure 7. Setting Runtime Parameters



RuntimeParameters

Items:

RuntimeParameterObject

RuntimeParameterObject

Properties

Name: LayoutSetGroup

ValueAsString

OnClick

Actions: StartSurvey

Properties:

Fields

InstrumentID: 30b60164-69e7-4d4

KeyValue: (ValuePropertiesOb

RuntimeParameters: (Collection)

RuntimeParameters

Items:

RuntimeParameterObject

RuntimeParameterObject

Properties

Name: Fields

ValueAsString

In the end, we have a system that allows an enumerator collecting data on the Colony Loss Survey to jump right into the annual Bee and Honey survey in cases where there is a match. While this is an upgrade from what we had in Blaise 4, we're still working on improvements.

4. Where We Want to Go

There are several features we are trying to figure out for future iterations of these surveys. One possibility would be trying to build a component that will allow us to determine the GUID of the target survey at run time. At NASS, we like to have all surveys share a blrd file so that we have a consistent flow between surveys. This means all surveys have the same receipt page, which means that we can only coordinate one way since we are hardcoding the GUID in the layout window. When we figure out how to calculate that at run time, we'll be able to coordinate both ways.

The next feature we're considering is how to jump into the second survey where we want to. Currently, the second survey opens to the dial menu, and the enumerator is forced to complete several introductory/administrative questions, including verifying respondent names and contact information, before getting into the main content of the study. Since the first survey has already asked those administrative/introductory questions, ideally, we'd like to copy that data forward to the second survey and open that instrument to the first survey-specific question. We think this can be done using session data, but we haven't yet figured out how to implement it.

We've also recently become aware that Blaise 5 offers a feature known as launcher and topic instruments. When time permits, we will investigate this tool as an alternative option to handle toggling between surveys.

> back to Table of Contents

Ultimately, what we hope to achieve is an experience where an enumerator can collect data for more than one study, but to make it easy for the enumerators to have respondents feel like they are going through only one survey. While we have demonstrated we can add some functionality in this direction in a specific case, our goal is to make the process more generic so we can readily apply it to other combinations of studies.

> back to Table of Contents

User Experience of the Blaise 5.12 Version Upgrade at Statistics Finland

Petri Godenhjelm, Pyry Keinonen, Statistics Finland

This article discusses Statistics Finland's transition to the Blaise 5.12 version. The transition had to consider the requirements of Statistics Finland's current production environment, the measures resulting from upgrading the surveys and standard layout (resource database), and the timing of the version updates of the survey installation packages. The timing of the update was perceived as a challenge, especially updating the work environment of the interviewers during the specified short period to avoid long interruptions in the interview work.

In addition, the resource database was reconfigured to work with the new rendering options, such as CSS (Cascading Style Sheets) grids to ensure a good user experience for both respondents and interviewers. In the update of the resource database, the key changes in relation to the standard resource database of the new Blaise 5.12 version were compared, and the necessary updates to the resource database of Statistics Finland were carried out.

In connection with the update, it was also researched whether to switch to using the Blaise DEP (Data Entry Program) application in CAPI (Computer-Assisted Personal Interviewing) mode and to test the Blaise 5 CATI (Computer-Assisted Telephone Interviewing) system. The goal was to ensure the operation of the user interface and that the survey package generated without pages in the DEP application. Regarding the CATI system, the transition from Blaise 4 to Blaise 5 was studied using a test survey. Based on the results, Statistics Finland's first Blaise 5 CATI production environment concept was created.

1. Background

At Statistics Finland, Blaise 5 has been used since 2017 as the main household data collection tool. The overall management of the mixed-mode data collection is implemented with the Ruuti system, produced by Statistics Finland itself. Together, these systems form Statistics Finland's main data collection system for household surveys. In business surveys, the in-house production system XCola is mainly used.

1.1 Ruuti System Characteristics

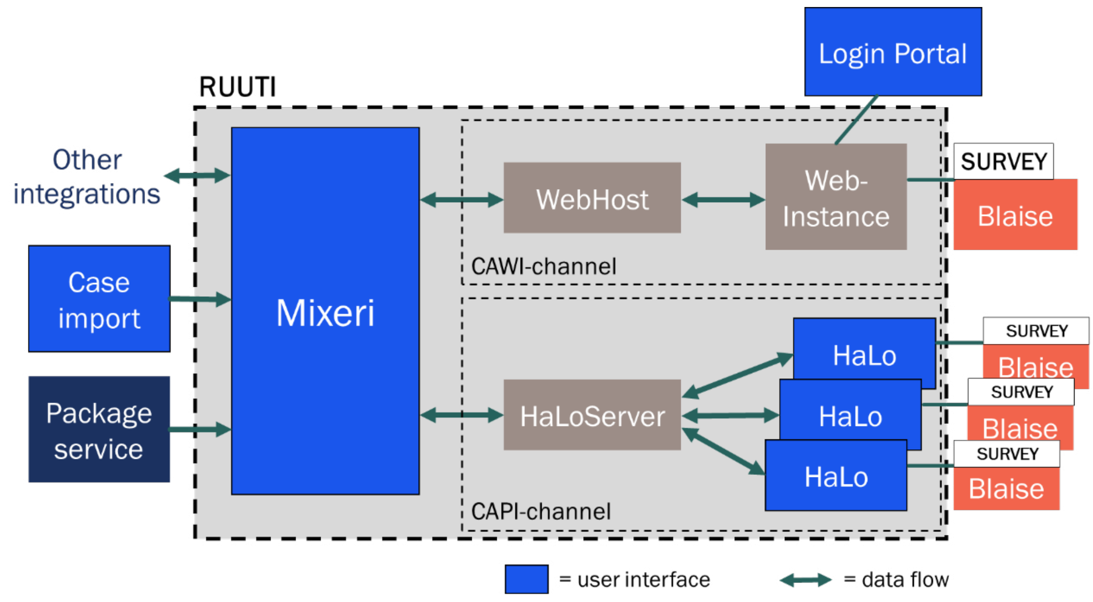
Ruuti is a tool for the daily operation of mixed-mode data collections. Ruuti understands data collection; collection rounds and cycles; two possible collection modes (online answering and interviewing), used together or separately; interviewers and their administration in relation to the interview mode; case/target; persons; and the persons' contact information, addresses, telephone numbers, and emails.

Case and related persons can be taken to Ruuti not only before the start of the collection round/period, but also during it. The characteristics of the case or target also include any prefilled information intended for the survey. When cases and persons are already in Ruuti, their information can be updated if needed.

Ruuti has features such as naming authors for collection periods, distributing items to authors, recording contacts and comments during collection, and producing mass communication (letter/text message/email) mailing lists.

> back to Table of Contents

Figure 1. Ruuti System Overview



Ruuti automatically deduces the case's status based on what has been done to the case during the collection period up to that point.

Ruuti distributes all generated information automatically within the system wherever information is needed.

Ruuti can ask the survey installation package service for the survey installation package versions available for each data collection. After the user has selected the installation package, Ruuti distributes the Blaise installation package to the necessary places, uses Blaise to install the survey ready for use, and commands Blaise to open the survey for answering. After the survey closes, Ruuti receives the information generated in the response.

Ruuti talks with the Login Portal to enable web responders to access the survey.

Within the limits of user rights, Ruuti offers the survey data output via application programming interface as it is, regardless of the content.

Ruuti is not responsible for the content and functionality of the forms, it is not a long-term storage place for collected data or a tool for planning data collection, nor is it a tool for developing data collection.

1.2 Blaise Version Upgrade Challenges

A part of the characteristics of Ruuti is that interviewers have their own case management application called HaLo. HaLo provides case management tools for interviewers and handles the Blaise survey package management. It also provides offline interview capability and handles the data from all the installed surveys.

In addition to the HaLo installation, the interviewers currently have a Blaise server installation installed on their workstations. This is because Blaise DEP has not been used in Statistics Finland, and the surveys have been opened in interviewer mode (CAPI) with a local web browser. Such a solution was decided on in 2017 because the user interface built for interviewers worked in the

> back to Table of Contents

browser without problems with the previous version of Blaise, while the dimensions of the page base and some of the controls did not work in the user interface while opened using DEP.

Because of this choice, upgrading Blaise for the interviewers has required not only the HaLo application update, but also a new Blaise installation, instead of just sending a new DEP solution in the distribution. When installing a new Blaise version, it has been necessary to carefully consider whether the new version is backward compatible with the installation packages built for the previous version. As the production is running all the time, it is either necessary to schedule the updates to the turning points of the key data collection rounds around the turn of the year, or to carry out the updates during an unwanted production break.

Due to the heavy update process of introducing new Blaise versions into use in Statistics Finland, the update interval is once or twice a year and mostly for bug fix upgrades, if possible. Also, new Blaise versions cannot be put into production flexibly and with short notice. As a result, the introduction of the new Blaise version involves long-term testing and waiting until the biggest bugs from the latest released versions have already been ironed out and the operation is relatively reliable.

2. Blaise 5.12 Upgrade

In Statistics Finland, the current production version in use is Blaise 5.8. In the selection of the production version, testing results have typically been used, and the opinion of Statistics Netherlands (CBS) has been sought on the production suitability of the version. A couple of previous production versions have been selected with these criteria. Major version upgrades are carried out at Statistics Finland every couple of years. Smaller updates are pushed to production more often if necessary. In this case, a “major version update” means an increase in the middle version number.

2.1 Testing Process, Briefly

Normally, a new Blaise version is initially tested with an in-house test survey, which contains all the most common answer types and layout definitions used in Statistics Finland. In this test phase, we will test that the survey works in both modes (CAWI and CAPI); that there are no problems with generating the layout in browsers or DEP; and that all the controls used in user interface, custom solutions, and different screen dimensions work within the layout. It is important to make sure that general usability is good and accessibility functions are working as they should. Also, a few common mobile devices are used in tests.

After a successful test in the first phase, the second phase consists of a few of our most complex surveys being upgraded with the new Blaise version and tested for bugs, especially if some unique or complex solutions have been used in a survey, such as household structure or custom features, which are not found in our test survey.

In the third phase, we create a Ruuti system compatible survey installation package and test the survey in the Ruuti test environment. Before this test is possible, we usually must upgrade Ruuti API libraries with the new ones that come with the new Blaise version package. After these tests, the new Blaise version is fit for production use in Statistics Finland.

2.2 Upgrading Resource Database

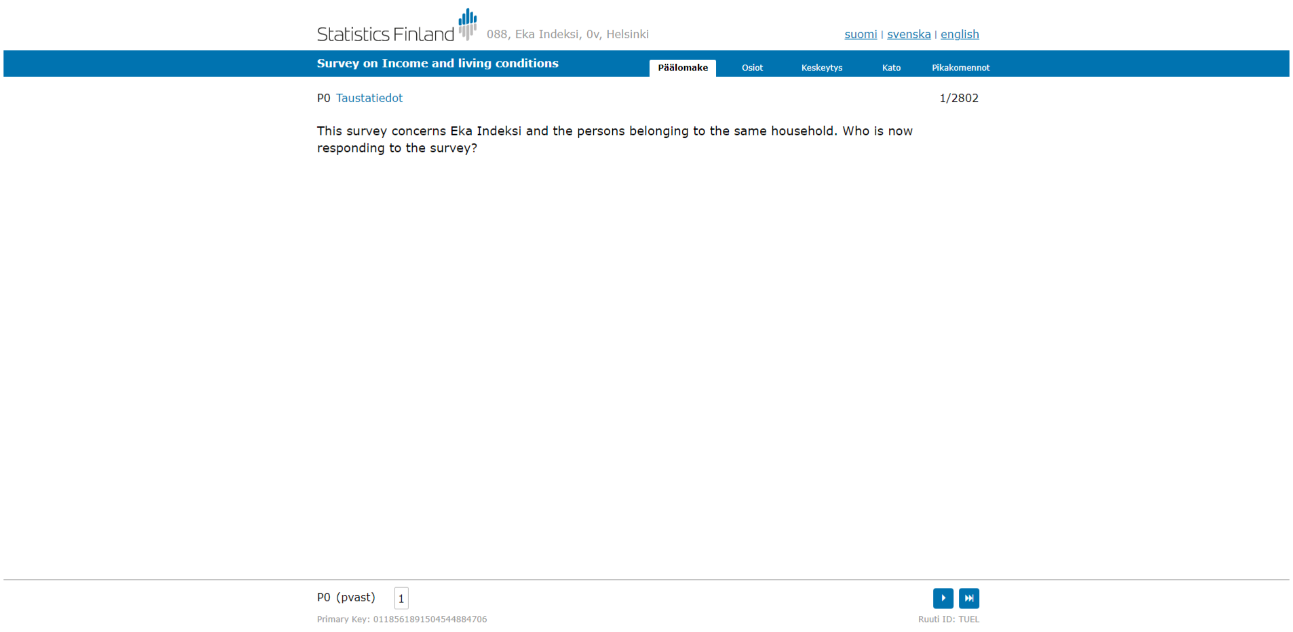
Before the testing process, it was assumed that a lot of work needs to be done with the resource database. The biggest fear was that the new rendering options, such as CSS, could cause changes in

> back to Table of Contents

the customized resource database and even force some kind of massive refactoring, or that there would be some other layout problems.

In this case, the biggest issues were with the CAPI mode layout. It is constructed in a way that the interviewer may answer an enumeration type of question by choosing an alternative or addressing an option number in the field. See Figure 2.

Figure 2. No Enumeration Buttons Visible While Using Browser without CSS Grids Options


Statistics Finland
088, Eka Indeksi, 0v, Helsinki
suomi | svenska | english
Survey on Income and living conditions
Päälomake
Osiot
Keskeytys
Kato
Pikakomennot
P0 Taustatiedot
1/2802
Ruuti ID: TUEL

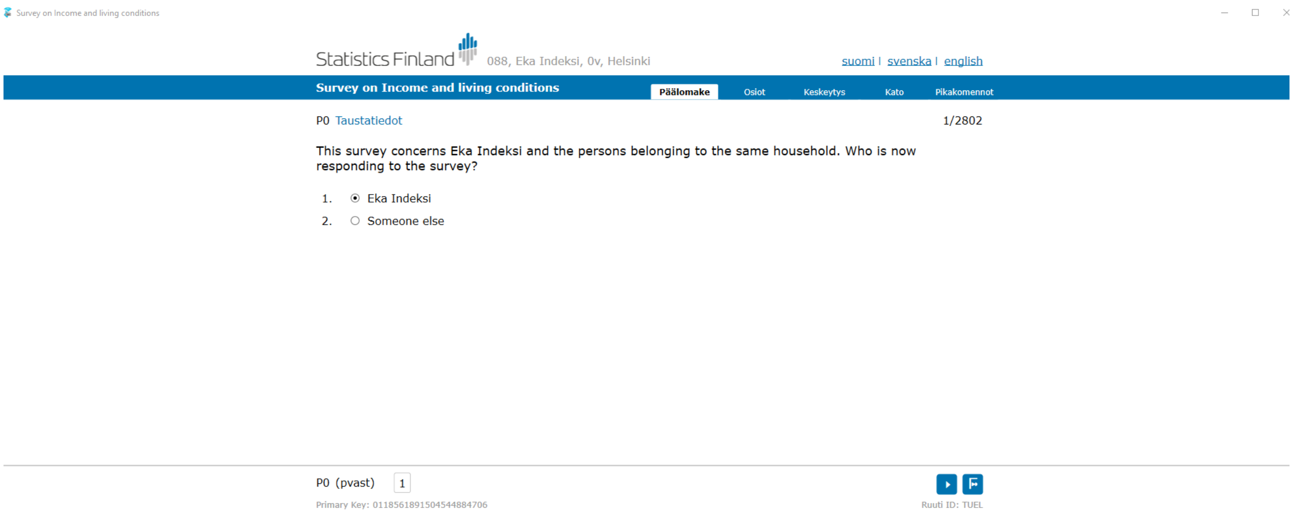
This survey concerns Eka Indeksi and the persons belonging to the same household. Who is now responding to the survey?

P0 (pvast) 1

Primary Key: 0118561891504544884706

This error was related to the new rendering options. Viewing the survey in a browser without using the "Use CSS grids instead of size tree" option in the data entry settings caused the enumeration buttons to vanish totally from the user interface. With these settings in DEP, everything worked as it was supposed to. See Figure 3.

Figure 3. No CSS Grids in Use Did Not Affect DEP


Survey on Income and living conditions
Statistics Finland
088, Eka Indeksi, 0v, Helsinki
suomi | svenska | english
Survey on Income and living conditions
Päälomake
Osiot
Keskeytys
Kato
Pikakomennot
P0 Taustatiedot
1/2802
Ruuti ID: TUEL

This survey concerns Eka Indeksi and the persons belonging to the same household. Who is now responding to the survey?

1. [✓ Eka Indeksi]
2. Someone else

P0 (pvast) 1

Primary Key: 0118561891504544884706

When the CSS grids option was checked, you could see all the enumeration buttons as they should be in the user interface when using a browser to view the survey.

> back to Table of Contents

In general, most of the fears did not come true, and the update of the resource database was successful without major complications.

2.3 Testing DEP

After the successful resource database update, it was necessary to study how DEP would work with our updated resource database and layout settings. It was decided in the spring that if DEP would work, it would replace the current way of work to ease the installation burden and to help with running multiple versions of Blaise installations on an interviewer's workstation at the same time by running surveys in ThickClient mode.

The testing itself was carried out by using one of our heaviest surveys, "Survey on Income and Living Conditions." In the test, it was necessary to successfully set up a standalone ThickClient survey and run the survey package without generated pages. It was necessary to have evidence that there is no risk the survey would not work in practice. It is not efficient to distribute installation packages with static pages when their size could be gigabytes. In the Ruuti system, there are multiple survey installation packages per survey per year assigned in use.

In the actual testing, the results were promising. There were no problems with running the survey, and the quick conclusion was that it will be implemented into production in autumn 2023.

These changes will affect the Ruuti system and the HaLo application, also.

3. CATI

Blaise 4 is still used in two enterprise data collections in Statistics Finland. Also, both are conducted in CATI mode, which we are not currently using in any other survey or in Ruuti data collections. A part of the Blaise 5.12 update plan was to create a clear picture of the Blaise 5 CATI system and find a solution for our old Blaise 4 CATI survey model. The problems with the Transport Layer Security protocol 1.2 compatibility issue with Blaise 4 was considered one of the reasons to get rid of Blaise 4. The first plan was to install and test Blaise's own sample survey to get used to the phases of setting up a simple CATI survey in Blaise 5.12. The selected sample was a traditional CATI survey called "Health Survey." The survey was selected because it is simple and reflects the need for use well.

In the installation phase, a provided installation readme-documentation was followed to the letter. These steps included how to build and install the multipackage, create a daybatch, add some users, and run the survey.

While running the survey as an interviewer role user, there was a notification message in the Blaise survey window after login that stated, "Cannot select case because no valid daybatch file was found for 12.7.2023." Despite trying to redo all the steps, it was a dead end finding a solution without help. So, this message escalated into a couple of months of communication with CBS. With their help, the error was found in the end by studying Windows application logs and using a special installation package that CBS provided for the troubleshooting.

After receiving a bug fix package from CBS for test usage, the CATI system could finally be tested as it was meant to be originally.

> back to Table of Contents

4. Conclusions and Next Steps

During October and November, the Blaise 5.12 update will be performed. The update requires upgrading active production surveys with the new Blaise version and packaging the Blaise ThickClient solution for distribution to interviewers. In addition, the Ruuti system must be updated as needed, so that the changed way of using the Blaise system does not cause problems in the operation between the systems. In the first distribution package, active production survey installation packages are also delivered to the interviewers. In the future, these installation packages will be distributed normally via the Ruuti system distribution. The amount of distributed survey installation packages in November is around 30 pieces.

At the time this paper was written, the next phase for CATI testing was to introduce and study the new Blaise 5 CATI system with a representative from the interviewer work organizing role. This step will provide specifications and create the conditions for the CATI Blaise 4 to Blaise 5 upgrade.

The next step concerning the CATI update is to create the first actual Blaise 5 CATI survey version from one of our Blaise 4 business surveys during the early weeks of October. In this usage case is the decision that the Blaise 5 CATI will be set up on its own dedicated server environment and interviewers will use a browser to log in to the system. Hopefully, we will have our AD users successfully synchronized to a server manager before going into production.

The experience of upgrading Blaise 5 versions to the next version has improved over the past six years. In the past, the version updates always felt like a big effort from the whole organization, and it seemed that there was always some troublesome bug in every Blaise version. Over the past few years, the experience has greatly improved, and the incidence of compatibility issues has decreased. In addition, the service desk model CBS has used for a few years now has been very customer friendly. The overall experience of handling cases with CBS has grown better and faster.

In the original abstract, there was a mention of studying Blaise 5.12 paradata viewer and testing tools as a part of this paper, but due to time constraints, these subjects are not included. If possible, these subjects will be referred to in the actual presentation in IBUC 2023.

> back to Table of Contents

Blaise 5 Journey: A Case Study of a CAWI/CATI Survey and the End-to-End Processes and Systems Used

Angela Belo, Alessio Fiacco, and Colin Miceli, National Centre for Social Research (NatCen)

We are on a journey to migrate to Blaise 5 for all our survey work. Now that we have completed a few CAWI and CAWI/CATI surveys on this new platform, we would like to share our experiences: what we have developed, what we have learned, and the challenges and opportunities along the way, as well as what is yet to come. First, we would like to provide a brief overview and background, then delve into various aspects of the case study based on our Technical Education survey, provide some details into the redesign of a key supporting system for managing sample and other functionality, and wrap up with what lies ahead and some of the challenges we anticipate.

1. Background and Overview

1.1 Why Blaise 5? What's Important to NatCen

The National Centre for Social Research (NatCen) has been operating two software systems for its data collection services for more than 10 years: Blaise 4.8 primarily for face-to-face interviewing and Unicom Intelligence/IBM Dimensions (UI) primarily for telephone and web surveys. It was recognized that their functionality is limited and soon they will both become unsupported.

NatCen's vision for 2025 includes web and mobile first, and multimode data collection services. More and more, it is crucial to have the flexibility needed to expand the services NatCen can offer, and to be ready for any substantive shift from face-to-face to other modes and multimode data collection.

Having a fully functional, future-proofed, multimode data collection survey platform running on up-to-date server architecture with a reduced reliance on legacy solutions and limited programming resources is critical to the business plan.

A full market scoping exercise was commissioned in 2019, and external consultants were appointed to assist with the selection of future-proofed software systems that can provide a lean and efficient operating model for multimode data collection. Blaise 5 emerged as the most suitable solution.

The Blaise 5 project was initiated in June 2020 to transform NatCen's survey platform and related procedures, as well as upgrade or replace associated legacy support systems. This will ensure NatCen is able to provide state of the art multimode solutions for data collection and remain competitive in its offering.

1.2 Overview of Our Business

NatCen is a registered charity and is the largest independent and not-for-profit social research organization in the UK, with an excellent track record in surveys and policy research.

We work with many different clients, including various government departments, academic institutions, and others, responding to tenders to gain project work. As a result, there is a wide variety of survey projects that we run, catering to different client requirements.

1.3 Overview of Our Technical Environments

Before any work could begin, we needed to set up our server environments.

> back to Table of Contents

The technical server infrastructure is comprised of the following servers, for production:

- 3 web servers, with a load balancer to distribute the load across them
- 2 Blaise application servers
- 1 database server
- 1 session server

We have the following environments for various purposes, with a similar configuration but geared to less capacity than production.

1.3.1 Development/Demo Environment

We have one environment dedicated to development and unit testing activities. We also use it to experiment with new ideas and proof of concept for various survey designs, as well as to test new Blaise releases before applying them more broadly to all other environments.

1.3.2 Pre-Prod Environment

We use this environment for testing new releases of the Survey Control System (SCS), along with Blaise 5. If new Blaise 5 software releases are needed for certain SCS features, then the upgraded versions are applied here.

1.3.3 UAT/Staging Environment

This environment is a mirror image of the Production environment, running the same software versions of Blaise 5 and the SCS systems. We use this environment for working on live projects during the development and testing of surveys prior to launch.

1.3.4 Production Environment

This environment is for running live surveys and collecting data from respondents/interviewers in all modes.

1.4 Templates Redesign

In the early stages of the project, we created some Blaise 5 question templates for our web and CATI projects. We discovered quickly that learning and working with the resource database has a reasonably steep learning curve.

We have been using Unicom Intelligence at NatCen for our web and CATI surveys since 2015 and had some existing question templates. This meant we had a starting point for the design and functionality of what we wanted for web. It also meant programmers and researchers had certain expectations on how the questionnaires would look and function.

The Blaise 5 templates team is headed up by a researcher from our methodology department and includes a technical resource, as well. Testing the question templates has been a combination of internal testing and testing with real people using cognitive interviewing techniques where a researcher leads a respondent through a test interview and asks for feedback.

Some initial requirements were:

- the templates should work on small and large screens
- they should work without using a mouse; this meant we would test the large screen template using a keyboard only

> back to Table of Contents

- they should have a hidden DK\RF feature
- they should use a concertina\collapsible grids
- they should use the NatCen corporate brand colors
- they need to handle soft and hard checks
- all data needs to be saved, even if the browser is closed
- they should allow the collection of paradata
- they should make it easy to program exclusive codes
- they should include a help feature for some questions to provide additional instructions for respondents

We did a review of available templates. We chose two of the resource databases shipped with Blaise in addition to ONS templates that they provided to us.

Looking at these three resource databases gave us a starting point and some ideas. We created a resource database with a small and large layout set in order to minimize maintenance.

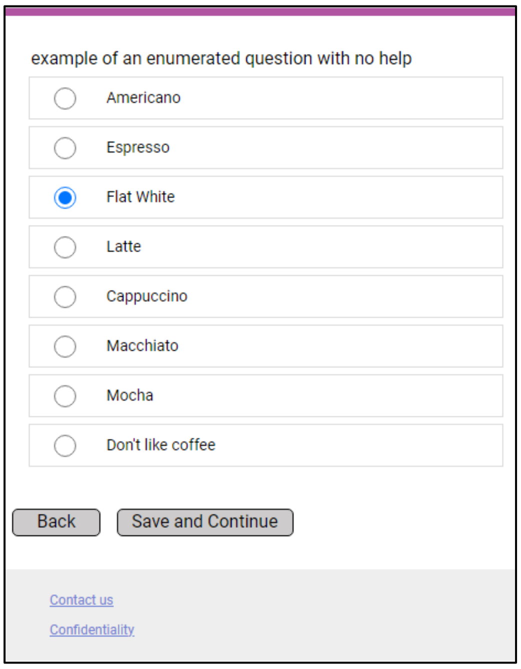
The small layout set is automatically selected if the screen size is less than 1024 pixels wide.

1.4.1 Designs for Categorical Questions

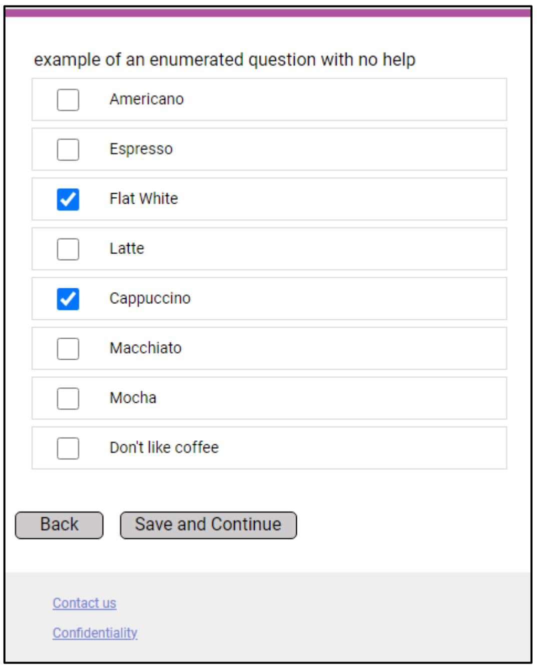
The categorical questions used a large input control, and the design requirement included changing the background color of the selected category.

In this case, all the category text is clickable, so it works similar to a button. In an earlier design, the background color of the selected category changed to show that it had been selected. This caused the screen to repaint and change the focus on questions (we called it the 'wibble'). With long category lists, this looked odd, and we abandoned this approach. In the current version of Blaise, which uses CSS, this is no longer an issue, so we may return to the original design.

Figure 1. Examples of Single and Multiple Category Selection Question Template, with No Help Information, for Small Screens


example of an enumerated question with no help

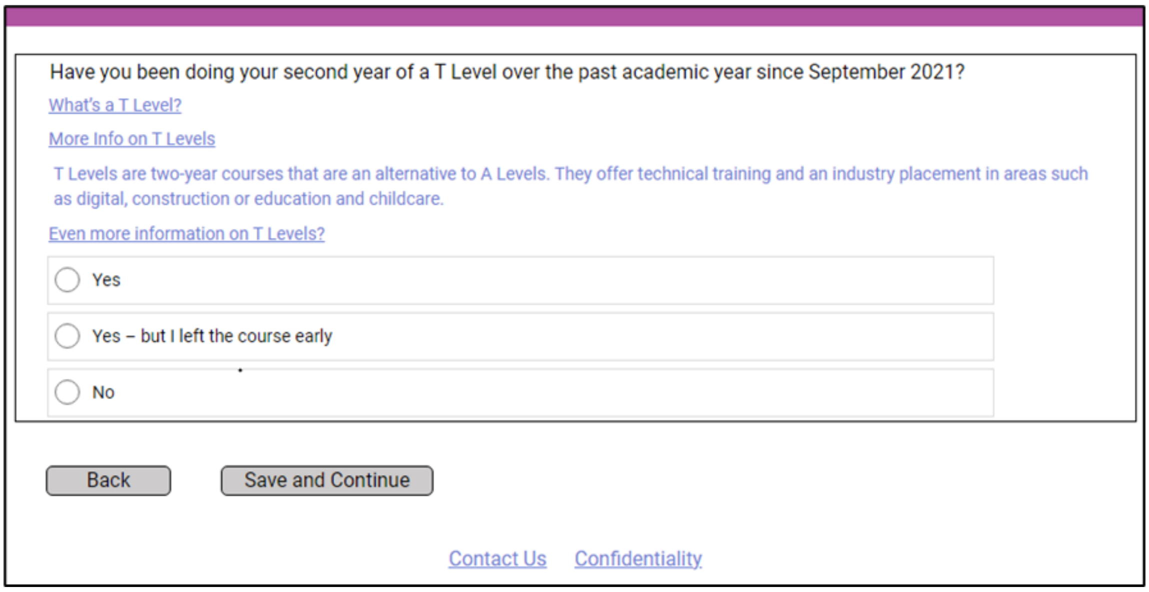
- ○ Americano
- ○ Espresso
- ● Flat White
- ○ Latte
- ○ Cappuccino
- ○ Macchiato
- ○ Mocha
- ○ Don't like coffee


example of an enumerated question with no help

- ☐ Americano
- ☐ Espresso
- ✓ Flat White
- ☐ Latte
- ✓ Cappuccino
- ☐ Macchiato
- ☐ Mocha
- ☐ Don't like coffee

> back to Table of Contents

Figure 2. Example of Categorical Question Template, with Expanded Help Information


Have you been doing your second year of a T Level over the past academic year since September 2021?

What's a T Level?

More Info on T Levels

T Levels are two-year courses that are an alternative to A Levels. They offer technical training and an industry placement in areas such as digital, construction or education and childcare.

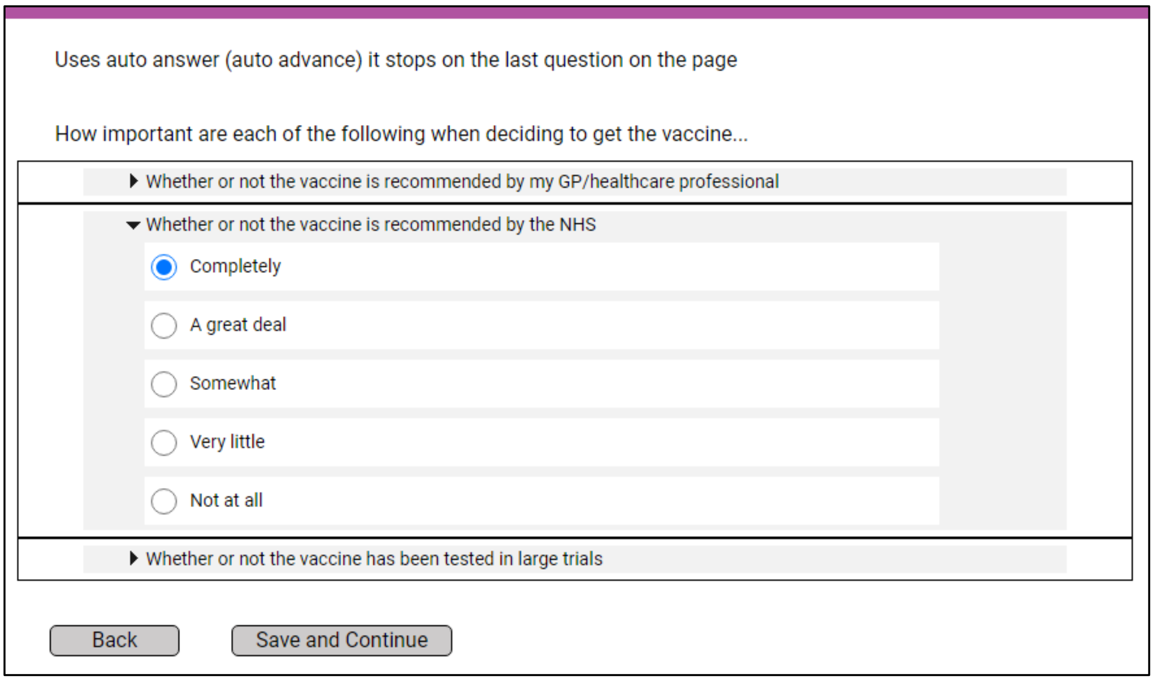
Even more information on T Levels?

- ○ Yes
- ○ Yes – but I left the course early
- ○ No

1.4.2 Designs for Grid Questions

We are using a collapsible grid instead of a traditional row and column grid. This works better on small screens and means we can use the same design for large and small screens.

Figure 3. Example of Collapsible Grid Question Template


Uses auto answer (auto advance) it stops on the last question on the page

How important are each of the following when deciding to get the vaccine...

- ▶ Whether or not the vaccine is recommended by my GP/healthcare professional
- ▼ Whether or not the vaccine is recommended by the NHS
  - ● Completely
  - ○ A great deal
  - ○ Somewhat
  - ○ Very little
  - ○ Not at all
- ▶ Whether or not the vaccine has been tested in large trials

> back to Table of Contents

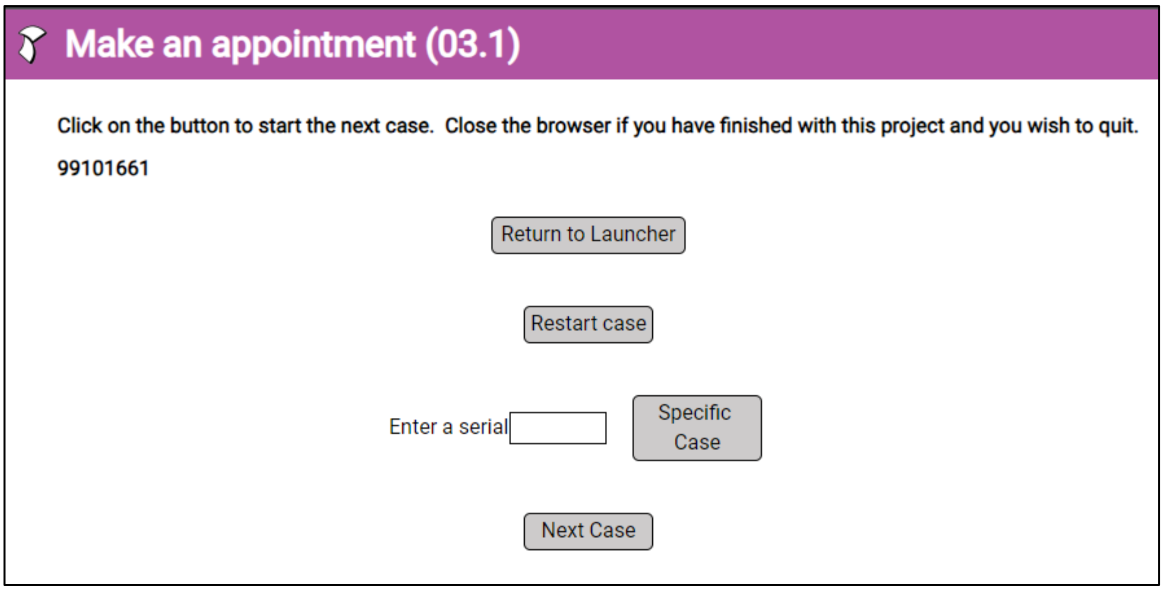
We conducted a number of tests with internal and external users.

For CATI, we wanted to reuse the same templates that we had used for web, but we had to add some additional features.

For the CATI resource database features, we made some information available in parallel blocks. We also added a status bar so interviewers could see the question they were working on. In some instances, we had question texts with different phrasing for different modes.

In the receipt page, we added functionality to be able to restart the previously opened case, start a specific case with a serial number entered, or move on to the next case.

Figure 4. Example of Receipt Page (After Appointment)



Make an appointment (03.1)

Click on the button to start the next case. Close the browser if you have finished with this project and you wish to quit.

99101661

2. Case Study—Technical Education Survey (Tech Ed)

2.1 Background of Tech Ed

Tech Ed is a longitudinal survey of young people enrolled in technical education courses, where we ask the same learners to take part in this study every year during their courses.

This is the first major piece of research into the government's technical education reforms and will help the Department for Education ensure that the new study programs are working as intended. The survey aims to provide insights into which young people are choosing the technical education route and why, what their experience of technical education is, and what short-term outcomes young people achieve.

The fieldwork has a 'web-first sequential mixed-mode' design with two modes: Web (CAWI) and CATI.

All participants were initially invited to complete the survey online. Subsequently, cases still not completed were released to the Telephone Unit in batches (using DayBatch select filters).

The sample consisted of named individuals, with email, postal address, and multiple phone numbers possible for each, (i.e., a landline or mobile number, and in some cases both numbers were available). The sample data in the call scheduler was managed at the case level (the individual).

> back to Table of Contents

The study runs over multiple waves (up to March 2024).

We used Blaise 5 for the following rounds of data collection:

- 2022
  - Wave 2 (W2) Survey (Pilot and Mainstage), sample of 1,100 cases
  - Wave 1 (W1) Survey (Pilot and Mainstage), sample of 23,000 cases
- 2023
  - "Post-course Mainstage" (for those that had just completed the program), sample of 1,300 cases
  - "In-course Mainstage" (for those that were still in the program), sample of 15,000 cases

2.2 Mixed-Mode Instrument Development

2.2.1 2022 Sweep

For the first (Blaise 5) round of data collection in 2022, we converted the code from our current web platform, UI, into Blaise 5. We used Traditional Solution for single-topic projects where the data collected in the treatment surveys (Dial and Appointment) were copied to fields in the main data model.

The Dial instrument was the entry point for the interview. One of the Parallel blocks displayed a selection of fields from the sample block (variables had to use labels).

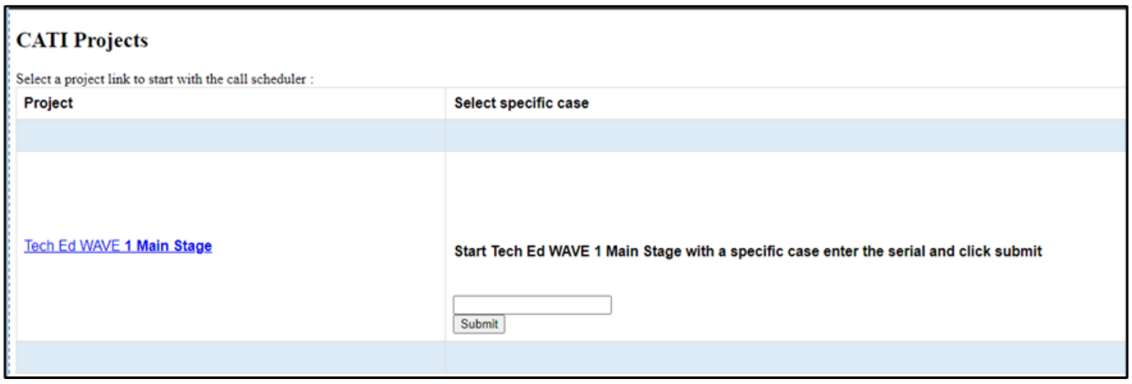
The appointment instrument was used to record an appointment and for interviewers to make remarks, which were copied to the Dial questionnaire. We didn't have a Nonresponse instrument.

The main instrument held the survey data (and allowed other outcomes, such as NoAnswer—Parallel block) and the CATI Admin data.

Using Traditional Solution meant that we did not have a starting page for the CATI interviewer to use (to select a survey from a list), so an HTML page was created to allow interviewers to select a project or a specific serial number within that project.

NOTE: CATI Dashboard in our organization is currently only used by Telephone Unit (TU) Supervisors and not CATI interviewers, so they don't have another way to start specific cases.

Figure 5. HTML Page Used in Traditional Solution without a Portal



CATI Projects

Select a project link to start with the call scheduler :

Project	Select specific case
Tech Ed WAVE 1 Main Stage	Start Tech Ed WAVE 1 Main Stage with a specific case enter the serial and click submit

> back to Table of Contents

2.2.2 2023 Sweep

For the 2023 round of data collection, we used the previous (2022 Sweep) Blaise 5 code as a starting point, but we set up the projects using the multi-topic solution.

The multi-topic approach consists of a series of linked instruments; namely, the Launcher, the Portal (NOTE: we did not use the Multi-Scheduler), an Appointment instrument, a Nonresponse instrument, and a Topic instrument.

The interviewer selects a Topic survey from the Portal and a case is delivered automatically.

2.3 Reasons for Switching to Multi-Topic Solution

Why did we change the approach between rounds of data collection?

Firstly, for this longitudinal project, it was deemed to be preferable to have the CATI Admin data separated from the Topic instrument. The Launcher questionnaire collects the CATI Admin data and includes an array for up to 30 calls (resulting in large file sizes).

Also, we required an entry point that allowed an interviewer to choose a project from a list of projects. In this event, we didn't have any overlap in the fieldwork of the projects in the various waves of Tech Ed, so the Portal only listed one project at the time. However, the Portal was deemed as an important element for CATI surveys.

The long-term aim was to be able to use the same Launcher for all our projects. However, there were different (sample) requirements between waves of Tech Ed (emails sent from the Launcher).

The Nonresponse instrument became necessary to ensure that cases coded as nonresponses in the Launcher/Topic parallel block were registered as nonresponses. (We need to evaluate if this will still be required in future releases of Blaise).

2.4 CATI Launcher for Handing Multiple Numbers and Call Results

The CATI Launcher is the instrument used by the interviewer to make the initial contact with the respondent and it contains (besides the sample block, which mirrors the sample block in the Topic) the call history and contact history.

Blaise gives you a call per case. In the Launcher, we have a dial outcome **per phone number** called. At the end of the dial loop, we pass the call result back to Blaise based on the rules set in the Launcher. This will result in one call.

Our Launcher, potentially, handles a series of up to 30 dials, which will result in one call.

The Launcher was adapted to meet these specific requirements:

- Dial outcomes are recorded for each phone number tried within a case. The telephone number status is based on dial outcomes.
- Prioritized call outcome is computed based on dial outcomes/questions to interviewer, and this is used for interaction with the call scheduler.
- Case-level status and outcomes are derived from call outcomes.

> back to Table of Contents

2.5 Removing Web-Completed Cases and Freephone Opt-Outs from CATI DayBatch

We had to give TU Supervisors a way to create the DayBatch and remove the completed cases (primarily from web) from the DayBatch during the day (NOTE: This was a bug reported to Blaise, which was fixed by the Blaise Team in a later build, but we weren't able to upgrade before fieldwork).

We scheduled a task, which invoked Manipula to create the DayBatch each morning and, after the creation, the completed cases were removed from the DayBatch. Completed cases were removed from the DayBatch at other times during the day as well. This worked OK.

We created a custom "report" on the Dashboard (i.e., a Manipula script that created the DayBatch and removed completed cases from the DayBatch). There were a large number of cases, so we used SETRECORDFILTER to speed up the processing.

Also, we had to invoke Manipula from an intermediary script, as we discovered that the script was timing out after two minutes.

2.6 Monitoring Fieldwork Progress, CATI Dashboard, Custom Reports

In order to monitor the progress of fieldwork, we exported and merged data in Manipula, including key information from the Topic (sample + outcomes) and Launcher (call history) data. (Longer term, we are aiming to use a data mart for reporting.) The exported .csv file was read in as a data model in Excel with a number of pivot tables showing the outcomes by telephone batches.

2.6.1 Dashboard

The TU used the Case Info tab, which shows the number of calls, call results, inclusion in a day batch, appointment details, etc.

One issue we had was that it can sort by number of calls (in the build of Blaise used for Tech Ed), but TU want to be able to filter all cases that meet a specific number of calls.

We also used the Appointment Overview custom report. The latest version, amended by the Blaise Team, includes the Primary key.

2.7 Conclusions

Tech Ed was our first Blaise 5 project, and as such, there was a lot of work involved to set up our first Blaise 5 project.

Further work is required to standardize the sample block and make the CATI Launcher project-agnostic. This will be challenging due to the complex and varied requirements of our projects.

There is continual development. What was done so far is not set in stone. We may revisit the multi-topic approach.

Separation of CATI Admin data from the Topic data works fine for longitudinal studies. We are exploring using an "all-in-one approach" for simpler, cross-sectional surveys, some of which do not have the same requirements as longitudinal studies like Tech Ed (and have no need for a complex Dial questionnaire). They are simpler and more streamlined, with less detail tracking.

New releases of Blaise have taken on board requirements such as the ability to identify hard appointments on Dashboard and filter cases by number of calls, both of which are important requirements for our TU.

> back to Table of Contents

The Blaise Team also took on board a requirement to display the Primary Key on the Appointment Overview report (Dashboard).

2.8 Challenges

One key challenge is that there are different requirements for various projects, making it difficult to standardize the Launcher!

Sample block standardization is also difficult to achieve.

3. Redesign of Sample Management (SCS)

In our current survey platform, we are using a Sample Management system designed and implemented over 10 years ago, which has evolved over that time to handle a variety of changing business requirements. As part of our Blaise 5 journey, it was deemed critical to replace this system with an improved SCS to provide the needed functionality and to extend that functionality for greater efficiencies and streamlined and automated business processes.

One key architectural and design decision was to have a tighter integration between this system and Blaise 5, and to synchronize data between them.

The system is developed in C# and uses the Blaise API for integration.

3.1 Integration with Blaise 5 Using APIs

The key features that the SCS system integrates with Blaise 5 are the following:

- Analyzing Sample definition in Blaise questionnaire and setting up corresponding data definitions in SCS
- Analyzing Administrative and Communications blocks in Blaise questionnaires and setting up corresponding data definition for imported data in SCS
- Loading sample data in SCS and synchronizing this to the Blaise questionnaire
  - Data quality checks are performed on data when it is loaded
  - The system checks for sample updates and synchronizes to Blaise on an hourly basis
  - It is possible to specify the Login Questionnaire, CATI Launcher, and main Topic questionnaire, and all of these will be synchronized from SCS to Blaise
- Automatically importing data from the Blaise questionnaire (Admin and Communications blocks) into SCS on an hourly basis.
- Synchronizing CAPI data with our Case Management System (CMS) using the SCS. For this, we are using the synchronization component of the Blaise CMA application. Once cases have been allocated to field interviewers and issued to CAPI mode, they are synchronized through the CMA Launcher database to our CMS system and the Blaise questionnaire.

3.2 Other Functionality

- Automating communications (survey invitations, reminders, thank-yous)
- Sending incentives (gift vouchers)
- Managing incentives and thresholds
- Offering freephone (Help Line) functionality
- Merging contact updates from multiple sources
- Allocating and issuing cases to field interviewers

> back to Table of Contents

3.3 Challenges

During the development cycle, we need to program the Blaise questionnaire and begin testing while setting up the SCS system for the new project and loading sample data for testing. Often the timetable is very tight and managing the quick turnaround of changes can be challenging.

As we start migrating more surveys to the new Blaise 5/SCS platforms, it will become increasingly challenging to manage Blaise version upgrades with live surveys running in Production. Extensive testing will be required, as well as a good understanding of compatibility between developer versions used for the questionnaires and the server versions.

We have had some issues with the performance of components in SCS that are integrated with Blaise using the API. The Blaise Team has provided some additional methods to help improve this. Also, the SCS development team are optimizing their code for other performance improvements.

There are some challenges with the notion of using Blaise APIs vs. Manipula for the development of extended systems and integration with Blaise 5. NatCen's approach is in favor of using APIs and C# for such developments. However, for ad hoc tasks with quick turnarounds, we may still need to use Manipula at times. We are uncertain whether Manipula performs any better than the API, and we may need to do some specific testing around this.

4. What Lies Ahead?

4.1 Multimode CAPI with CAWI and/or CATI

CAPI interviews are done offline, while CATI/CAWI surveys are done online. We are exploring the best approach for keeping them in sync and reconciling any differences afterwards. This involves coordinating the assignment of cases to CAPI interviewers while web and/or CATI interviews are being conducted. It is also necessary to determine the best means for informing CAPI interviewers about surveys completed in CAWI or CATI mode after they have been allocated to field interviewers.

We are working through the various options and best approaches for handling the multilevel hierarchy of sample data for some surveys (e.g., household and individuals) across different modes. For web mode, we may want each individual to complete their section of the survey simultaneously, which implies splitting the individuals into a separate questionnaire and possibly merging them back into one combined household and individual survey questionnaire (and one case) later. Alternatively, we may have to restrict it to one individual completing the survey at a time, and thus keep the household and individual sections of the survey together in one case. For CATI or CAPI mode, we likely would want the interviewer to access the household and all individuals in it to complete various parts of the survey at once.

> back to Table of Contents

What Was Experienced by a Skilled Blaise Programmer in Transitioning from Blaise 4 to Blaise 5

David Dybicki and Peter Sparks, Survey Research Center, Survey Research Operations, University of Michigan

This paper discusses how we transitioned to Blaise 5 from Blaise 4, taking into consideration organizational requirements and standards. To accomplish this, we had to be open to using features in the new environment and be willing to change from the prior platform.

1. New Features

Blaise 4 is essentially a single-user application that multiple users can use simultaneously but which runs as a single instance on a computer. Everything it needs to do is self-contained within the Dep.exe. Blaise 5 is well suited to work in a server environment but is complex because the functionality has been divided into separate server roles that communicate with each other (audit, CATI, data entry, resource, session, web, manager) and that may be on different servers (we typically have front-end web servers and back-end servers that handle the other roles).

Blaise 5 uses server events and listeners to communicate between the services on various servers and has also allowed the Blaise survey to raise custom server events. So, the management of Blaise surveys can be highly customized. For example, several surveys use the server events (such as the survey Completed, Expired, Aborted, and custom values) and write the resulting case status to a management database. The recipient of the events, usually a survey-specific Windows service, can parse custom events and perform complex operations, run stored procedures, write logs, and so forth that make the survey(s) operation seamless. This extension of Blaise is more than just running the survey and waiting for the program to run; it is interacting with Blaise at a new level.

Managing users within the Blaise 5 environment also changed dramatically. In Survey Research Operations' Blaise 4 CATI sample management system (SMS), users are assigned roles and groups, and passwords are managed by Maniplus using additional Blaise databases. In Blaise 5, these functions are maintained within the Blaise server manager in the Users tab. Because of how Blaise works with Active Directory, the user's password is no longer explicitly managed by SMS but can instead be synced so that it matches their login. This makes it easy for the user to no longer remember two separate passwords and places the management of users in one area.

These are just a couple of examples demonstrating significant payoffs for investing the time to learn the new development environment and its capabilities.

2. Layouts

It was not easy to assimilate all these new ways of doing the work without considering them from the Blaise 4 perspective. As a result, there was an effort to try and force Blaise 5 to look and behave in familiar ways. The reasoning was to try and make the transition smooth for interviewers and projects, but the attempt was ineffective. By analogy, it was a case of trying to put a new sporty engine and features into an older style vehicle body, and it doesn't work well. We did learn a great deal in the process, but eventually, we embraced Blaise 5 and took advantage of the new capabilities to expand our capabilities. Layouts in Blaise 4 can also be complex (mode library, menu, configuration, datamodel properties), but not to the same degree as in Blaise 5. The Blaise 5 Resource Database, .blrd, contains all of the Blaise 4 display items but is organized in a way that requires a different way of thinking. Using the layout view and property windows in the Control Centre was key to working with the layouts. Initially, we struggled

> back to Table of Contents

to relate all the different pieces together (templates, properties and parameters, layout sets, master pages, expressions, field references, etc.). However, as we gained experience, these tasks became more manageable. We learned that "less is more" (use less customization per page and more standards).
We realized from our experience in using Blaise 4.8 that we needed standards in the new environment. It was essential to hash out what is required, and similar to having a universal starting mode library file, we worked on (and are still working on) a standard blrd file. We have found that we "borrow" features that work from one study to another but maintain standards for the look and feel of the survey where possible. This builds solid and predictable input screens familiar to all interviewers, study staff, and programmers.

3. Texts and Interfaces

Working with screen text and layouts in Blaise 4 is well known and comfortable. Managing how a particular question looked on the screen meant working with the mode library editor (InfoPane, FormPane, and Grids), font enhancements (like "@B" for bold and "@I" for italic text), datamodel properties (language character encoding and culture), and the menu editor (menu items, panels, and actions). Because of our expertise, surveys can be programmed quickly and efficiently.

But Blaise 5 opened a new world to managing the survey look, feel, and user interaction. The Resource Database, along with the Layout tab and properties panel stored in the .blax.layout file and the Layout section in the source code gave a great deal more flexibility. Font definitions are now named, language text can be copied and pasted directly into the source code (no ANSII transformations needed), and HTML-like commands can be used for additional formatting with text areas.

There are also runtime interactions between the survey and the interface via field reference mappings, languages, roles, type references, function declarations, and template parameters. The complexity of putting text on the screen has increased, but the reward is having an adaptive context-adjusted display that couldn't be done in Blaise 4.

4. Blaise 5 Resource Database

The Resource Database can be considered its own programming area of expertise. Many new concepts are presented: layout sets, templates (such as master, field pane, input controls, and grouping), template parts, default templates, style elements, styles, text roles (embedded), media, master pages, containers (such as grids, panels, groupbox, and areas), controls (like labels, buttons, image, and hyperlink), size allotment (stretch, auto, percent, and pixels) and embedded sizes, applicability conditions, parameters, events, actions, shortcuts, and custom properties.

All of this presented new challenges and forced us to think in novel ways about defining the look and feel of our surveys. Although some parts were easy to understand (styles, style elements, and font definitions), others were harder: text roles, field references, texts, media, templates, and resource sets. One of the most significant concepts we had to grasp was the use of containers within a template and how the properties of auto, stretch, undefined, *, %, and pixels all worked, especially when a container was within a container with different size attributes. It took many hours of trial and error before we understood how everything worked.

5. Abandon Prior Tools and Features?!

Many tools and features are tailored to the environment being used. For example, in Blaise 4, there's the Menu Editor, Data Centre, Bascula, Cameleon, Delta, X-Tool, Mode Library Editor, Dep Configuration, Hospital, and Database management (copy, delete, move, rename, create). In Blaise 5, many of these are no longer needed because the environment is very different. For example, all Blaise 4 menu functions (assigning layouts to fields, layouts of pages, menu items, actions, and panels) are instead found in the Blaise 5 Resource Database, which is managed in the Layout view of the .blax file, or are synchronized

> back to Table of Contents

back to the source code in the Layout section. As a result, there’s no “Menu Editor” in Blaise 5, and so it is strictly a Blaise 4 tool that’s no longer “needed.”

Other utilities, such as Cameleon, essentially became redundant when full access to the meta information in Manipula became available. Blaise 5 moved all of its data storage to relational databases, so data management features (like Hospital) were no longer needed or were moved to new tools. For example, the ability to monitor databases is in Blaise 4 but is accomplished somewhat by the Session Viewer or by running queries against the audit or session data.

Some features of the Blaise 4 language, such as the alien procedure, were not brought forward to Blaise 5. This is used to make custom entry boxes, scan through the datamodel structure for additional checks, and perform logging or other special actions but is no longer needed because the Resource Database has conditional logic and access methods that provide the same functionality.

While knowing how to use tools and features for a particular environment is always beneficial, allowing them to be left behind when moving to a new one is necessary.

6. Programming Standards

At the Survey Research Center at the University of Michigan, we have had many years of programming in the Blaise 4 environment and have developed a set of survey programming specifications and standards. These are used as a guide for successful software development and are applied to every Blaise survey we do. The longer we used Blaise 4, the more complete our standards became.

The challenge in transitioning to Blaise 5 was to make similar standards for code writing and presentation to the interviewer/user of the instrument. This task is complicated to accomplish when the capabilities are unknown and the programming environment is new. We used the Blaise 4 standards as a starting point, but since then, we have been adapting and changing them to fit the new system. These standards are critical for helping programmers build an instrument that meets and exceeds the client's expectations.

We discovered that we had to learn the new system comfortably before effectively writing new standards. We started with what we knew and then adapted them as needed.

7. Programming Concepts Changed

Although it’s very comfortable to remain in the old way of thinking and using programming methods and utilities, it is best to change with the new environment. However, there’s a price to be paid, and that price is a steep learning curve.

For example, when introduced to the new GROUP structure, it was confusing, and we assumed it functioned the same way as BLOCK but with some tie-in with the layout. Instead, the GROUP structure is a hybrid where it allows fields to be displayed together using a grouping template, but in terms of implementation, the RULES for the fields involved belong to the GROUP (like it is within BLOCKs). But it is unlike a BLOCK in that fields and auxfields *are not* stored within the GROUP level but are stored at the same level where the GROUP is defined. This “old” way of thinking caused a great deal of head-scratching until the new concept was learned and our assumptions were abandoned.

In Blaise 5, there were many new added features that alter how the surveys are programmed: constants, user-defined attributes, field properties (e.g., isVisited), conditional logic within templates, field parameters, role texts, and many more that control the logic flow and interface. The key to programming is knowing exactly how these new features work and using them correctly (not the old ones).

> back to Table of Contents

As we became better experienced and more confident in the Blaise 5 environment, we took advantage of new features and methods to improve the surveys and performance. We let go of old strategies that no longer worked well or were prohibitive to maintain. In addition, we continued to learn by trying new features and methods. We worked on improving Blaise by reporting bugs and asking for help when we couldn't figure it out independently.

8. Interview Interface Changed

In Blaise 4, we were comfortable with a display of question text at the top panel (InfoPane), information about the question answers in the middle panel (code responses or range information), and then entry for the field in the bottom panel (FormPane). There was just one question per page, and navigation between fields was all done in the field pane. DK/RF/Remarks symbols appeared next to the entry box, and help popped up in a separate window.

In Blaise 5, all of this changed: we could freely rearrange all these parts, the entry field could be placed practically anywhere, there could be multiple entry fields, help appeared in a different text box, controls could be hidden/shown on demand, menu controls were largely removed, previous/next buttons became the norm, and screen displays became dynamic.

This impact is a far more interactive display for the interviewer and the respondent and matches what is used in modern applications. It's important to "keep with the times," even though the basic functionality is present in both Blaise 4 and Blaise 5. That means being a Blaise 5 programmer also means another set of skills is needed in the programming toolbox, and that is being a web designer.

9. APIs and Versions

Blaise 4 is very rich in terms of providing the ability to dive into the metainformation of the datamodel and retrieve everything: every bit of rules logic, parameters, fields and attributes, texts, languages, layouts, code texts, and so forth. Blaise 5 is also very rich and provides all the same information.

However, trying to navigate through the datamodel using the application programming interfaces (APIs) is very different between the two platforms, and it simply did not work to apply Blaise 4 API methods to the Blaise 5 API. We found the best method to learn the new API was to step through sample programs using the .Net debugger and examine the areas of interest (objects, properties, and methods). Once we found how to retrieve the information and the context in which it was used, we implemented it within our own utilities and built up our expertise.

We also had to learn what files and DLLs had to be present for running our utilities and licensing requirements to deploy and use them. Part of the lessons we learned is that any program that uses a version of Blaise tends to be version specific. That is, suppose a data utility is compiled using Blaise 5.6, and then is used against a Blaise 5.13 database; it may not function properly, but a compiled utility using 5.13 may work with Blaise 5.6 data. Blaise extends the Blaise 5 API with each new release so utilities can reference older methods using the new API.

We recommend upgrading utilities to use the new API (as appropriate) and methods whenever possible to take advantage of bug fixes and new features.

10. Data Storage Expanded

In Blaise 4, there was one "flavor" of data storage: a proprietary format that consisted of several files, along with additional utilities to manage the storage (such as Hospital, Control Centre copy, move, rename, and delete). All this changed to a relational database in Blaise 5. In addition, many more options were given regarding internal table structures with data partitioning (Stream, Flat No Blocks, Flat Blocks,

> back to Table of Contents

In-Depth, and single table), storage types (Non-generic and generic), and platforms (SQLite, SQL Server, Oracle, Excel, MS SQL, among others).

In addition, the audit information changed from a file into a database with two tables. Both require parsing, but working with the database using SQL queries is much easier.

What constitutes a data change is also different between the two versions. Changing the name of a field in Blaise 4 is not considered a data change, but moving that field to a different position or changing an attribute will require a data migration. In Blaise 5, the opposite behavior occurs (name changes require migration, but moving names/changing attributes don't).

Because of the relational database, Blaise programmers had to pick up additional database skills and understand that the data was no longer directly accessed but instead went through a data link file (connection properties) managed by Blaise. Hence, if the data link referenced a survey on a server, no matter where the .BDIX resided, the data could (potentially) be retrieved, modified, and written. As a result, we learned a great deal about database user accounts, logins, schemas, and the like.

An area that we have not explored yet but are willing to try is cloud storage and hosting. It is possible for specific surveys, but additional areas must be scrutinized: security, availability, reliability, cost, etc.

The basic concepts of working with data remain a great starting point, but the new environment requires specific knowledge of how the system works. Only the minimum of data manipulations can be accomplished without the additional training.

11. Audit Information

Likewise, the audit information in Blaise 4 was no longer stored in a single text file (or even multiple files or multiple cases within one text file) but instead is stored in a relational database in Blaise 5. Our utilities that parsed the keystroke-level information started as the base for new software but quickly changed and became customized to the new environment. However, the function of our utilities remained the same: to analyze the data for section and field timings, recover cases, and understand areas where a given survey might be having issues.

12. Session Data

In Blaise 4, just one database contained the initial preload, working cases, and completed cases. If a case needed to be restarted, it was simple to run a Manipula Script. However, the Session Database (the concept first introduced in Blaise IS) took more effort to work with. The data from the main database is copied to the Session Database, and then all further breakoffs and resumes use the Session Database. We quickly discovered that we must retrieve cases from both Main and Session to get a complete snapshot of production.

This has several important implications: any data pulled from the main database will not have partial case data unless the session data to the main database is explicitly copied (usually by a Manipula Script), or the case has been completed. Therefore, tracking the overall status of a web survey can be inaccurate when only looking at the main database. In addition, the Session Database stores the "state" of the survey: every local, auxfield, and field property is present in the data so that when a case resumes, it is in precisely the same place where it left off. This caused all sorts of problems for our Blaise 4 way of thinking (auxfields and locals were initialized every time a case started) until we understood that we had to explicitly manage this "reset" by passing in a parameter/flag to initiate code to perform this function.

> back to Table of Contents

The automatic storing of case data in the Session Database has proved to be a bonus for some situations and a hindrance for others. For example, when case data is accidentally reloaded to the main database, it is possible to restore cases quickly from the Session Database by copying them to the main database. However, changing preloaded incentive amounts is more challenging because the session data cannot be written directly. (There are a few methods to address this: one way is to download the session data to the main, delete the session data, and then update the main data with the new incentive. When the case is restarted, a new session will be created from the main data.)

Like other new methods in Blaise 5, this initially caused us problems from our lack of understanding, and we tried all sorts of ways to get around what we thought was a restriction. After we gained the necessary knowledge and worked with the feature, then we gained the benefits it offered.

13. Manipula

Manipula has been the mainstay workhorse for our use in Blaise 4. It has been critical for our processes: loading preload, moving and updating data, exports, alien procedures, specialized parsing, interaction with CATI, creating crosswalks from the meta information, and so much more. We literally could not run our surveys without this valuable tool.

Blaise 5 Manipula is just as valuable. However, like many areas in Blaise 5, it was the same and not the same. During the transition period, as Blaise 5 Manipula was still being developed, we had to learn/relearn the language because it was a work in progress. Language features were removed because those areas were no longer part of the Blaise 5 system, and others were added to work with new features (like write interceptors, CMA, and session data). Maniplua was also updated to use Action Setup Manipula Scripts or Manipula Dialogs.

Because Blaise 5 Manipula is so diverse, powerful, and flexible, we are still learning its capabilities and are hard at work to become even better experts. For example, many API functions are available within Manipula, but it takes advanced programming skills to dive into them.

14. Blaise Control Centre Changed

A big challenge has been for programmers to learn the Blaise 5 Control Centre, and it seems that everything changed. We had been familiar with menus, file types, projects, tabs, and where all the parts could be found. However, in Blaise 5, there were now solutions, packages, different previews for different environments, tool and property windows that can be dragged/dropped/resized, different file types, context-sensitive menus, and so much more. It took some time to become familiar with the panel arrangements, especially the placement of the Source, Meta, Settings, and Layout tabs for the .blax file.

Editing in the source code was different; some familiar keystrokes were no longer present, new ones were added for new features, and a few changed how they behaved. This is typical for different software environments.

Regarding functionality, Blaise 4 and Blaise 5 perform admirably for what they're designed to do. Multiple parts of Blaise 4 are integrated into Blaise 5 in a way that makes sense. For example, the Delta tool in Blaise 4 shows diagrammed program flow for a datamodel. It works with the datamodel and acts like a "plug-in" added to the Blaise system. In Blaise 5, it's integrated into the menu by opening the .blax in the Control Centre—selecting the Meta view—then clicking the Statement graph button.

There's additional code handling functionality that's built into Blaise 5. Some handy shortcuts are comment/uncomment code, bookmarks (also found in Blaise 4), and source structure browser. In Blaise 5, a new editor concept of code snippets can be customized to save time writing standard programming

> back to Table of Contents

code. The Find/Search & Replace also have been changed and function differently. For example, there's no "enhanced" find/replace in Blaise 5 because it's controlled by the scope of the action (selection, document, project, and solution).

Autocompletion and parsing while editing are features that are strictly in Blaise 5. Because of the new editor and functionality of the Blaise 5 Control Centre, there has to be a period of adjustment before becoming proficient. The transition period may be frustrating and slow, but the result is a greater ability to program efficiently.

15. Interviewer Help

The help file for Blaise 4 has always been a mainstay for our interviewers and desired by our project staff, but it has become problematic: support for the help file format has been deprecated. This means there is extra setup on the interviewers' laptops to install software/DLLs, and sometimes extra effort for the Blaise programmers to keep this feature working. It may become unusable if the operating system no longer supports the software to run the help.

In Blaise 5, the help moved into the survey itself as a role text and can be mode specific and easily translated in multiple languages. The change means the staff no longer edits an RTF file, but the programmer has to place the text into the survey. The interviewer views help differently by typically using a button on the screen to show the help text on the page, rather than seeing the help in a separate window.

16. All Interviewing Modes Possible

Blaise 4 had one mode with data collected in different environments (laptops, desktops, and via management systems), so each survey was typically programmed for each particular survey mode. However, Blaise 5 has true modes (CAPI, CASI, CATI, etc.) that can be used in different environments, such as iwer-assisted CAPI + DEP on remote laptops, iwer-assisted CAPI + Web on local computers, respondent self-collected CASI + Web on desktops/phones, and so on. Typically, the attributes in our Blaise 4 environment have been defaults of DK, RF, and NOEMPTY, but Blaise 5 is richer in that all these are contained within one datamodel and the mode can be set at runtime. This gives excellent flexibility and consistency for the surveys.

17. Mode Switches

Because our Blaise 4 data was collected in essentially one mode, switches between modes was rarely an issue. However, planning for mode switches has become a necessity when using Blaise 5.

As noted above, Blaise 5 allows multiple modes in one survey, and the attributes of self and interviewer assisted tend to be opposite of each other. Going from DK and RF allowed to not allowed implies these values are dropped, while going from EMPTY to NOEMPTY means questions that have been skipped will now be asked. Different methods have been used to implement mode switches, such as blocking completed sections, restarting partially completed sections, copying unique mode-related fields between mode switches (and assigning correct values), and not allowing mode switches.

18. Security Enhancements

The Blaise 4 data is stored in a proprietary format that is readable via the data viewer, Manipula, and BCP programming—all different ways that access is granted openly. Security on the data is maintained at the file and directory levels.

> back to Table of Contents

However, in Blaise 5, the data may still be in a file (.bdbx using SQLite—and the same file security methods apply) or in an SQL server or other relational database software where it is referenced from a data link file (.bdix). Learning how best to secure the survey data (audit, session, and main survey data) in the new environment was essential. This .bdix can be put anywhere and the data can be accessed (with caveats) as long as there's a connection to the server hosting the data. Therefore, careful attention to the location of production .bdix files is needed to avoid unauthorized access to respondent data; that is, production .bdix files should be kept in secure locations. Blaise 5 allows password protecting the .bdix file to limit this possibility, which is a feature not available in Blaise 4. Additional welcome security measures have been implemented in general to the Blaise system, such as in the AD synchronization, server manager, and server manager roles and user passwords.

19. Increased Use of Client/Server

Blaise 4 (using Blaise 5 terminology) is a thick client survey (survey + data collection is done without server contact). In Blaise 5, most surveys use a thin client (web survey with heavy server use) to retrieve survey pages and store survey data; otherwise, the Blaise 5 thick client is similar to the Blaise 4 survey. We did not grasp this concept initially, and only after working with Blaise 5 after a while did it make sense. We had to move out of the prior thinking and into the client/server world. We're now much more familiar with the capabilities of Blaise 5 and have been able to use it for self- and iwer-assisted surveys in DEP, custom Dep, and client/server. It took some time to gain expertise and understand how the pieces work. We had to invest time and effort into building login applications (portals) for respondents, learning IIS management, Blaise server management, building a custom DEP, and writing upload and download interceptors in Manipula.

20. Conclusion

Blaise 4 and Blaise 5 are rich, complex, versatile, adaptable, and robust, and they can handle difficult situations. In moving to Blaise 5 we viewed nearly all the new features, methods, editors, and utilities by looking through the Blaise 4 lens, which, in effect, distorted the new reality, caused more work, decreased our productivity, and limited what could be accomplished. As we gained understanding and embraced the new ways of working with the system, we enjoyed more excellent capabilities in our surveys, faster programming, and more possibilities. Not everything is peaches and cream in the new environment, and there are still difficulties to overcome, but we have grown to appreciate Blaise 5 without the Blaise 4 overtones. We can apply these lessons to any new software or environment and admit that we still have much more to learn.

> back to Table of Contents

A Short History of Blaise Code Generation

Leif Bochis Madsen, Statistics Denmark

1. Abstract

The history of Blaise is also a history of finding easy ways to make specifications for questionnaires without needing to write Blaise code—either because the subject matter specialists who are designing the surveys are not familiar with the writing of code or because the writing of code is simply a tedious and unwelcome task. Besides, many surveys originate from specifications already described in, for example, database tables or metadata repositories—and nobody wants to write similar specifications twice in different syntaxes.

This paper describes various ways of generating Blaise code from external sources—or via alternative ways of describing questionnaires—primarily from work done in Statistics Denmark, but with reference to experience from other organizations. It includes a discussion of benefits and limitations of the different approaches.

2. Introduction

Recently, I had the pleasure of introducing a group of IT staff members from the Jordanian Department of Statistics to the Blaise Survey system and showed them how to use Blaise to implement questionnaires.

When I started to make some simple examples of Blaise code and continued to show the famous Flight Survey demo code in order to exemplify all the nice features and the structure of Blaise code, I was met with a polite, but unmistaken, disappointment.

It was clear that an easier way of "putting the questions into Blaise" was requested, and I remembered the old fact that IT professionals don't want to write code when it is so much easier to (make other people) generate it.

Luckily, I could—after a few hours of work—get back and successfully demonstrate one of our Blaise Code Generators picking a Word document and nicely producing a Blaise data model ready to prepare.

In the end, I was asked if we could provide the generator for use in the Jordanian Department of Statistics, and I promised to do so, of course, while also starting to think of how much work it would take to liberate the tool from our own IT environment.

Afterwards, I started thinking about all the various Blaise Code Generators that people wrote in order to ease "putting questions into Blaise" and realized that a summary of these efforts and approaches to the subject would be a good topic for an IBUC paper and presentation.

3. History of Blaise Code Generators in Statistics Denmark

The history of generating Blaise code in Statistics Denmark goes back to the past century.

A variety of attempts to make the "right" and "(almost) complete" generator comprise a long list of approaches and tools:

> back to Table of Contents

At first, various Excel spreadsheets were used to define the questions and answers, with a number of different structures in order to get as much metadata as possible described. Scripting languages, such as VB or Manipula, were used to generate Blaise code. Merely copying the contents of a sheet and pasting it into a Blaise source code would also make sense.

Secondly, a structured Word document supplied with macros could form the basis of a Blaise questionnaire.

A third and more successful attempt is the Word-based Blaise Code Generator for Household Surveys, which is still used. It is described further in the next section.

The fourth generator is the so-called Business Survey Configurator, also described below.

Finally, attempts have been made to build a Blaise Code Generator in Blaise (i.e., to define a data model describing a Blaise questionnaire) using the data entry program to describe blocks, fields, datatypes, etc., and using Manipula scripts to generate a target Blaise data model. This generator worked fine as a proof of concept, but due to lack of interest among the targeted users, it was never finished or used for any production.[1]

4. Blaise Code Generator for Household Surveys

The aim of this code generator is to provide an easy-to-use way to describe questionnaires. Thus, the description tool is merely a Word document (or any text-processing document that can be converted into Word) and is aimed at any customer (internal or external to the organization) with limited or no knowledge of Blaise.[2]

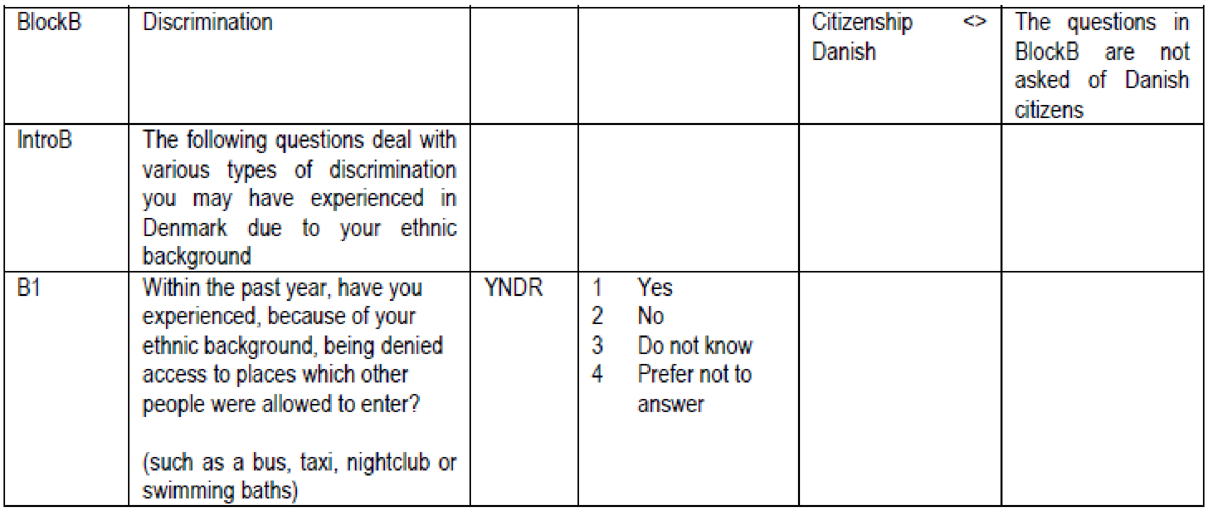
It defines a simple "description language" based on tables (blocks), rows (fields), and columns (name, question text, type name, type definition, filter condition, and comments).

[1] I presented this generator at the BCLUB meeting in Copenhagen, January 2016, under the code name "Blaise Monkey." I might be able to dig up some remains of the code if someone is interested. I was inspired by [Vreugde, 2003].

[2] [Madsen et al., 2013] contains a detailed description of this code generator.

> back to Table of Contents

Figure 1. Fragment of Sample Word Document (from [Madsen et al., 2013])



BlockB	Discrimination			Citizenship <> Danish	The questions in BlockB are not asked of Danish citizens
IntroB	The following questions deal with various types of discrimination you may have experienced in Denmark due to your ethnic background				
B1	Within the past year, have you experienced, because of your ethnic background, being denied access to places which other people were allowed to enter? (such as a bus, taxi, nightclub or swimming baths)	YNDR	1 Yes 2 No 3 Do not know 4 Prefer not to answer		

An automated process carries out the following tasks:

1) The Word document is converted into its XML equivalent.
2) The XML document is extracted by an XSLT process, generating Blaise code.
3) Standard templates suited for the overall system architecture are applied.[3]
4) A version of the data model without rules is prepared in order to facilitate the checking of filter instructions (the rules part is the difficult part of the data model to produce correctly).
5) This version of the metadata is applied in a process that generates valid rules instructions.
6) Often, a preparable (though not necessarily complete) source code is the result of this process.

The generator was built using XSLT, C#, and Manipula for different processes. It has been slightly maintained over the years (e.g., updated to possibly produce Blaise 5 code) and is considered a "90 percent generator" (approximately).

This tool has met the primary aims of being an easy-to-use tool for describing questionnaires and is still in use in our household surveys division. It is mainly efficient for the generation of first versions of questionnaires, ready for refinements like layout and more complex rules.

5. Business Survey Configurator

The Business Survey Reporting System started in 2006 using Microsoft InfoPath as a questionnaire development tool. Due to the end of support for this tool, from 2018, Statistics Denmark started examining and later migrating Business Reports to Blaise 5. In the beginning, we were using the Word/table-based generator to generate Blaise 5 code, but soon we realized that we needed to address compatibility between the XIS[4] repository and the Blaise instrument.[5]

[3] This comprised a template for a standard blax file for Blaise 5 and included standard blocks for taking care of, for example, preloaded data about the survey and/or respondent. Thus, it is possible to use this generator for different setups by changing a reference to the templates.
[4] XML Input System (XIS) is a general system for storage of business survey data at Statistics Denmark.
[5] The Business Survey Framework is described in detail in [Madsen, 2018], along with some related tools in [Madsen, 2020].

> back to Table of Contents

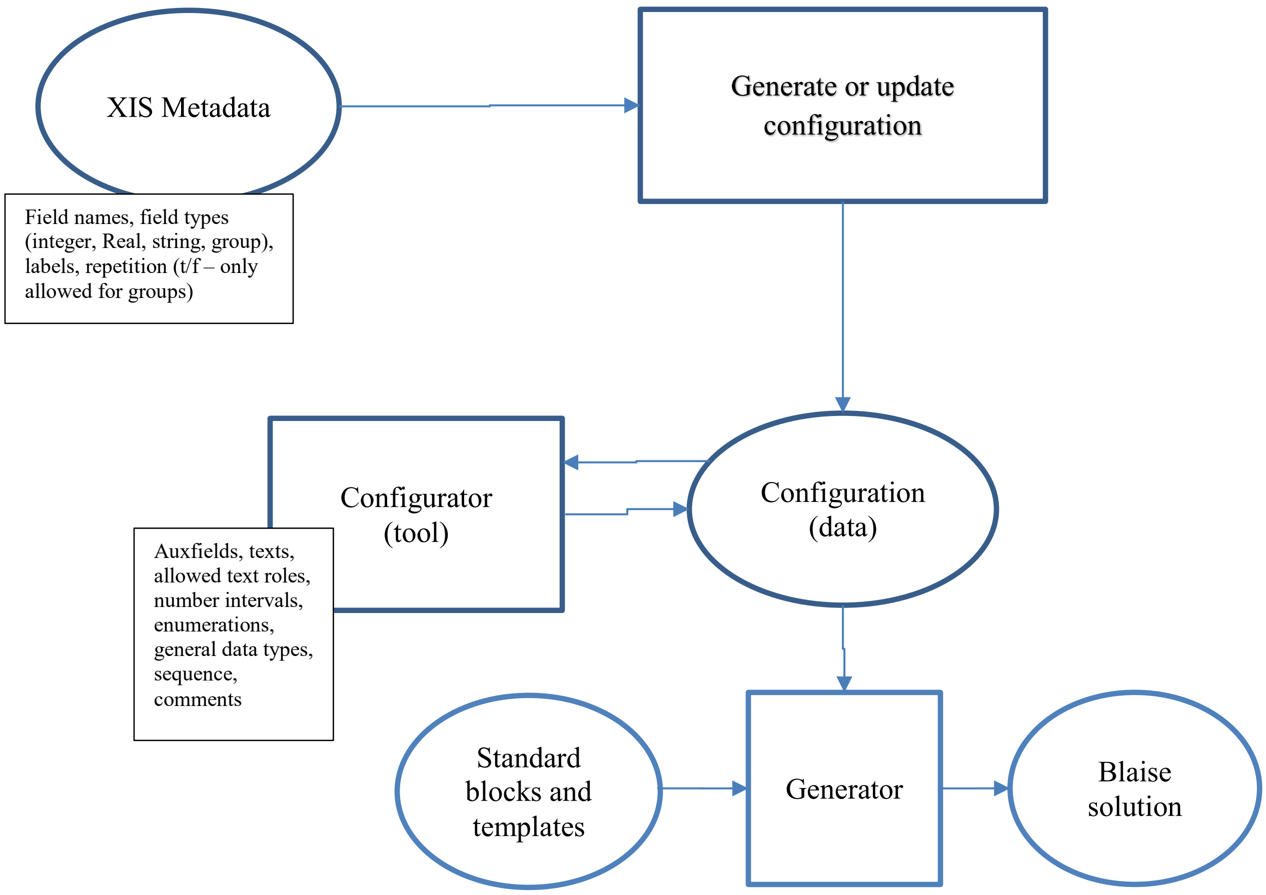
The aim of the Business Survey Configurator is to generate a Blaise project ready to prepare—and compatible with the metadata from the XIS repository. Field names in the XIS repository (as opposed to Blaise) are treated as case sensitive, so in order to automatically fetch data from a Blaise database via its metadata and store these data in XIS, it is important that fields in the Blaise instrument use the same casing as in the XIS database.

Because the XIS repository contains relatively poor metadata—and therefore, only a poor Blaise instrument could be generated directly from the XIS repository—the Configurator was developed, so supplementary metadata could now be added to the code generation. These metadata comprise auxfields, role texts, and precise data types (the XIS repository only defines the basic datatypes integer, real, and string).

Due to the limited functionality implemented in the Configurator, we are talking about a one-way generator, and it is only used the first time for an instrument. Needed modifications (e.g., layout instructions) are all carried out with the Blaise Control Centre. When subsequent changes need to be implemented for repeated or periodic surveys, maintenance will be done via the existing Blaise solution. Possibly, if a large number of fields or entire blocks are added, a new version may be generated and new elements copied into the existing Blaise solution.

> back to Table of Contents

Figure 2. Process Diagram for the Business Survey Generator


XIS Metadata
Field names, field types (integer, Real, string, group), labels, repetition (t/f – only allowed for groups)
Generate or update configuration
Configurator (tool)
Auxfields, texts, allowed text roles, number intervals, enumerations, general data types, sequence, comments
Configuration (data)
Standard blocks and templates
Generator
Blaise solution

Soon, the problem of synchronization between the metadata in the configuration and the actual metadata in the instrument became visible and important to handle.

In order to facilitate use of the Configurator as the first choice of description—and as maintained storage of survey documentation—we launched an experiment. The experiment was based on the idea that we could automate the process of adding new elements and retain the changes (primarily rules and layout instructions):

1) Generate a new solution and prepare it.
2) Extract the metadata as XML.
3) Extract the metadata from the previous solution as XML.
4) Compare the two XML structures, add new elements, remove elements (fields and blocks only) not present in the new version, and let existing elements stay unchanged (including rules).
5) Generate new source code from the changed XML document (via XSLT process).

Actually, we succeeded in making a useful source code that could replace the previous version, with rules and layout instructions concerning the previous questions retained. Proof of concept—with one exception:

> back to Table of Contents

All comments are removed from the source due to steps (2) and (3) because they are not part of the metadata.

Keeping the comments would require that they are included in the metadata (e.g., by adding annotations as a text role), but that would not help with keeping general comments attached to rules instructions (i.e., apart from conditions and checks). Thus, this grand idea still needs some further care in order to be practically useful.

All in all, the Configurator and Generator make a good beginning, and the instruments may be refined further by editing rules and adding layout instructions in the Blaise Control Centre. It has been an important tool in order to create Blaise instruments to replace Infopath report forms during our conversion process. It is, however, not clear how important it will be in the future, where the focus will instead be to maintain and make changes to new versions of the mainly periodically repeated business surveys.

6. Blaise Colectica Questionnaires (BCQ)

The combination of Colectica and Blaise has gained popularity among NSIs in the recent years. Many organizations have a policy of documenting their data in DDI, and Colectica offers a user-friendly access to DDI.

At Statistics Denmark, we have one example of using Colectica-generated questionnaires, as we designed a questionnaire in 2018 for the survey *European Statistics Code of Practice* (COP) in Colectica and generated the Blaise code. However, a lot of changes had to be carried out afterwards, especially concerning constructs for improving layout. The COP survey is annual, and minor changes need to be implemented for each version. We realized that the easiest way to accomplish this was to adapt these changes by editing the Blaise source in the Blaise Control Centre, and that is how we have maintained the COP questionnaire since the first version.

Apparently, other users of BCQ have the same experience using BCQ to generate the first version and implement subsequent changes in BCC, though it requires double editing—in Colectica as well as in BCC [CSO Ireland, 2023].

This autumn, however, we have initiated a project with the aim of modernizing the Labour Force Survey (LFS). Among the goals are replacement of older software and legacy code (e.g., replacing Blaise 4 with Blaise 5) and usage of Colectica/DDI for documentation. One of the tasks of this project is about defining design features in BCQ that may allow us to regenerate LFS questionnaires for every new data collection period.

7. Conclusions

Many have tried to find their best and most efficient way to generate Blaise code. For example, the data collections department of Statistics Netherlands uses a very impressive generator based on a Word document comprising descriptions of questions and a Visio document containing a graphical representation of the rules, including conditions.[6]
But also, there are many others, as suggested by the vague selection of papers about the subject in the references.

[6] Described by [Bolster, 2009], [CBS Netherlands, 2023], and [Pisiren, 2023].

> back to Table of Contents

Most successfully, probably, are solutions with emphasis on the local culture, language, and environment. Also, reuse of already available data from different kinds of data sources, including existing metadata or repositories, is an important foundation for efficient generation of instruments. Existing frameworks for processing survey data may be necessary to take into account, and cultural views or design traditions may have impact on the success of a solution.

Maybe the Blaise Colectica Generator will be a good choice for the future of Blaise Code Generation with its firm base on the DDI standard. However, not all organizations are using Colectica, so there might still be a need for other kinds of generators.

In general, Blaise Code Generators tend to be deeply rooted in one organization—its culture, language, design traditions, and habits of documenting and storing metadata. This is definitely also true about the generators we developed in Statistics Denmark.

Therefore, the implementation of a generator in an organization is a delicate task, and a number of questions should be taken into account when doing it. Among the questions to address are:

1) Who is going to use the generator and how much knowledge of the survey tool (i.e., Blaise) do they need to possess?
2) Level of ambition: How much of the code needs to be generated automatically?
3) Are (some of) the data already described in a repository of some kind?
4) Which kind of "description language" can we provide in order to define a questionnaire?
5) How much time may the user spend in order to learn how to describe a questionnaire?
6) How can we automate processing from a description to (part of) a solution?
7) How would we manage regeneration in case of repeated surveys?

The references include a noncomprehensive list of papers from recent Blaise conferences dealing with this topic.

> back to Table of Contents

8. References

[Bolster, 2009] BlaiseIS at Statistics Netherlands, Gerrit de Bolster; Statistics Netherlands, in: *Proceedings of the 12th International Blaise Users Conference*, Riga 2009, https://blaiseusers.org/2009/papers/4d.pdf

[Bolster, 2013] Generating Blaise from DDI, Gerrit de Bolster; Statistics Netherlands, in: *Proceedings of the 15th International Blaise Users Conference*, Washington DC 2013, https://blaiseusers.org/2013/papers/3a.pdf

[CBS Netherlands, 2023] Unrecorded presentation on Zoom, May 10th, 2023

[CSO Ireland, 2023] Unrecorded presentation on Zoom, June 6th, 2023

[Filippenko et al., 2007] Questionnaire Specification Database for Blaise Surveys, Lilia Filippenko, Joe Nofziger and Valentina Grouverman; RTI International, USA, in: *Proceedings of the 11th International Blaise Users Conference*, Annapolis 2007, https://blaiseusers.org/2007/papers/E3%20-%20Questionnaire%20Specification%20Database.pdf

[Filippenko et al., 2013] Web-based CAI System for Blaise Instruments Development, Lilia Filippenko and Sridevi Sattaluri; RTI International, in: *Proceedings of the 15th International Blaise Users Conference*, Washington DC 2013, https://blaiseusers.org/2013/papers/3d.pdf

[Joyal et al., 2013] Blaise Code Generator, Eric Joyal, Jason Gray and Sam Threinen; Statistics Canada, in: *Proceedings of the 15th International Blaise Users Conference*, Washington DC 2013, https://blaiseusers.org/2013/papers/3c.pdf

[Madsen et al., 2013] Automatic Generation of Blaise Data Models, Leif Bochis Madsen, Saliha Zayoum, and Lars Peter Jørgensen; Statistics Denmark, in: *Proceedings of the 15th International Blaise Users Conference*, Washington DC 2013, https://blaiseusers.org/2013/papers/3b.pdf

[Madsen, 2018] Implementation of Blaise 5 Web Questionnaires into an Existing Business Survey Management System, Leif Bochis Madsen; Statistics Denmark, in: *Proceedings of the 18th International Blaise Users Conference*, Baltimore 2018, https://www.blaiseusers.org/2018/papers/5_5.pdf

[Madsen, 2020] Managing a Complex Infrastructure for the Collection of Business Report Forms, Leif Bochis Madsen; Statistics Denmark, in: *Proceedings of the 19th International Blaise Users Conference*, Limassol (Zoom) 2020, https://blaiseusers.org/2020/papers/IBUC2020_S2_4.pdf

[Ostergren, 2013] A New Tool for Visualizing Blaise Logic, Jason Ostergren; University of Michigan, Survey Research Center, in: *Proceedings of the 15th International Blaise Users Conference*, Washington DC 2013, https://blaiseusers.org/2013/papers/8c.pdf

[Pisiren, 2023] Coskun Pisiren: Presentation at BCLUB meeting. London, May 2023 (available from BCLUB basecamp site)

[Seymour, 2010] Moving to a Metadata Driven World, Chris Seymour, Statistics New Zealand, in: *Proceedings of the 13th International Blaise Users Conference*, Baltimore 2010, https://blaiseusers.org/2010/papers/3a.pdf

> back to Table of Contents

[Vakhaetov, 2010] MetaDEx - From Data Dictionary to Complete Blaise Data Model Code Generation, Farit Vakhaetov, Department of Population Health Research, Alberta Health Services - Cancer Care, in: *Proceedings of the 13th International Blaise Users Conference*, Baltimore 2010 (abstract), https://blaiseusers.org/2010/slides/p3c%20Vakhaetov%20MetaDEx.pdf (presentation)

[Vreugde, 2003] Blaise Survey Generator, Carlo J.C. Vreugde, Jacques J.M.J. de Groot, Christiaan van 't Hoft, VNG, SGBO, StimulansZ, of the Netherlands, in: *Proceedings of the 8th International Blaise Users Conference*, Copenhagen 2003, https://blaiseusers.org/2003/papers/Blaise_survey_generator.pdf

[Wensing et al., 2007] Exploration of Blaise Instrument Generation from Metadata, Fred Wensing and Juanita Pettit; Australian Bureau of Statistics, Australia, in: *Proceedings of the 11th International Blaise Users Conference*, Annapolis 2007, https://blaiseusers.org/2007/papers/F3%20-%20Exploration%20of%20Blaise%20Instrument%20Generation%20from%20Metadata.pdf

> back to Table of Contents

Speedy Incentives from Blaise 5 Instruments

Emily Caron, Jerry Copperthwaite, and Rhymney Weidner, RTI International

1. Abstract

One of the popular ways to encourage respondents to complete a survey is to provide them with a monetary incentive. Blaise 5 does not have a direct way to instantly send incentives to respondents, but by leveraging existing Blaise functionality and PowerShell scripts—along with a separate, specially developed system—we were able to implement "instant" incentives for a recent project. The respondent was able to receive a digital incentive within minutes of completing the survey. This paper will examine features utilized in Blaise 5 and provide a brief explanation of the accompanying scripts and other pieces involved that made this feature possible.

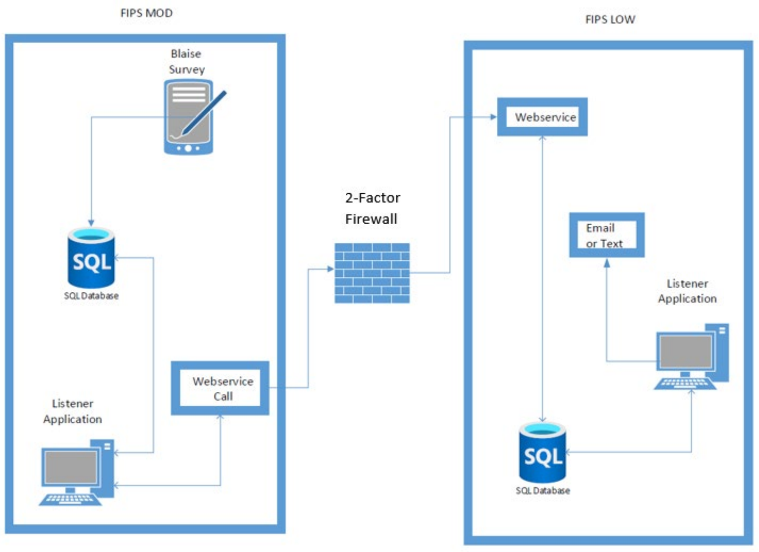
2. Introduction

One of the best ways to encourage respondent participation in a survey is to provide a monetary incentive for completing it. In the past, these incentives were provided by RTI either via a mailed check or an e-gift card. E-incentives were processed by overnight batch jobs, which resulted in a delay of roughly 24 hours. Mailed incentives were slower and took up to a few weeks to reach respondents.

Recently, RTI endeavored to implement a "speedy" incentives process, which enabled respondents to receive an incentive via email or text within minutes of completing the survey if one of those digital delivery methods was selected. These incentives came in the form of an e-gift card that could be retrieved via a link included in the email or text (as selected by the respondent). This paper will cover the specialized process developed to trigger these incentives, focusing on the Blaise 5 features utilized to hook into the backend processes that take care of delivering the incentives. We will also outline some of the challenges encountered and lessons learned while implementing this process.

3. High-Level Overview of the Speedy Incentives Process

Figure 3a. An Overview of the Speedy Incentives Process Flow



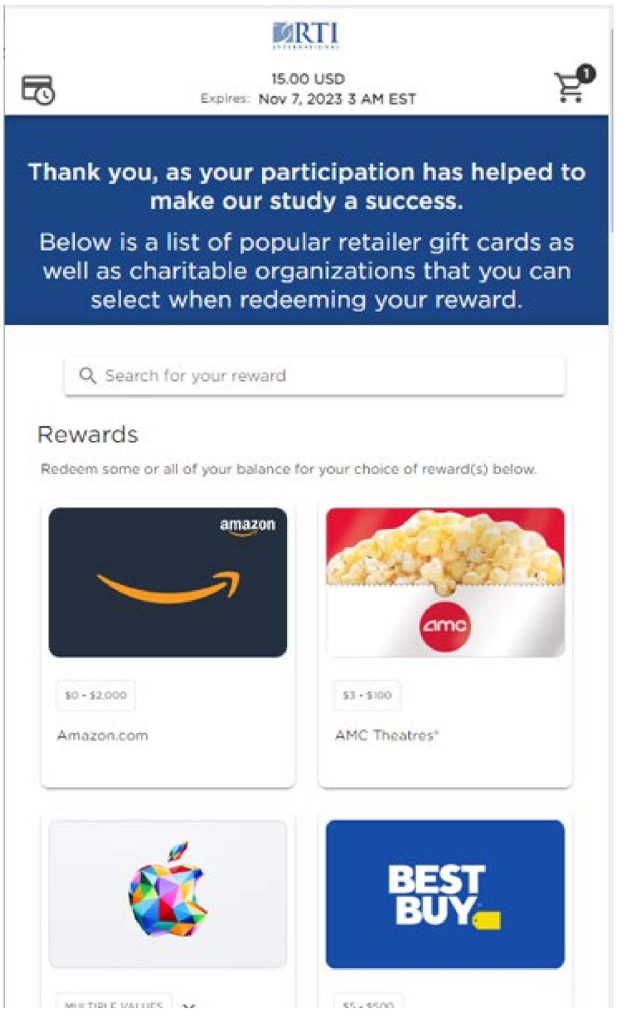
> back to Table of Contents

The processes in the Figure3a diagram flow from the FIPS Moderate (FIPS Mod) side of the firewall to the FIPS Low side. Our FIPS Mod network is a standalone, dedicated network that employs a highly restrictive set of security controls and requires multifactor authentication. This is where the Blaise-collected survey data are stored. The starting point of the speedy incentives process is the Blaise survey, which contains logic to write incentive metadata to a SQL database table. The next process is a Listener Application that runs in the background in the FIPS Mod network and contains a timer to query the SQL database table for new incentive records inserted via the Blaise survey.

If a new record is found, an insert function of a web service located in the FIPS Low network is called to load the incentive metadata to a corresponding SQL database table. Next is another Listener Application that runs in the background in FIPS Low. This process contains a timer to query the SQL database table for new incentive records inserted via the web service. If a new record is found there, the process calls an incentive API to purchase a new incentive, which is then sent to the respondent via email or text.

A Simple Mail Transfer Protocol application is used for building and sending the email messages, and an in-house ARTEMIS application builds and sends the text messages. The email and text messages contain a link for respondents to claim their incentive via Tango®, where they have a number of choices for "cashing in" their incentive amount.

Figure 3b. Example Screen for Redeeming Incentives


15.00 USD

Expires: Nov 7, 2023 3 AM EST

Thank you, as your participation has helped to make our study a success.

Below is a list of popular retailer gift cards as well as charitable organizations that you can select when redeeming your reward.

Search for your reward

Rewards

Redeem some or all of your balance for your choice of reward(s) below.

$0 - $2,000

Amazon.com

$3 - $100

AMC Theatres®

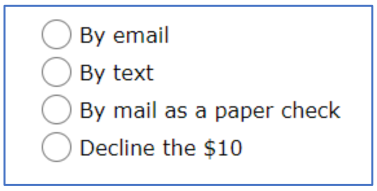
> back to Table of Contents

4. Blaise Features Utilized to Trigger Incentives

The first implementation of the speedy incentives process triggered by a Blaise survey was in a survey designed in version 5.12.8. Triggering the incentive required a specialized template, a call to a PowerShell script from an actions setup function, and a call to a stored procedure from the PowerShell script.

While developing the templates for the incentive process, we had to determine at what point we wanted to send the incentive and how we could prevent duplicate incentive attempts. Even though the backend processes contained checks to prevent duplicates, we wanted to avoid unnecessarily triggering any of these from Blaise in the first place. First, it was decided that the incentive should be sent as soon as the respondent completed the screens containing incentive questions. This location was towards the end of the survey, although it was not required that respondents fully *complete* the survey to receive the incentive. The respondent could elect to receive their incentive either via email or text (digital delivery) to be eligible for a "speedy" type of incentive. Other options not involving this new process included receiving the incentive via mail or choosing not to receive any incentive.

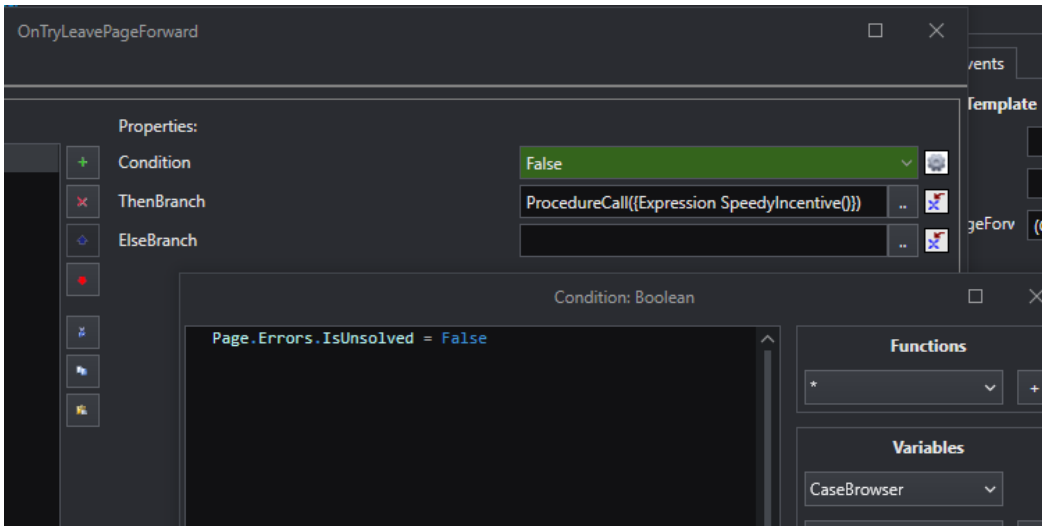
Figure 4a. Incentive Delivery Options



◯ By email
◯ By text
◯ By mail as a paper check
◯ Decline the $10

After selecting one of the digital incentive methods, the respondent is routed to a screen where their email address or cell phone number is collected, depending on which delivery mode was selected. This is followed by the appropriate screen routing to confirm the email or cell phone number, which also serves as verification of consent for this type of contact. On these confirmation screens, a special "SpeedyIncentive" template is used to trigger the actions setup process from the OnTryLeavePageForward event. On the screen immediately following the confirmation screens, the "Back" button is removed to prevent respondents from backing up and retriggering the actions setup.

Figure 4b. The SpeedyIncentive Process is Triggered in the "OnTryLeavePageForward" Event of a Special Master Page Template



OnTryLeavePageForward

Properties:

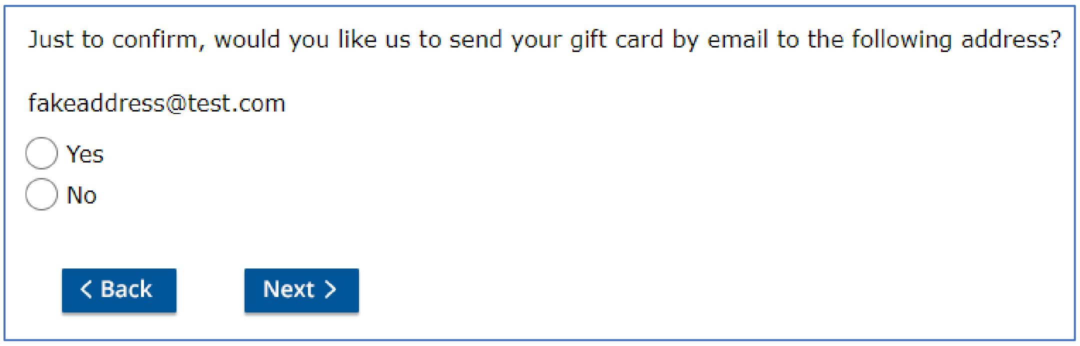
Condition	False
ThenBranch	ProcedureCall({Expression SpeedyIncentive()})
ElseBranch	

Condition: Boolean

```
Page.Errors.IsUnsolved = False
```

> back to Table of Contents

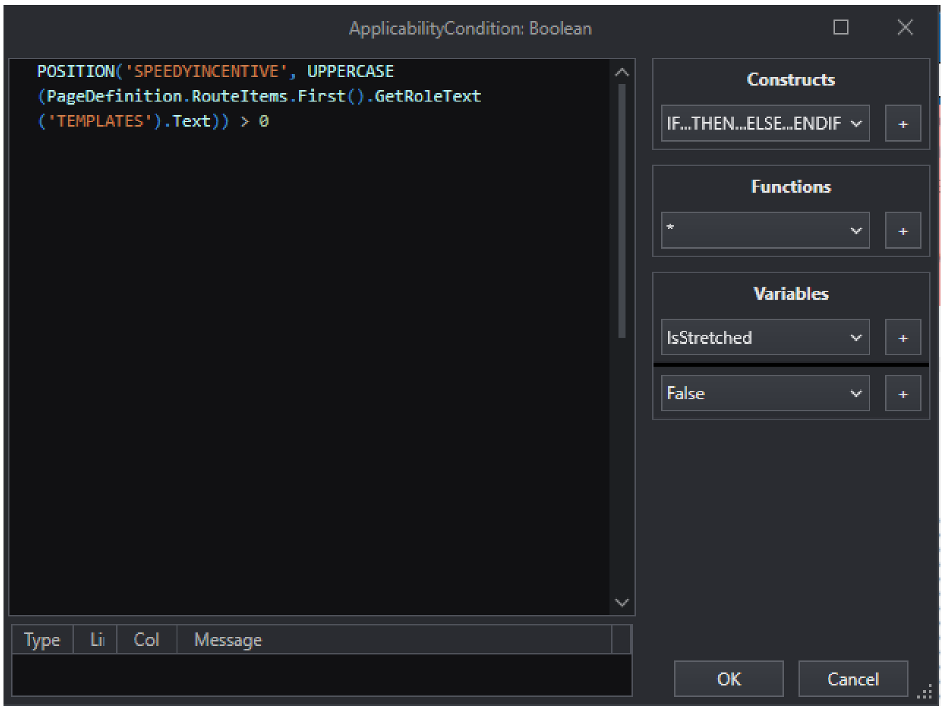
Figure 4c. Confirmation of Email Address After Email Incentive Is Selected; When the Respondent Selects “Yes” and Clicks “Next,” the SpeedyIncentive Process Is Triggered


Just to confirm, would you like us to send your gift card by email to the following address?

fakeaddress@test.com

The speedy incentive template is automatically assigned to appropriate fields using an applicability condition on the template. Each field that should trigger a speedy incentive has “SpeedyIncentive” in the Templates role.

Figure 4d. This Applicability Condition Ensures That All Fields with “SpeedyIncentive” Included in the Templates Role Are Assigned the Speedy Incentive Template


ApplicabilityCondition: Boolean

```
POSITION('SPEEDYINCENTIVE', UPPERCASE
(PageDefinition.RouteItems.First().GetRoleText
('TEMPLATES').Text)) > 0
```

Constructs: IF...THEN...ELSE...ENDIF

Functions: *

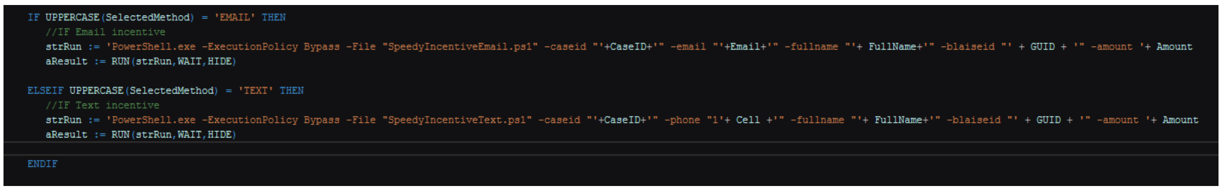
Variables: IsStretched

False

The “SpeedyIncentive” process is in the “Actions Setup” file defined in the project settings and mapped using Blaise 5’s “Mappings Management” screen. The process pulls the relevant information from the active survey using the SURVEYRECORD file definition. This information is then passed to the PowerShell scripts, as shown in Figure 4e.

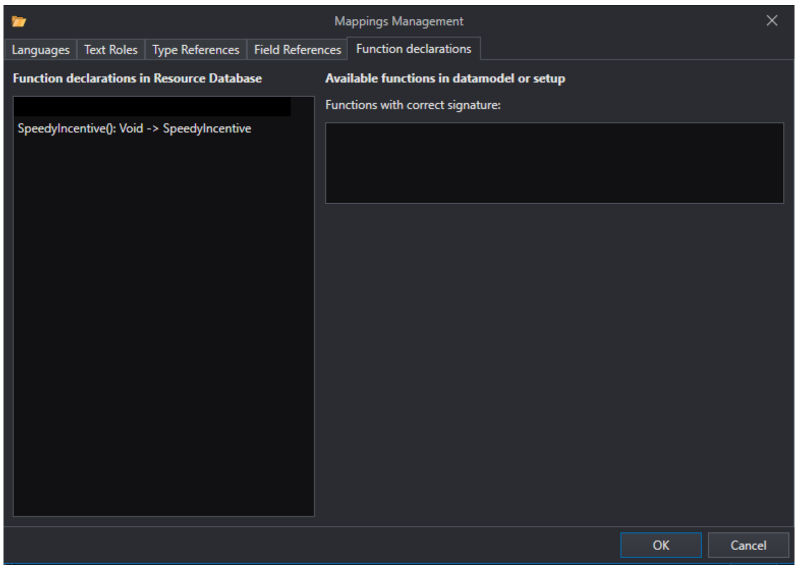
> back to Table of Contents

Figure 4e. Code in the Actions Setup Process That Calls the PowerShell Scripts



```
IF UPPERCASE(SelectedMethod) = 'EMAIL' THEN
   //IF Email incentive
   strRun := 'PowerShell.exe -ExecutionPolicy Bypass -File "SpeedyIncentiveEmail.ps1" -caseid "'+CaseID+'" -email "'+Email+'" -fullname "'+ FullName+'" -blaiseid "' + GUID + '" -amount '+ Amount
   aResult := RUN(strRun,WAIT,HIDE)

ELSEIF UPPERCASE(SelectedMethod) = 'TEXT' THEN
   //IF Text incentive
   strRun := 'PowerShell.exe -ExecutionPolicy Bypass -File "SpeedyIncentiveText.ps1" -caseid "'+CaseID+'" -phone "1'+ Cell +'" -fullname "'+ FullName+'" -blaiseid "' + GUID + '" -amount '+ Amount
   aResult := RUN(strRun,WAIT,HIDE)

ENDIF
```

Figure 4f. The Process Must Be Mapped to a Function Declaration in the Resource Database to Be Called from the Appropriate Template



Function declarations in Resource Database

[redacted]

SpeedyIncentive(): Void -> SpeedyIncentive

Available functions in datamodel or setup

Functions with correct signature:

The PowerShell script then makes the final call to the stored procedure that will insert an entry to a specific SQL database in the FIPS Mod network. The information added to the table includes case ID with attached language indicator (surveys conducted in multiple languages aren't an issue), confirmed email or cell phone number, a placeholder value for the full name (since the resulting text message or email does not include the respondent's name in this case), the Blaise GUID (for identification purposes since the table is shared with other surveys), and the amount of the incentive to be sent. The Listener Application will pick up new incentive information from this SQL table and kick off the rest of the process discussed in Section 3, "High-Level Overview of the Speedy Incentives Process."

5. Challenges and Lessons Learned

5.1 Initial Setup and Testing Challenges

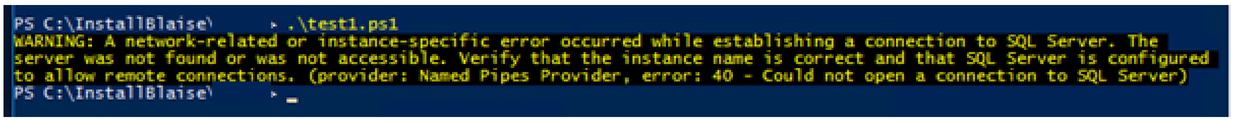
Initial challenges in the FIPS Low environment where we conducted testing included getting an appropriate PowerShell script programmed to trigger the insert into the stored procedure. We iterated through a number of test attempts while tweaking the call, both to adjust the syntax in general and to account for passing new parameters when new information was found to be needed in the incentives database—namely, for language indicator and GUID.

We also had trouble with the mapping of the actions setup function. Blaise 5 appeared to drop the mappings frequently, so we consistently checked to make sure it was still mapped correctly and remapped when needed.

> back to Table of Contents

After moving programs into the FIPS Mod network, the first set of tests for our new process was unsuccessful. Once we had confirmed everything in Blaise was running as expected as part of troubleshooting, we tested the PowerShell script by running it manually. Attempts to run the PowerShell script manually resulted in the error shown in Figure 5a. The firewall required modification to allow communication between the server hosting the Blaise instrument and the server hosting the SQL Server. After proper ports were opened, the process was successful.

Figure 5a. The Error Message Displayed When Initially Running the PowerShell Script from the Webserver to Insert into the SQL Table for Speedy Incentives



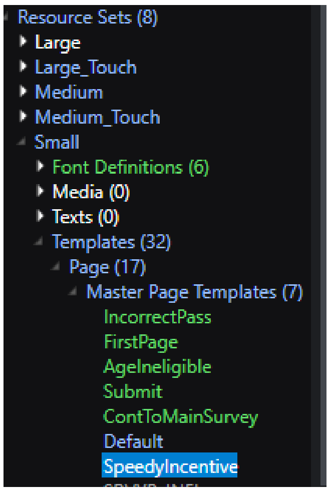
```
PS C:\InstallBlaise\       > .\test1.ps1
WARNING: A network-related or instance-specific error occurred while establishing a connection to SQL Server. The
server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured
to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)
PS C:\InstallBlaise\       > _
```

5.2 The Importance of Thoroughly Testing All Layout Sets

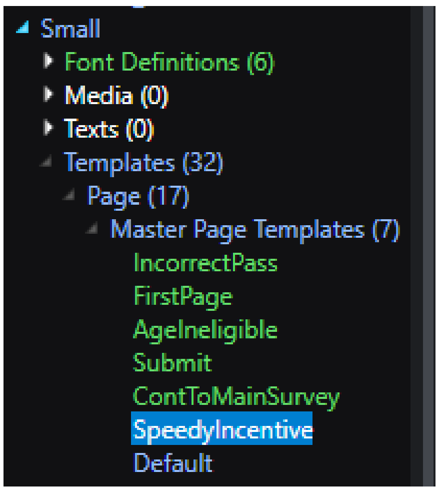
One of the most important lessons learned from implementing the speedy incentive process is that *all* layout sets need to be thoroughly tested for speedy incentive functionality. Our process relied on the PowerShell script being triggered by a function associated with a specific template. Without that template, the entire house of cards would collapse, and the speedy incentives wouldn't go out.

Initial instrument testing was heavily focused on the large layout set due to 508-compliance needs and testing out special text and functionality related to CATI mode. The small layout set was tested, but fewer test cases went through the small layout set, and unfortunately, it appeared most testers did not select digital incentive modes. Shortly after launch, it was discovered that surveys completed in the small layout set were not receiving digital incentives. While implementing the speedy incentive template in the small layout set, there was an issue with hierarchy that originally went unnoticed. The speedy incentive template was below the default Master Page template, which led to screens that should have had the speedy incentive template receiving the default Master Page template instead. Incentives are not triggered on the default Master Page template, which meant these small layout set cases were missing a key step in the speedy incentive process.

Figure 5b. On the Left Is an Image of the Incorrect Hierarchy for Master Page Templates; This Hierarchy Allowed the Assignment of the "Default" Master Page Template Prior to the Applicability Condition on the SpeedyIncentive Template Being Considered; The Image on the Right Shows the Correct Hierarchy for the Master Page Templates



```
Resource Sets (8)
  Large
  Large_Touch
  Medium
  Medium_Touch
  Small
    Font Definitions (6)
    Media (0)
    Texts (0)
    Templates (32)
      Page (17)
        Master Page Templates (7)
          IncorrectPass
          FirstPage
          AgeIneligible
          Submit
          ContToMainSurvey
          Default
          SpeedyIncentive
```



```
Small
  Font Definitions (6)
  Media (0)
  Texts (0)
  Templates (32)
    Page (17)
      Master Page Templates (7)
        IncorrectPass
        FirstPage
        AgeIneligible
        Submit
        ContToMainSurvey
        SpeedyIncentive
        Default
```

> back to Table of Contents

As soon as this issue was discovered, an update was released with this slight, but very important, error correction. Yet another lesson for future development: make sure to double check the assigned template for key fields when functionality is dependent on the correct template.

Once the error was corrected, queries were run against the survey data and speedy incentives dataset to pinpoint cases that had elected to receive a speedy incentive but had no entry in the speedy incentives table. We manually triggered incentives for these cases by constructing appropriate calls to the PowerShell script to insert the data into the speedy incentives table. The queries used to find cases missing from the incentives table became part of future, regular quality check routines to ensure that the speedy incentives process continued to work as expected once the small layout set was fixed.

Another lesson learned from this mistake is that the number of respondents using mobile devices to complete the survey was far higher than expected. Based on the number of missing incentives, over 50% of cases were being completed on a mobile/small screen device. Future surveys should include an equal amount of testing between the large and small layout sets.

5.3 Consider Carefully When Events Are Executed

After the correction was made for the small layout sets not receiving incentives, another issue was discovered. When respondents attempted to move forward without selecting a response on the email address or cell phone number confirmation screens, a double entry was made in the speedy incentives table. This issue stemmed from a misunderstanding of where the OnTryLeavePageForward event was called. It was assumed that this event was not called until just before the page was left, but it was also triggering as soon as the "Next" button was clicked. This resulted in the speedy incentives process being called twice in situations when an error occurred on the page. An entry for the case was inserted when the error was triggered, and then inserted a second time when the respondent corrected the issue and was able to move forward.

Fortunately, the backend processes ensured that only one incentive was sent per case in these situations. To correct this issue, another build was released with a conditional statement checking to make sure no errors existed on the page prior to calling the speedy incentives process. The corrected OnTryLeavePageForward event is shown in Figure 4b. The conditional statement was added to make sure the SpeedyIncentive process was not triggered until any errors on the page got resolved.

6. Future Uses and Potential Enhancements

The backend processes involved (refer to Section 3) were set up to handle receiving entries into the speedy incentives database not only from Blaise software, but also from other survey data collection systems utilized by RTI International, which makes the process flexible enough to serve a wide array of projects.

We foresee future projects making use of this speedy incentive process, as it adds a new layer to the methodologies behind incentives by rewarding the respondent with a form of immediate gratification. Our survey included a household screener component followed by a selected respondent survey, so we felt getting the incentive into the hands of the screener participant ASAP would be highly encouraging for participation in the survey component.

Situations where sending other timely digital communications to respondents during or immediately after their survey would be useful could also be considered for this process. A couple of examples: (1) specific

> back to Table of Contents

reminders to the respondent or someone in their household for completing additional surveys; (2) handing another portion of the survey off to a coworker as part of a business survey where different sections are completed by various individuals. As clients approach us with challenging criteria for complex surveys, we will have this process available in our toolbox.

7. Conclusion

We are continuing to monitor this new speedy incentives feature and will be analyzing results of its use. For example, it will be interesting to review survey component participation rates for cases where the screener respondent opted for a digital incentive vs. a mailed incentive or no incentive. Lessons we learned will surely be applied to future projects that implement this feature.

> back to Table of Contents

The Many Faces of F1

Siu Chong Wan and Sheba Ephraim, Westat

Presenter: Siu Chong Wan

1. Abstract

The display of help text to interviewers or respondents, depending on the mode of collection, is an essential feature in Blaise. While we used function keys (F1) for online help in Blaise 4, we have additional options in Blaise 5.

In addition, we can use the same techniques to display other materials, like showcards, that we often need in interviewer-administered surveys.

However, what we are going to explore here is not limited to interviewer-administered surveys.

This paper discusses our use of different triggers in Blaise 5 and the types of actions that support the needs of displaying supplementary survey materials.

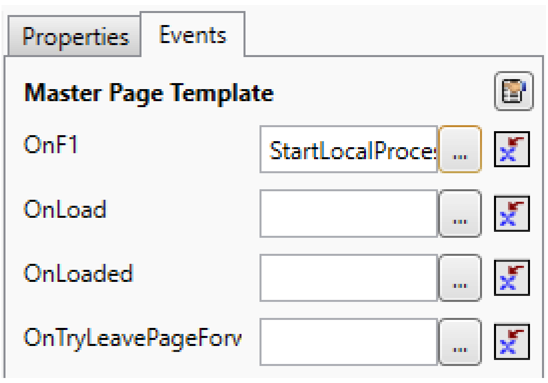
2. Triggers

There are a variety of ways for the users to access the help text. While traditionally trained interviewers might still favor keyboard shortcuts, the growing availability of touchscreen devices makes buttons and hyperlinks more attractive. We will discuss the triggers that we have tried so far.

2.1 Keyboard Shortcuts

Be it the F1 key again, or any other function key or shortcut key, we can use template events to define actions that display help text.

This is an example of the "OnF1" event defined in a Master Page Template.



Master Page Template

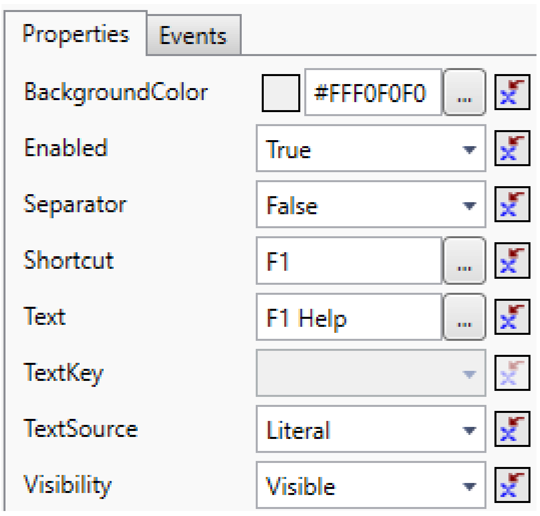
OnF1	StartLocalProce:
OnLoad	
OnLoaded	
OnTryLeavePageForv	

2.2 Menu Items

Using a menu control, we can create menu item events to define actions that display help text. In addition, we can use the shortcut property in a menu item to define an accelerator key that works just like the abovementioned keyboard shortcuts.

> back to Table of Contents

This is an example of the F1 key defined in the shortcut property of a menu item.



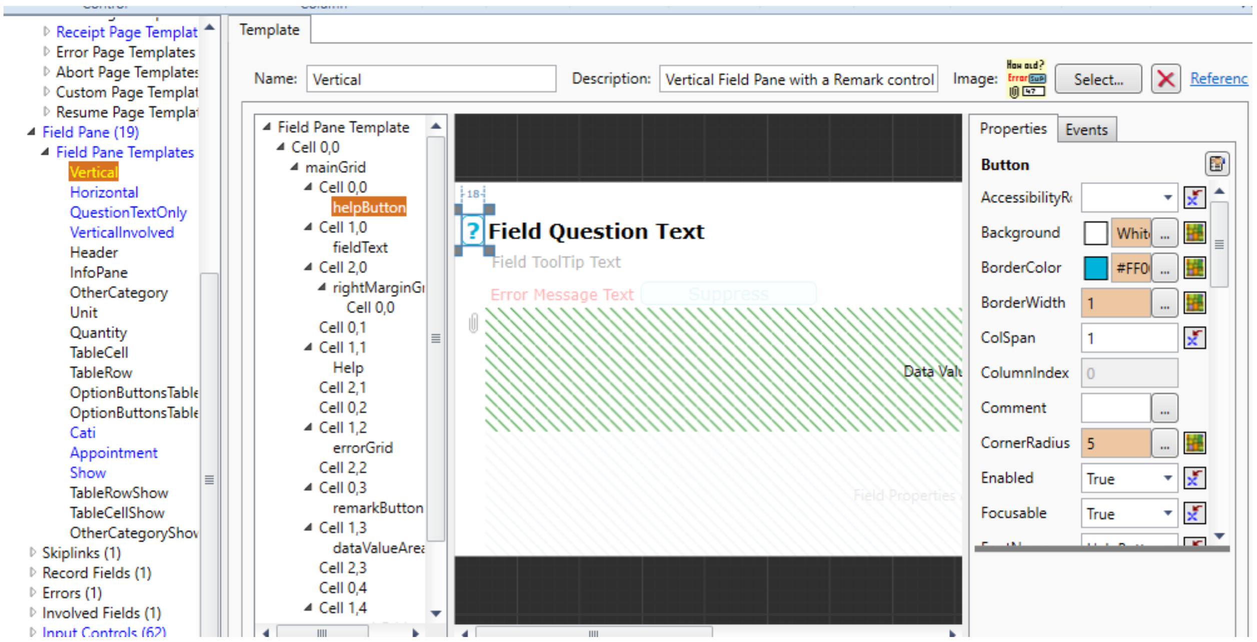
Properties | Events

BackgroundColor	#FFF0F0F0
Enabled	True
Separator	False
Shortcut	F1
Text	F1 Help
TextKey	
TextSource	Literal
Visibility	Visible

2.3 Buttons

Buttons might be one of the most versatile ways to show help text. Depending on what kind of template the button resides in, we can display help text associated with a page, a field, a response category, or even a table column header.

This is an example of a button shown as a blue "?" that triggers help text in a Field Pane Template.



- Receipt Page Templat
- Error Page Templates
- Abort Page Templates
- Custom Page Templat
- Resume Page Templat
- Field Pane (19)
  - Field Pane Templates
    - Vertical
    - Horizontal
    - QuestionTextOnly
    - VerticalInvolved
    - Header
    - InfoPane
    - OtherCategory
    - Unit
    - Quantity
    - TableCell
    - TableRow
    - OptionButtonsTable
    - OptionButtonsTable
    - Cati
    - Appointment
    - Show
    - TableRowShow
    - TableCellShow
    - OtherCategoryShow
- Skiplinks (1)
- Record Fields (1)
- Errors (1)
- Involved Fields (1)
- Input Controls (62)

Template

Name: Vertical Description: Vertical Field Pane with a Remark control Image: Select... Referenc

- Field Pane Template
  - Cell 0,0
    - mainGrid
      - Cell 0,0
        - helpButton
      - Cell 1,0
        - fieldText
      - Cell 2,0
        - rightMarginGr
          - Cell 0,0
      - Cell 0,1
      - Cell 1,1
        - Help
      - Cell 2,1
      - Cell 0,2
      - Cell 1,2
        - errorGrid
      - Cell 2,2
      - Cell 0,3
        - remarkButton
      - Cell 1,3
        - dataValueArea
      - Cell 2,3
      - Cell 0,4
      - Cell 1,4

Properties | Events

Button

AccessibilityR	
Background	Whit
BorderColor	#FF0
BorderWidth	1
ColSpan	1
ColumnIndex	0
Comment	
CornerRadius	5
Enabled	True
Focusable	True

2.4 Hyperlink Tag

The hyperlink tag supports the OnClick attribute, where we can define actions that display help text. This can be defined in the field text alone in the FIELDS section of the data model without defining anything in the resource database.

> back to Table of Contents

This is an example of a hyperlink defined in the question text.

```
Q2 (Q2)
"This is an example for online help.<right>
<hyperlink Name=\"helplink\" Text=\"HELP\"
   OnClick=\"{Action
      StartLocalProcess('C:\\Windows\\hh.exe',
         'mk:@MSITStore:C:\\Help\\Help.chm::/RespondentHelp.htm',
         '',True,Normal)}\"
>Help</hyperlink></right>"
: (OK)
```

3. Actions

There are different actions to display extra reading materials. Here, we focus on ways to open local files, web pages, and simply extra field text in the current page. While local files can be accessed by the StartLocalProcess action, web pages can be accessed by the GotoUri action. If we want to remain on the current page the whole time, we simply hide and show field text.

3.1 StartLocalProcess Action

The StartLocalProcess action introduced in Blaise 5.13 provides a lot of possibility in Blaise 5. As long as we can find a local program that can display additional materials, we can use it. Here are a few examples:

To display HTML-Help with topic identifier:

{Action StartLocalProcess('C:\\Windows\\hh.exe', 'mk:@MSITStore:C:\\Documents\\Help.chm::/RespondentHelp.htm', '', True, Normal)}

To display a JPEG file in MS Edge:

{Action StartLocalProcess('C:\\Program Files (x86)\\Microsoft\\Edge\\Application\\msedge.exe','--app=&quot;C:\\Documents\\ English_2022_PP_16_593x730.jpg&quot;', '', True, Normal)}

To display a PDF file in MS Edge at a particular page:

{Action StartLocalProcess('C:\\Program Files (x86)\\Microsoft\\Edge\\Application\\msedge.exe','--app=&quot;C:\\Documents\\Help\\LayoutDesigner_Tutorial_PartI.pdf#page=5&quot;', '', True, Normal)}

3.2 GotoUri Action

The GotoUri action is definitely the way to go when we need to access help text on a web page. As long as the respondent has access to the internet, we can display anything from a web source. On the other hand, it can also display local files. Here are a couple examples:

To go to a web page:

{Action GotoUri('https://www.westat.com/about-westat/')}

> back to Table of Contents

To go to a local file:

{Action GotoUri('C:\\Documents\\Help\\HTM\\RespondentHelp.htm')}

The advantage of maintaining the material in a web source is that the updates to the web source can be available immediately. This is particularly good for time-sensitive information. However, the requirement of internet access could also be a weakness because we do not always have internet access in interviewer-administered surveys. In other words, the lack of internet access is almost assumed in interviewer-administered surveys.

3.3 ToggleVisibility Action

One thing we often try to avoid in web surveys is popups. Since we cannot control web respondents' browsers and cannot disable popup blockers, we usually try to display help text in the current page. In this case, we do not start any local process or go to any URI. We simply create a text role for the help text and use the ToggleVisibility action to show and hide the role text.

The predefined text roles "Help" and "ToolTip" are handy for this purpose, but additional text roles can be defined for additional text displaying needs.

In the default Blaise 5 resource database, a "Help" text role is already defined to work with the "helpButton" in the Vertical Field Pane Template to show/hide the help text.

```
<TextRole>
  <Name>Help</Name>
  <Description>A comprehensive instruction for the respondent</Description>
  <TextRoleType>UserDefined</TextRoleType>
</TextRole>
```

```
<Button Name="helpButton" ... OnClick="{Action
ToggleVisibility('Help')}">
```

4. Examples

We will discuss in more detail how things work in a few examples.

4.1 StartLocalProcess Action Working with a Menu Bar Item

We first define the text role "HelpFile" in both the resource database and the data model for the name of the file that stores the help text associated with the field. This will serve as the topic identifier in HTML-Help.

```
<TextRole>
  <Name>HelpFile</Name>
  <Description>The help text file name.</Description>
  <TextRoleType>UserDefined</TextRoleType>
</TextRole>
```

```
Q1 (Q1)
"This is an example of showing help text with Menu (F1 Help). Please click the menu bar item or press F1.<br>"
HelpFile "LIVEUSHelp"
: (OK)
```

> back to Table of Contents

Then we define a menu item in the Master Page Template.

```
<MenuItem Text="Help F1" Shortcut="F1" Visibility="{Expression IF
Page.ActiveField.RoleTextExists('HelpFile') = True THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action Conditional({Expression
Page.ActiveField.RoleTextExists('HelpFile') = True},{Action
StartLocalProcess('C:\\Windows\\hh.exe',{Expression
'mk:@MSITStore:C:\\Documents\\Help.chm::/' +
Page.ActiveField.GetRoleText('HelpFile') +
'.htm'},'',True,Normal)},'')}" />
```

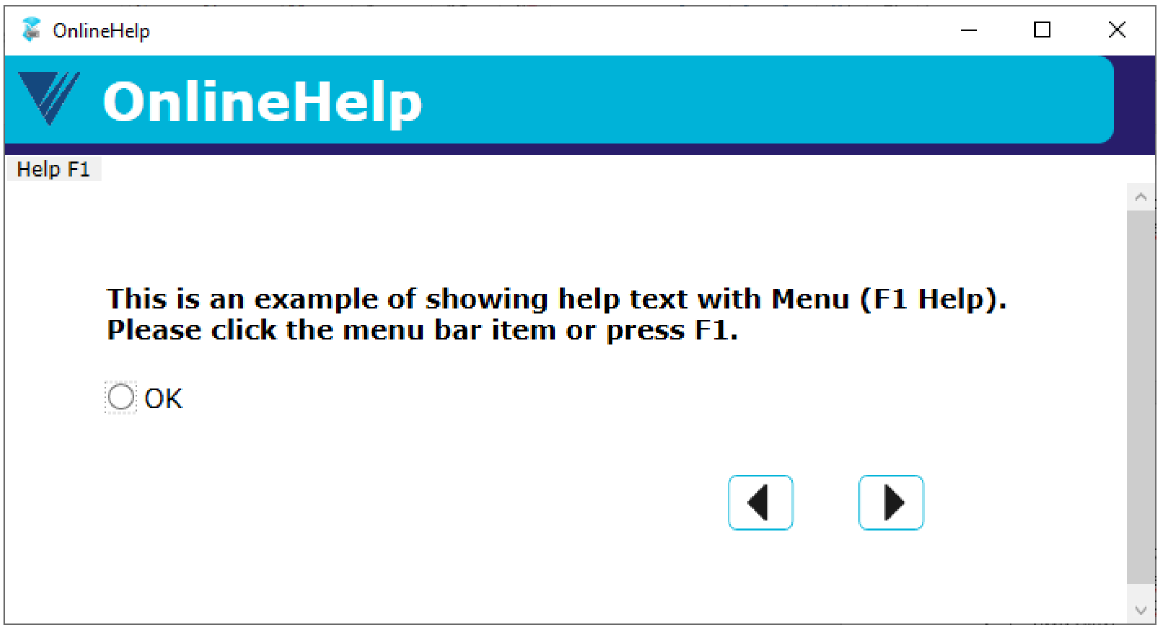
We use “F1” in its shortcut property so that both the menu bar and the F1 key work the same way.

We control its visibility based on whether the active field has “HelpFile” role text defined.

In the OnClick event, we use the StartLocalProcess action to run the local program “hh.exe” to open the HTML-Help file, “Help.chm,” at the page defined in the “HelpFile” role text.

In this example, we define the HelpFile role text as “LIVEUSHelp” so that the StartLocalProcess will run the local hh.exe program to open the topic identifier “LIVEUSHelp” in the C:\Documents\ Help.chm file.

As a result, the Q1 page shows “Help F1” on the menu bar.

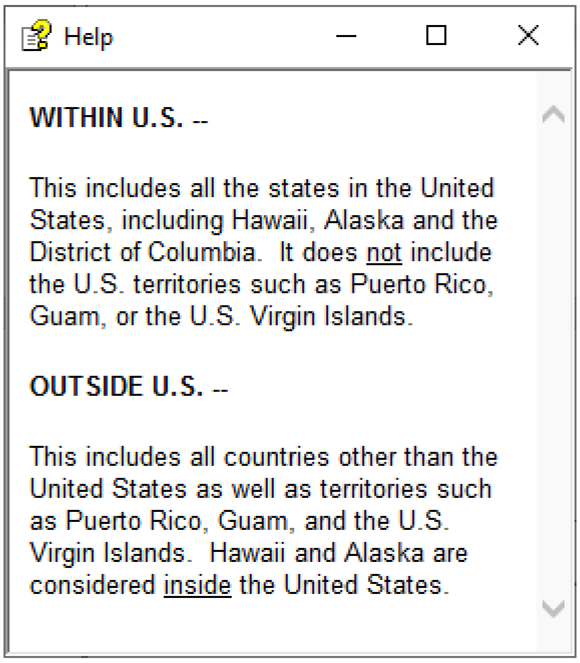


This is an example of showing help text with Menu (F1 Help).
Please click the menu bar item or press F1.

◯ OK

By either clicking the menu bar item “Help F1” or pressing the F1 key, the help window pops up with the definition of living within or outside the U.S.

> back to Table of Contents



WITHIN U.S. --

This includes all the states in the United States, including Hawaii, Alaska and the District of Columbia. It does not include the U.S. territories such as Puerto Rico, Guam, or the U.S. Virgin Islands.

OUTSIDE U.S. --

This includes all countries other than the United States as well as territories such as Puerto Rico, Guam, and the U.S. Virgin Islands. Hawaii and Alaska are considered inside the United States.

4.2 StartLocalProcess Action Working with a Keyboard Shortcut

Continuing with what we built above, we add the text role "HelpPage" for the page we want to show in a PDF file.

In this example, we use page number 30.

```
Q3 (Q3)
"This is an example of showing help text in a certain page of a PDF file in Edge with the F3 key. Please press F3.<br>"
HelpPage "30"
: (OK)
```

We add the shortcut key F3 to the Vertical Field Pane Template.

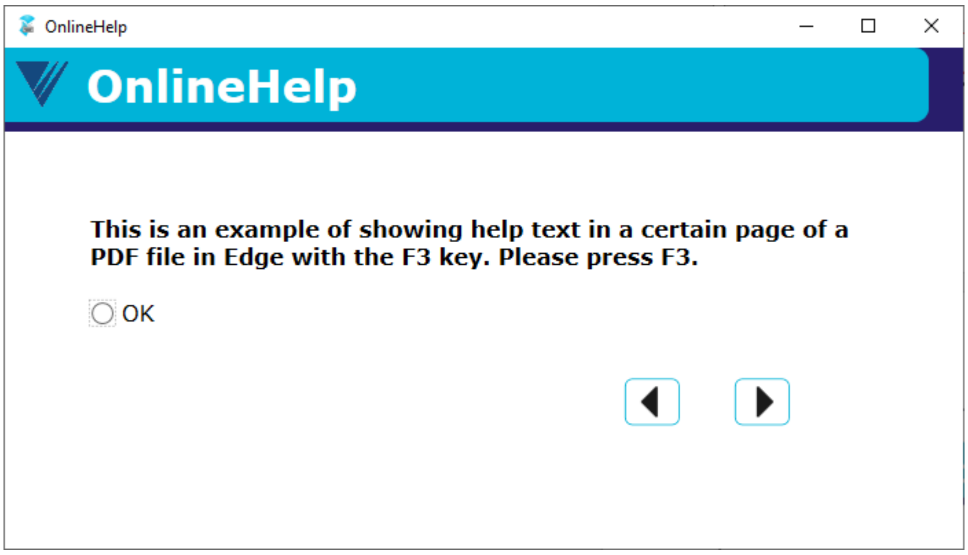
```
<Shortcut Key="F3" OnExecute="{Action Conditional({Expression
Field.RoleTextExists('HelpPage') = True},{Action
StartLocalProcess('C:\\Program Files
(x86)\\Microsoft\\Edge\\Application\\msedge.exe',{Expression '--
app=&quot;C:\\Documents\\Help\\ Blaise.pdf#page=' +
Page.ActiveField.GetRoleText('HelpPage') +
'&quot;'},'',True,Normal)},'')}" />
```

This time, in the OnF3 event, we use the StartLocalProcess action to run the local program "msedge.exe" to open a PDF file in MS Edge at the page defined in the "HelpPage" role text.

In this example, we are showing the Blaise 5 Manual file, Blaise.pdf.

As a result, this page does not need to display any buttons or the menu bar.

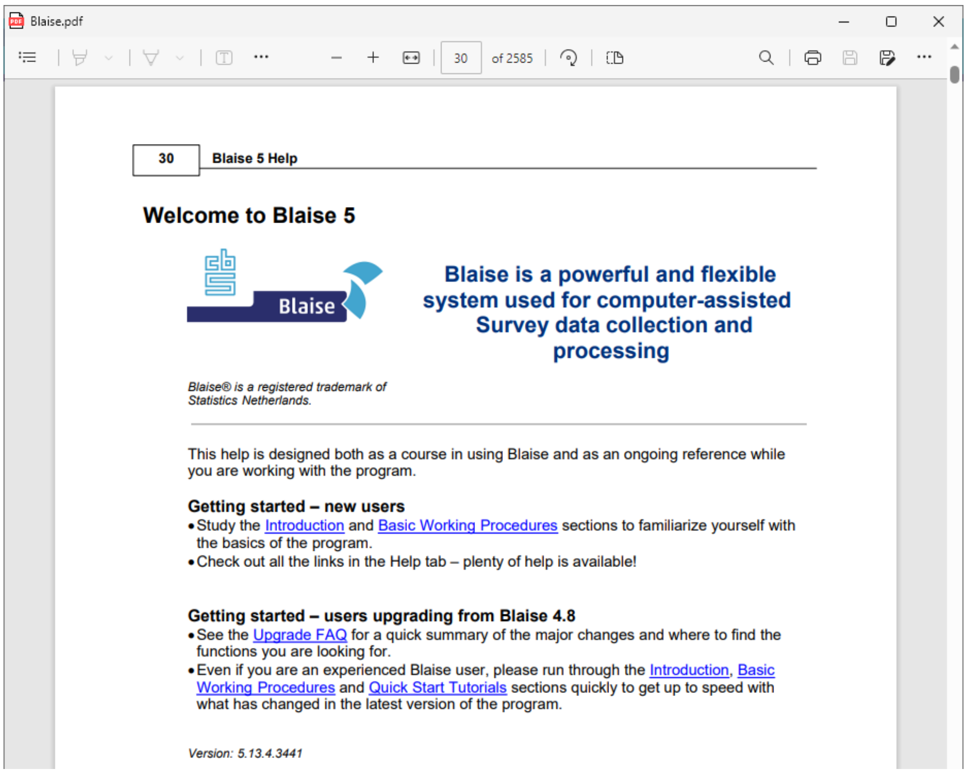
> back to Table of Contents



This is an example of showing help text in a certain page of a PDF file in Edge with the F3 key. Please press F3.

◯ OK

When we press F3, MS Edge pops up to display page 30 of the Blaise 5 Manual.



Welcome to Blaise 5

Blaise is a powerful and flexible system used for computer-assisted Survey data collection and processing

Blaise® is a registered trademark of Statistics Netherlands.

This help is designed both as a course in using Blaise and as an ongoing reference while you are working with the program.

Getting started – new users
- Study the Introduction and Basic Working Procedures sections to familiarize yourself with the basics of the program.
- Check out all the links in the Help tab – plenty of help is available!

Getting started – users upgrading from Blaise 4.8
- See the Upgrade FAQ for a quick summary of the major changes and where to find the functions you are looking for.
- Even if you are an experienced Blaise user, please run through the Introduction, Basic Working Procedures and Quick Start Tutorials sections quickly to get up to speed with what has changed in the latest version of the program.

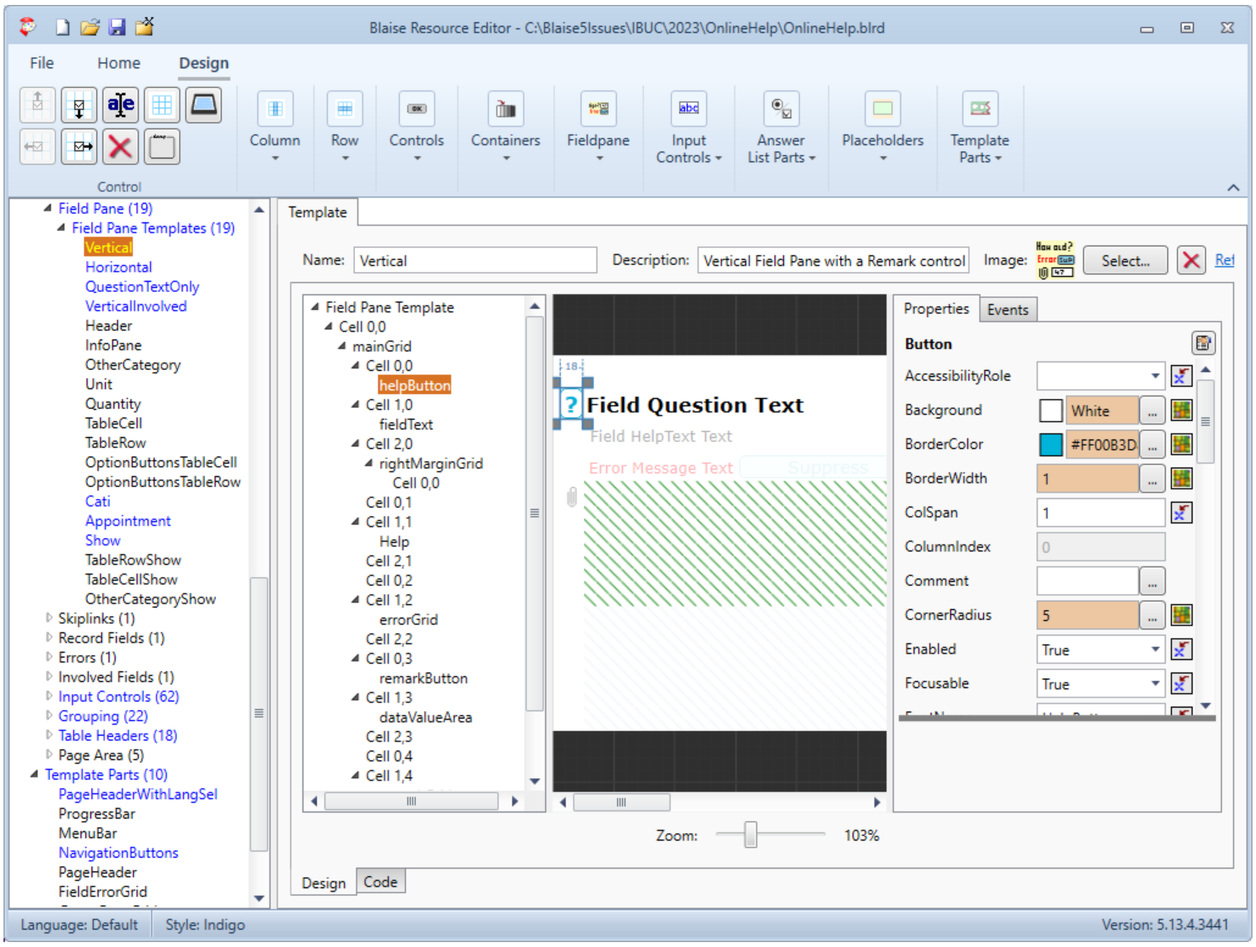
Version: 5.13.4.3441

4.3 Button to Toggle Role Text Visibility

In web surveys, where we cannot start a local process and in general avoid popup windows, role text fits the purpose of showing help text. We usually use a button to toggle the visibility of the role text so that the user can choose to show or hide the role text.

> back to Table of Contents

In the resource database, we define a “helpButton” in the Field Pane Template where its visibility is determined by whether the field’s role text “HelpText” exists. Its OnClick event simply shows or hides the HelpText.


Blaise Resource Editor - C:\Blaise5Issues\IBUC\2023\OnlineHelp\OnlineHelp.blrd

Name: Vertical Description: Vertical Field Pane with a Remark control

Field Pane Template
- Cell 0,0
  - mainGrid
    - Cell 0,0
      - helpButton
    - Cell 1,0
      - fieldText
    - Cell 2,0
      - rightMarginGrid
        - Cell 0,0
    - Cell 0,1
    - Cell 1,1
      - Help
    - Cell 2,1
    - Cell 0,2
    - Cell 1,2
      - errorGrid
    - Cell 2,2
    - Cell 0,3
      - remarkButton
    - Cell 1,3
      - dataValueArea
    - Cell 2,3
    - Cell 0,4
    - Cell 1,4

Button	
AccessibilityRole	
Background	White
BorderColor	#FF00B3D
BorderWidth	1
ColSpan	1
ColumnIndex	0
Comment	
CornerRadius	5
Enabled	True
Focusable	True

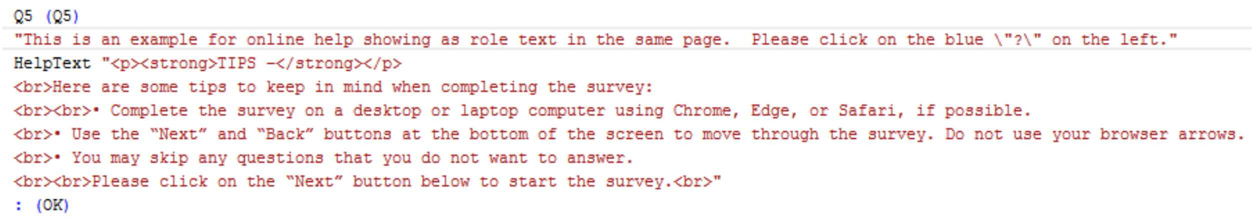
Zoom: 103%

Language: Default Style: Indigo Version: 5.13.4.3441

```
<Button Name="helpButton" Visibility="{Expression IF
LEN(Field.GetRoleText('HelpText')) = 0 THEN 'Hidden' ELSE 'Visible'
ENDIF}" HorizontalAlignment="Center" Background="{Style
HelpButtonBackGround}" Margin="{Style HelpButtonMargin}" Width="{Style
HelpButtonWidth}" Height="25" ScreenReaderText="Help"
ToolTipFontName="ToolTip" OnMouseDown="{Action
SetControlProperty('helpButton','Background','Brush','{Style
ButtonBackgroundSelected}')}" OnMouseUp="{Action
SetControlProperty('helpButton','Background','Brush','{Style
HelpButtonBackGround}')}" Text="?" FontName="HelpButton"
ScreenReaderTextSource="Literal" BorderWidth="{Style
HelpButtonBorderWidth}" BorderColor="{Style HelpButtonBorderColor}"
CornerRadius="{Style HelpButtonCornerRadius}" OnClick="{Action
ToggleVisibility('Help')}">
```

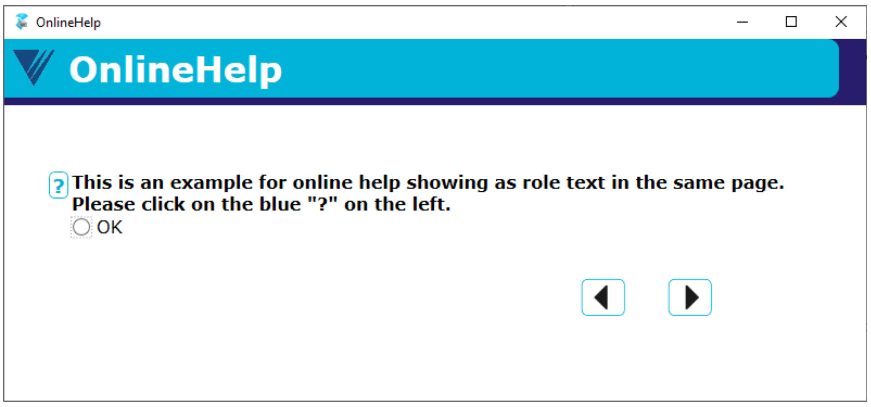
> back to Table of Contents

In the data model, we define the role text “HelpText” for the field.



```
Q5 (Q5)
"This is an example for online help showing as role text in the same page.  Please click on the blue \"?\" on the left."
HelpText "<p><strong>TIPS –</strong></p>
<br>Here are some tips to keep in mind when completing the survey:
<br><br>• Complete the survey on a desktop or laptop computer using Chrome, Edge, or Safari, if possible.
<br>• Use the “Next” and “Back” buttons at the bottom of the screen to move through the survey. Do not use your browser arrows.
<br>• You may skip any questions that you do not want to answer.
<br><br>Please click on the “Next” button below to start the survey.<br>"
: (OK)
```

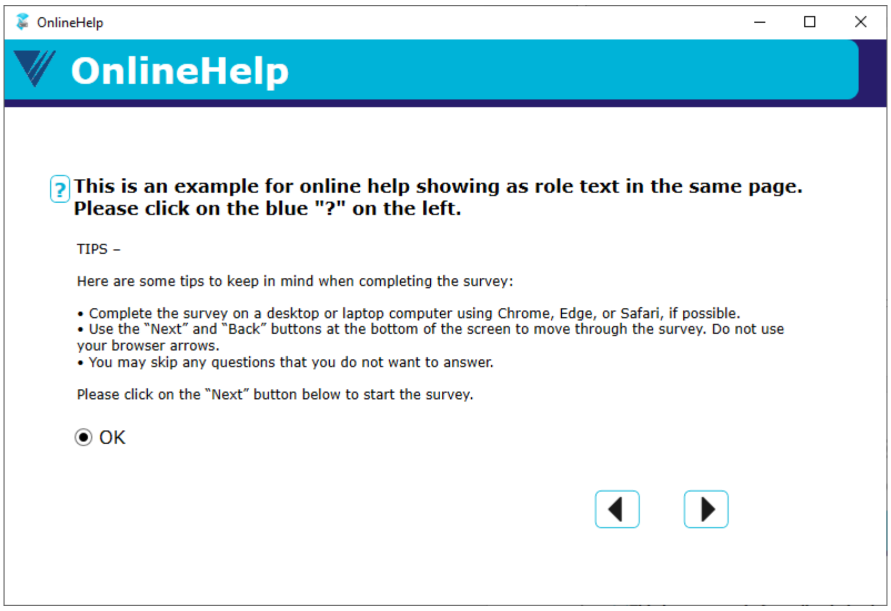
As a result, we have a blue “?” button to the left of the question text.



?This is an example for online help showing as role text in the same page.
Please click on the blue "?" on the left.

○ OK

When the blue “?” button is clicked, help text is displayed below the question text.



?This is an example for online help showing as role text in the same page.
Please click on the blue "?" on the left.

TIPS –

Here are some tips to keep in mind when completing the survey:

• Complete the survey on a desktop or laptop computer using Chrome, Edge, or Safari, if possible.
• Use the “Next” and “Back” buttons at the bottom of the screen to move through the survey. Do not use your browser arrows.
• You may skip any questions that you do not want to answer.

Please click on the “Next” button below to start the survey.

◉ OK

This is not limited to the field level. For instance, we can add the same help button and field text in a category template, and then add role text to the type library so that the help text becomes answer category level.

> back to Table of Contents

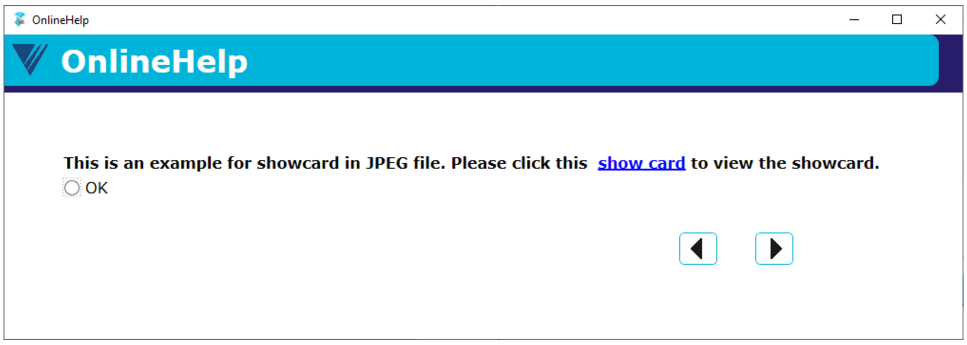
Although this is usually the way we show help text in web surveys, this can be applied to interviewer-administered surveys in the same way.

4.4 Hyperlink in Field Text to Display Showcard

In face-to-face surveys, showcards are often used to present information to respondents. Either we choose to present the showcards on our devices, or simply use that as the means for the interviewers to verify the correct showcard. There is value in having the ability to display the showcards along with the questions on the screen during the survey.

This time, we will do everything in the field text and not bother with the resource database.

```
Q6 (6)
"This is an example for showcard in JPEG file.  Please click this <blue>
<hyperlink
   OnClick=\"{Action
      StartLocalProcess('C:\\Program Files (x86)\\Microsoft\\Edge\\Application\\msedge.exe',
         '--app=C:\\Documents\\Showcards\\Showcards_English_2022_RE_3.jpg',
         '',True,Normal)}\"
>show card</hyperlink></blue> to view the showcard."
: (OK)
```

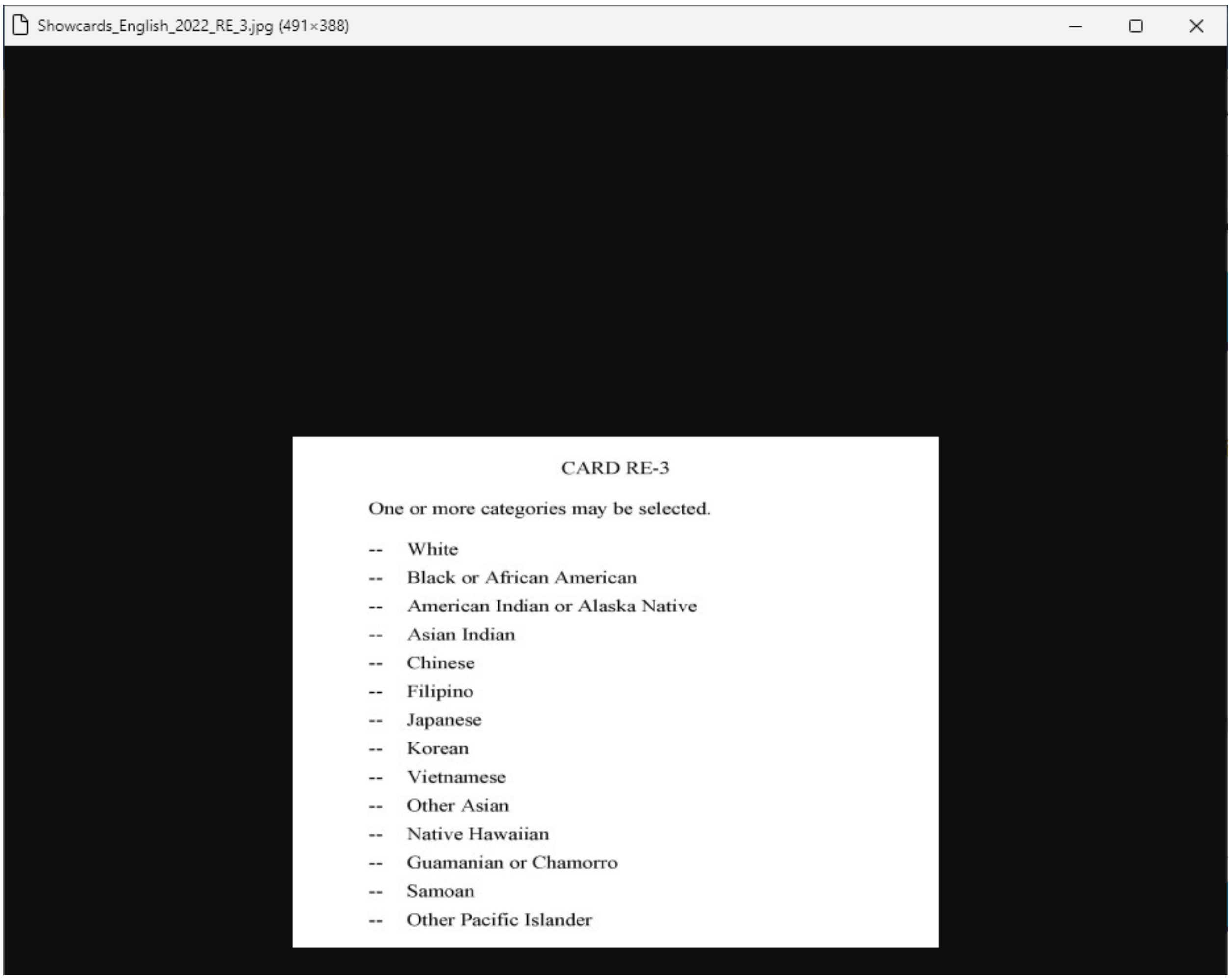


This is an example for showcard in JPEG file. Please click this show card to view the showcard.

◯ OK

> back to Table of Contents

Clicking the hyperlink triggers the StartLocalProcess action to run the local program “msedge.exe” to open a JPEG file in MS Edge.



CARD RE-3

One or more categories may be selected.

- -- White
- -- Black or African American
- -- American Indian or Alaska Native
- -- Asian Indian
- -- Chinese
- -- Filipino
- -- Japanese
- -- Korean
- -- Vietnamese
- -- Other Asian
- -- Native Hawaiian
- -- Guamanian or Chamorro
- -- Samoan
- -- Other Pacific Islander

5. Conclusions

We have seen many different ways to display supplementary materials.

If the needs are limited (e.g., only help text for a couple questions), we can easily program that in the field text without extra settings in the resource database.

However, for large surveys that need to show a lot of help text or showcards, maintaining all the external texts and images outside of Blaise would be more manageable. In that case, setting up text roles, events, and actions in the resource database up front would be a more efficient way to deal with it. For instance, we only need to use role text to determine the file name or page number in the field definition, and the resource database can take care of the rest.

It is worth noting that, as helpful as it is, the StartLocalProcess action works only in Windows, not in browsers.

> back to Table of Contents

On the other hand, in web surveys, the rule of thumb would still be avoiding popups, whether we can start a local process or not. In this case, using field text to show help text is more desirable. It should be noted that survey designers might need to beware of too much help text displayed on the page that becomes more of a distraction than a help.

> back to Table of Contents

Creating a Respondent Self-Scheduling Interface Using Blaise 5

Andrew D. Piskorowski, Peter Sparks, and Andrew L. Hupp, University of Michigan

1. Background

With refusals increasing and contact rates decreasing, researchers are having to expend more effort to reach people than ever before. When an interviewer makes contact, some portion of that time is spent finding a time when the person actually has the time required for the survey request. One possible way to minimize that effort is to have the person self-schedule a time that is convenient for them. Self-scheduling is common in daily life, for things like salon appointments, restaurant reservations, and automobile service appointments, so why not allow, at least for those who want to, an opportunity to set an appointment without involving an interviewer?

2. Scheduling System

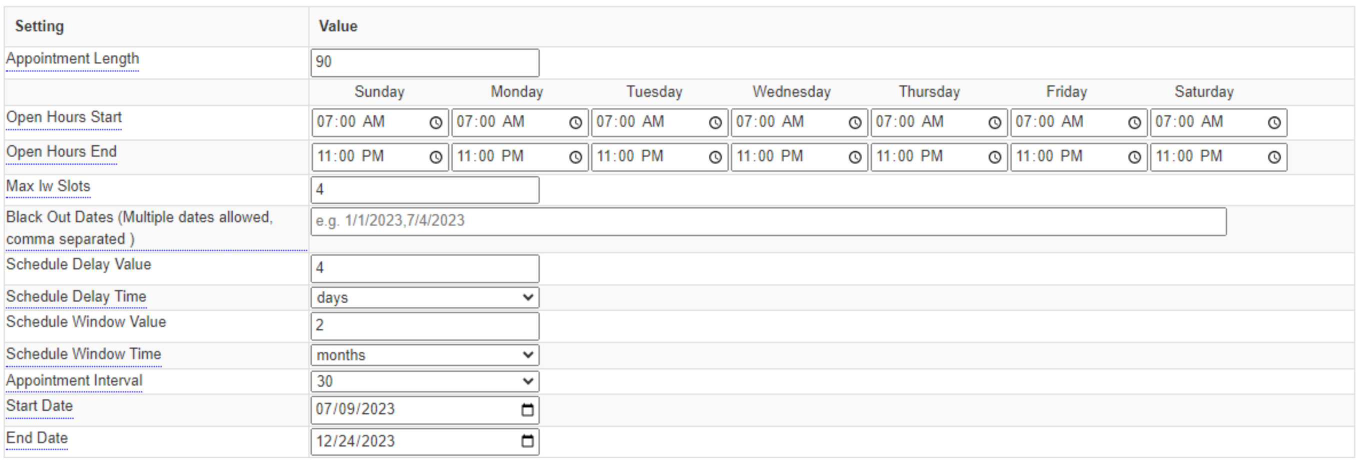
The University of Michigan has created a scheduling system that allows a person to schedule an appointment time that is convenient for them. The system has three main components:

1. An *appointment configuration settings* interface that allows the project to define parameters that are used to calculate appointment slots,
2. A *logon portal* that controls access to the self-scheduling interface (survey), and
3. A *Blaise datamodel* used as the self-scheduling interface.

2.1 Appointment Configuration Settings

There are 12 different appointment configuration settings a project defines (see Figure 1). The scheduling system uses the defined parameters to calculate available appointment slots for the respondent to select from.

Figure 1. Appointment Configuration Settings



Setting	Value						
Appointment Length	90						
	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Open Hours Start	07:00 AM	07:00 AM	07:00 AM	07:00 AM	07:00 AM	07:00 AM	07:00 AM
Open Hours End	11:00 PM	11:00 PM	11:00 PM	11:00 PM	11:00 PM	11:00 PM	11:00 PM
Max Iw Slots	4						
Black Out Dates (Multiple dates allowed, comma separated)	e.g. 1/1/2023,7/4/2023						
Schedule Delay Value	4						
Schedule Delay Time	days						
Schedule Window Value	2						
Schedule Window Time	months						
Appointment Interval	30						
Start Date	07/09/2023						
End Date	12/24/2023						

> back to Table of Contents

2.1.1 Flexible Time Slot Generation

The program initiates its operation by generating potential time slots based on the defined parameters. These parameters include the intervals (e.g., every 15, 30, 45, 60 minutes) displayed to the respondent specified by [Appointment Intervals], the daily start and end times of availability (specified as [Open Hours Start] and [Open Hours End]), and the allowable scheduling period defined by [Start Date] and [End Date].

2.1.2 Flexible Time Slot Generation

Each time interval generated is assigned a maximum number of interview slots, as specified by [Max Iw Slots]. This allocation accounts for the anticipated capacity for interviews within those intervals, ensuring efficient resource utilization.

2.1.3 Accounting for Existing Appointments

To provide an accurate representation of available slots, the program deducts the number of current appointments from the maximum slots available by matching interval time slots and all time slots within the next [Appointment Length] minutes. This mechanism guarantees that only genuinely open slots are presented to users, factoring in the expected duration of each survey.

2.1.4 Dynamic Date Range Selection

The program next dynamically determines the available date range for scheduling appointments. This range starts with a delay, as defined by [Schedule Delay Value] and [Schedule Delay Time]. For example, if today is October 1 and the delay values are [2] and [days], the first available time slots presented to users would begin on October 3. The program also limits the scheduling horizon by considering [Schedule Window Value] and [Schedule Window Time], ensuring that respondents can't schedule appointments too far in advance. For instance, if those values are [2] and [months] with the delay values above, the date range of dates available for time slots would be from October 3 to December 3. Different unit types are available, including days, weeks, and months.

2.1.5 Blackout Date Filtering

Lastly, the program ensures that any dates defined as [Black Out Dates] are filtered out from the available options, preventing respondents from scheduling appointments on these restricted dates. Additionally, it enforces all dates to fall within the project's specified [Start Date] and [End Date], maintaining alignment with project timelines.

Overall, these configuration settings, which are customizable for each project, empower project leaders to adapt and respond to changing interviewer availability and organizational needs. This systematic approach not only streamlines the scheduling process, but also enhances the user experience by providing a comprehensive and tailored set of options for survey completion.

3. Logon Portal

The portal is a .NET web application that collects the user's login credentials, performs checks, retrieves

> back to Table of Contents

current case status from the management system, and launches the Blaise self-scheduling survey if appropriate. It has been implemented in both English and Spanish (see Figure 2).

The usual method of entering through the portal is via a fully qualified URL. That is, the full web address and parameters are supplied. For example,

https://[TestingSite]/[SS_Survey]/logon.aspx?l=[loginID]&p=[pin]5&iwerid=[iwerid]&S=[SessionContact]

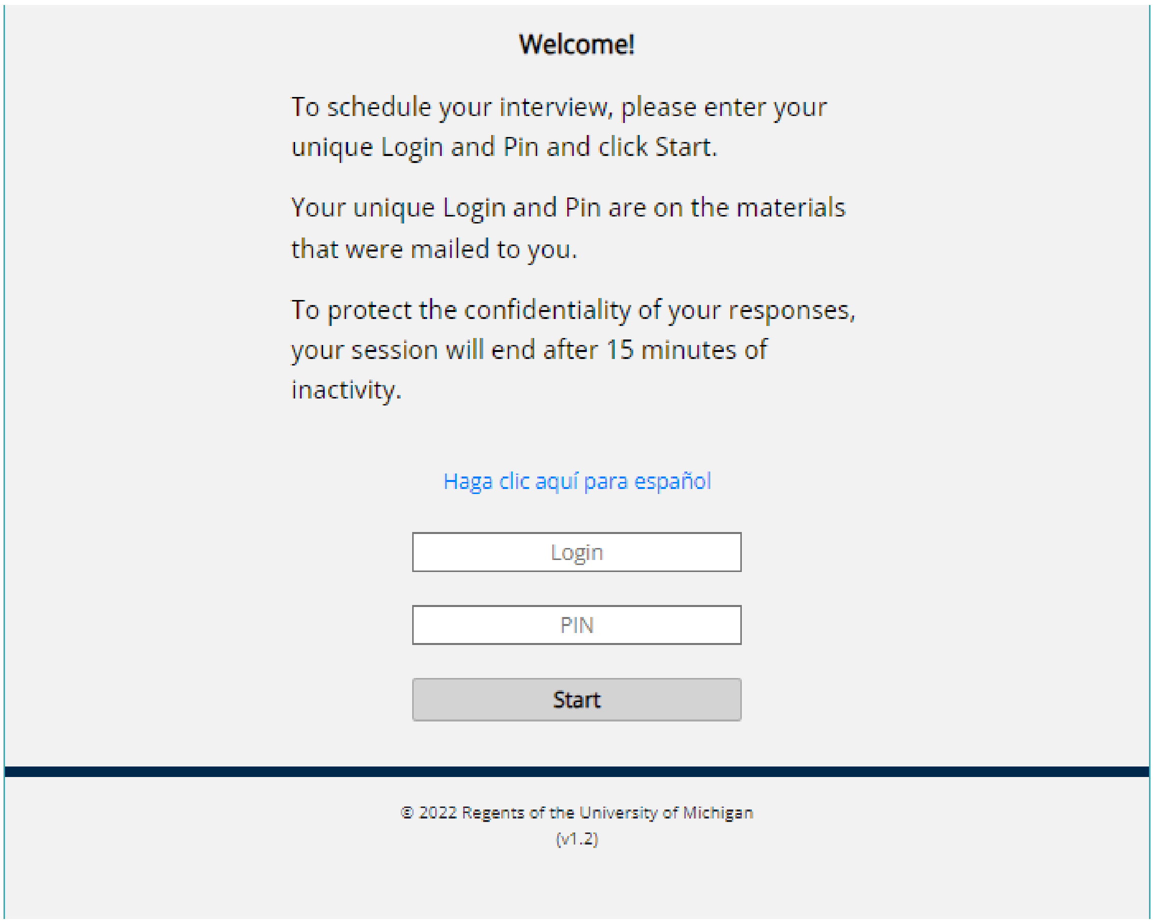
The portal has been designed so the respondent is taken to the logon site with the login credentials passed from the URL, but the respondent needs to manually click/tap the "Start" button in order to launch the self-scheduling survey.

After the user has entered their login credentials, or has had these credentials and other parameters automatically supplied via an email, QR code, or other means, the portal checks to see if the browser is supported (i.e., not too old) and that cookies and JavaScript have been enabled. If these checks pass, then the portal passes the supplied credentials to an authentication endpoint (that communicates with the management system) and receives back a status and the primary key (if appropriate) for the attempt:

- Authorized (login credentials are okay, and the self-scheduling survey is active and available)
- NotAuthenticated (wrong login credentials)
- Paused/Locked/Stopped/Canceled (self-scheduling survey is not available)
- Expired/Closed (study has been closed and is not available)
- NotAvailable (study is not available)

Figure 2. Scheduling System Logon Portal

> back to Table of Contents


Welcome!

To schedule your interview, please enter your unique Login and Pin and click Start.

Your unique Login and Pin are on the materials that were mailed to you.

To protect the confidentiality of your responses, your session will end after 15 minutes of inactivity.

Haga clic aquí para español

© 2022 Regents of the University of Michigan
(v1.2)

The portal page also checks for concurrent sessions of the self-scheduling survey and, if found, will deny the later login attempt from accessing the self-scheduling survey (a wait message is displayed to try again later). The portal will attempt to clear any existing Blaise session using the returned primary key before starting the self-scheduling survey with several parameters passed into the survey (such as the project ID, a web server identifier, mode, and others).

4. Blaise Datamodel

4.1 Alien Procedure Calls

The Blaise data model communicates with the management system and with the scheduling database via alien procedure calls to a Windows service installed on the web server (see the Blaise Fibonacci sample program for an example of using an alien procedure). Implemented calls include:

4.1.1 Scheduling Database Specific

- procSchedDateRange: Start and end date for the study, days of week available, and enabled and disabled dates. This information had originally been used with a date control, but in the current implementation, the dates are displayed in a drop-down list. The dates are still filtered by the

> back to Table of Contents

service configuration settings. A study-specific rule to restrict appointment start dates by three days from the current date has been implemented in the data model.

- procSchedTimeSlots: Retrieves all the time slots available for a selected date. This is populated to the drop-down list of times.

4.1.2 Management System Specific

- procGetApptInfo: Whether this is a prior appointment, along with date, time, time zone, and appointment ID of the appointment.
- procSetNewAppt: Set a new appointment with the management system using start date/time, end date/time, time zone, contact phone, session contact method, alternative contact phone, and appointment ID.
- procRescheduleApt: Reschedule an existing appointment with the management system using start date/time, end date/time, time zone, contact phone, session contact method, alternative contact phone, and appointment ID.
- procCancelAppt: Cancel an existing appointment using appointment ID and session contact method.
- procRecordContact: Write to the management system the type of appointment contact: canceled, scheduled, or rescheduled, with the parameters of start date/time, end date/time, time zone, contact phone, session contact method, alternative contact phone, and appointment ID.
- procGetContactInfo: Get the first and last name, email, email type [confirmation/cancellation], phone, and phone type [confirmation/cancellation].
- procGetLanguage: Get the initial current language to use for the appointment (English or Spanish).
- procSetContactInfo: Write out to the management system the first and last name, email, email type [confirmation/cancellation], phone, and phone type [confirmation/cancellation].

4.1.2 Service Specific

- getDateTime: Converts date and time with time zone into a single string representation of the time.

4.2 Windows Service

A Windows service, SS_Svc.exe, was developed to reside on the web server in the Windows services and to communicate with Blaise via the alien procedure method. A port was "opened" to enable these communications to the Blaise self-scheduling survey. The service then sends/retrieves information from the survey, the management system, and the database endpoint. It transforms raw data from any of the three sources to appropriate formats; sets up secure communications; and handles authentication, logins, and so forth behind the scenes.

In prior releases, the service had communicated directly with the database via stored procedures. An even earlier implementation had no external database with dates and time slots, but just a few settings that were read from the configuration file: start/end dates, start/end shift times, daily schedule, enabled dates,

> back to Table of Contents

disabled dates, and interviewer-assisted flags. Appointment information was still retrieved and stored using the management system.

4.3 Respondent Interface

The Blaise data model serves as the interface for the respondent to enter their appointment. The self-scheduling survey uses just five pages: three are for setting up a new/rescheduled appointment, one is for choosing to cancel/reschedule an appointment, and one is for canceling.

4.3.1 Scheduling an Appointment

If a respondent is scheduling an appointment, they will see the first of three appointment entry screens. The respondent selects the time zone they want, and then the date and time controls are populated with the available dates and times adjusted by the time zone difference between the call center (in EST) and the respondent's time zone. If there is a big time difference between time zones, then either the starting time will be later than the call center beginning shift time or the ending time will likely be in the afternoon. In either situation, the appointments made by the respondent will fall somewhere within the call center hours of operation.

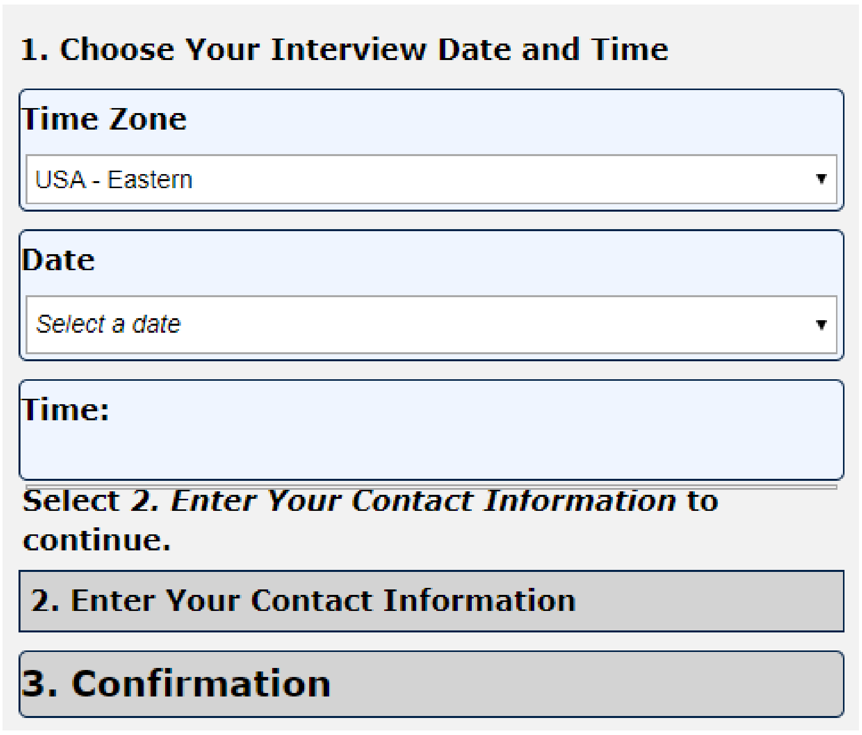
Time zones will adjust the date/time drop-down lists shown to the respondent. In addition, the time zone information collected has to be transformed between standard time zone names and time zones as they are stored in the management system. As a result, a time zone external lookup database was created to handle the name translations, as well as retrieve the appropriate time zone offset the management system expects.

Navigation between pages is accomplished by clicking the section buttons (see Figure 3). These have been made extra wide so they appear as a button bar. Respondents are free to move between the first two screens (*Choose Your Interview Date and Time* and *Enter Your Contact Information*) and make changes.

Once the respondent has selected the time zone, date, and time of their appointment and has navigated to the contact information section, a summary of the appointment date and time is displayed in the first section. In the second section (see Figure 4), the respondent is asked for their contact name (first and last), email, and phone number, and to provide any relevant notes for things like an international phone number or the need to be called at a different number than the one entered in the phone field, etc.

> back to Table of Contents

Figure 3. Section Buttons



1. Choose Your Interview Date and Time

Time Zone

USA - Eastern

Date

Select a date

Time:

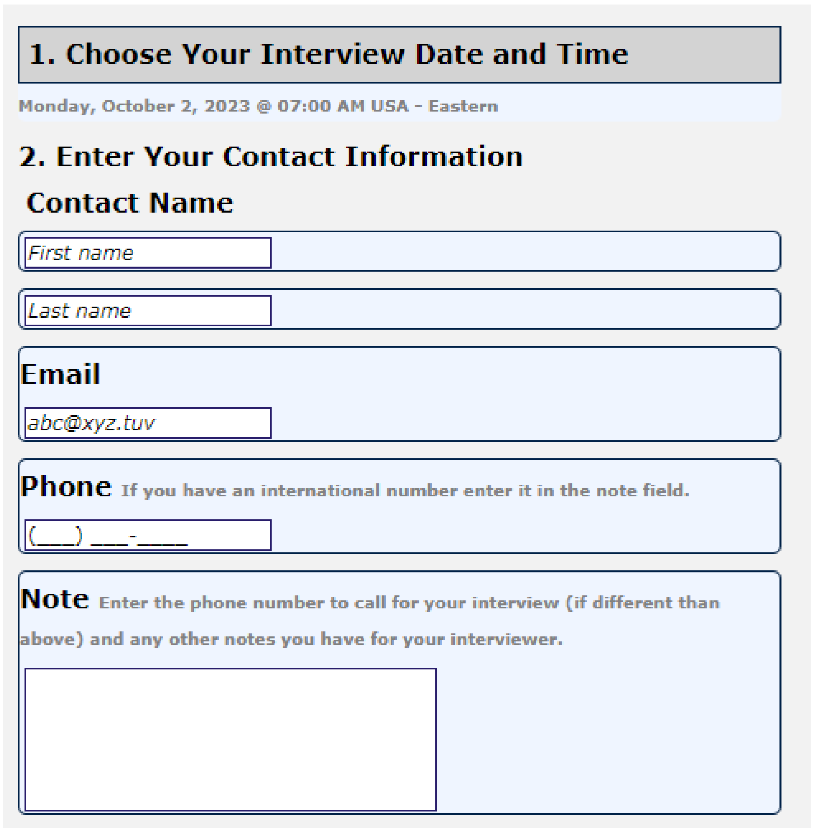
Select 2. *Enter Your Contact Information* to continue.

2. Enter Your Contact Information

3. Confirmation

> back to Table of Contents

Figure 4. Contact Information Screen


1. Choose Your Interview Date and Time

Monday, October 2, 2023 @ 07:00 AM USA - Eastern

2. Enter Your Contact Information

Contact Name

First name

Last name

Email

abc@xyz.tuv

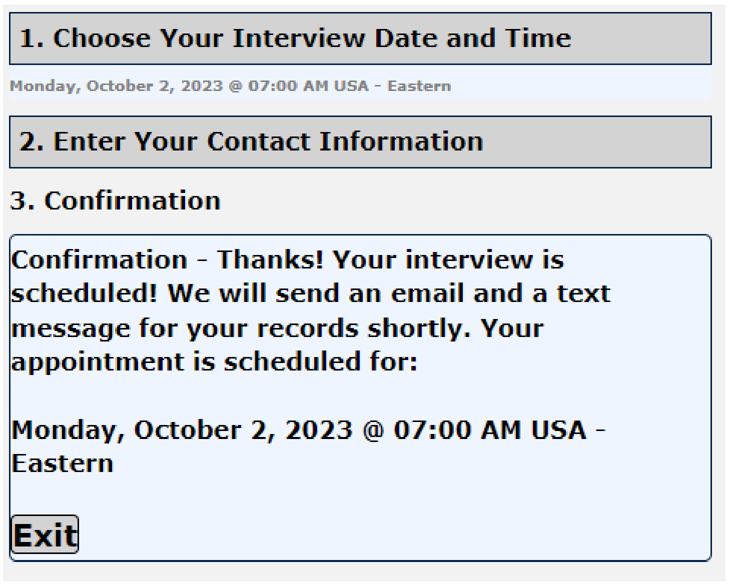
Phone If you have an international number enter it in the note field.

(___) ___-____

Note Enter the phone number to call for your interview (if different than above) and any other notes you have for your interviewer.

Once the respondent navigates to the confirmation screen, the system checks to make sure there is still an appointment slot available. If there is, the respondent is presented with the confirmation screen (see Figure 5). If there are no slots available, the respondent is asked to select a different date or time.

Figure 5. Confirmation Screen


1. Choose Your Interview Date and Time

Monday, October 2, 2023 @ 07:00 AM USA - Eastern

2. Enter Your Contact Information

3. Confirmation

Confirmation - Thanks! Your interview is scheduled! We will send an email and a text message for your records shortly. Your appointment is scheduled for:

Monday, October 2, 2023 @ 07:00 AM USA - Eastern

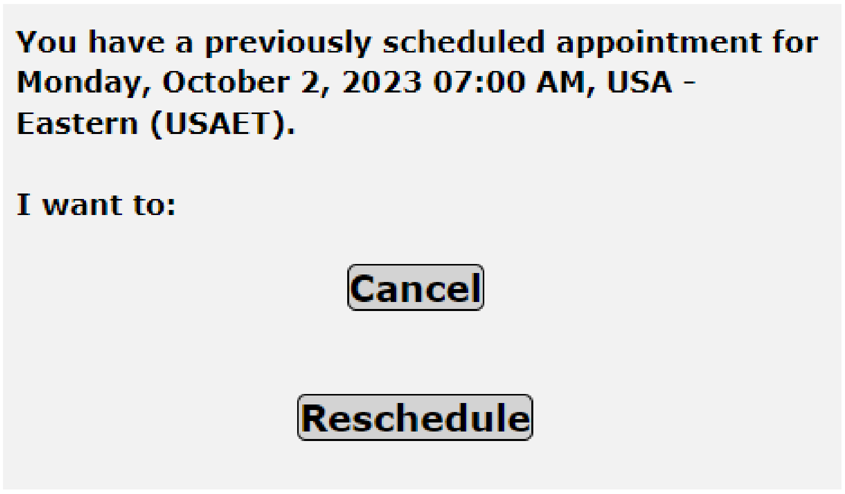
> back to Table of Contents

Successful appointments decrement the number of slots available for a date/time, and once that count is less than one, it will no longer be available in this interface. Each time the respondent changes time zone, date, or time on this screen, the service is contacted again to refresh the values in an effort to avoid overbooking appointments.

4.3.2 Rescheduling an Appointment

When a respondent already has a scheduled appointment, they have an option to reschedule (or cancel) that appointment date/time (see Figure 6). When an appointment is being rescheduled, the prior appointment is automatically canceled, and the self-scheduling survey collects the new appointment information. The information is collected in exactly the same way as a new appointment.

Figure 6. Reschedule/Cancel Screen


You have a previously scheduled appointment for Monday, October 2, 2023 07:00 AM, USA - Eastern (USAET).

I want to:

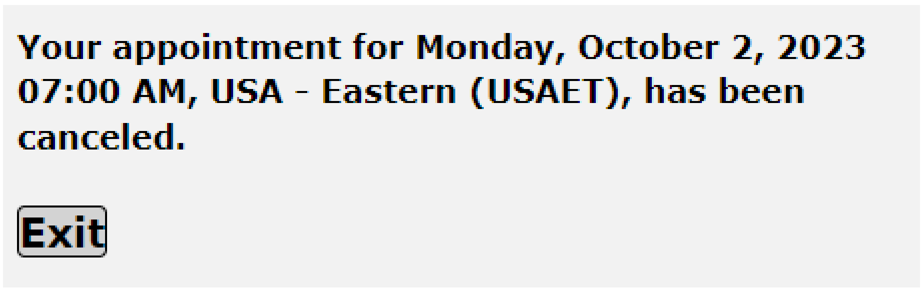
For rescheduled appointments, the self-scheduling survey will create a new appointment ID (the letter "A" + primary key) and launch the self-scheduling survey again. The management system needs to know the difference between a newly scheduled appointment and one that has been rescheduled, so this mechanism was chosen to pass that information, and then start with a fresh survey to gather the new appointment.

4.3.3 Canceling an Appointment

A simple verification screen is displayed to show the appointment has been canceled (see Figure 7). In this case, the appointment is marked as canceled, and the number of available slots for this date/time is incremented in the database.

Figure 7. Reschedule/Cancel Screen

> back to Table of Contents


Your appointment for Monday, October 2, 2023 07:00 AM, USA - Eastern (USAET), has been canceled.

5. Future Work

The logon portal needs to be changed to use .NET's resource files, .resx, for language-specific texts. Currently, the portal switches between two logon pages. This means there is code duplication, a small delay as the page loads, and some extra programming required to transfer parameters between the pages. By using .resx files, there would be one source page with no code duplication, no delay between language switches, and easier maintenance of texts.

The endpoints in the Windows service have been kept as work progressed and are currently backward compatible with all prior versions of the self-scheduling survey but will be removed to keep the code clean.

> back to Table of Contents

Large Scale Lookups, from an End-User and a Programmer Perspective

Peter Stegehuis and Naxin Zheng, Westat

Presenter: Peter Stegehuis

1. Introduction

With many lookups in a CAPI instrument, the contents of the lookup database are typically the same for every survey respondent. When we ask respondents what prescribed medicines they use, for example, the list we use in the background for their search is the same for all respondents. However, when we want them to select a physician or medical professional in a lookup search, it doesn't make sense to present all possible selections from the entire country to everyone, as the lookup would be too large and, even more importantly, it would end up showing many irrelevant results to the end user, making it too hard to find the desired result. This paper discusses ways to make medical provider lookups relevant from an end-user perspective and manageable from a programming and file management viewpoint.

2. The Challenge

We are asking respondents for medical information, including doctor and hospital visits, over the period since the previous interview in the panel survey. Even with the help of records at hand, it can be difficult for respondents to recall this information for all household members, especially if the reference period is, for instance, six months long. There is also a separate follow-up survey with the providers, to get more accurate data on the exact procedures and cost. In order for providers to be able disclose this data, we ask household members to sign so-called forms to authorize data collection from providers. It is therefore crucial to collect accurate data on the providers and their contact information.

3. What is Needed?

So, rather than letting the interviewer just type provider names, addresses, and phone numbers as remembered by the respondent, we want to have them select a record from a lookup that has accurate medical provider information.

In order to implement this within the interview program, we need to have the following elements in place:

- A good resource for provider records
- A mechanism for a good lookup
- Meaningful, localized lookup data files
- A user-friendly presentation of lookup search results

We will describe these elements in the following sections.

4. Resource for Provider Data Records

There are different data sets of medical providers in the entire US available, some publicly and some for an annual fee. We have looked at a number of them and found that the most comprehensive data set seems to be the National Provider Index (NPI).

There is a strong incentive for medical providers to sign up and get a unique NPI ID: any provider—person or facility—who wants to ever bill a federal entity like Medicare or Medicaid needs to have an

> back to Table of Contents

NPI ID. This makes the NPI a more comprehensive list of providers than others, a crucial distinction for our purpose.

It is far from a perfect list however: there is no incentive for providers to delist when a doctor retires or a medical facility shuts down or moves. This means that the list keeps growing year over year, and many entries are no longer relevant. It also means that the lookup search mechanism becomes of even greater importance if the interviewer is to find the correct medical provider in an enormous mix of useful and useless records.

5. The Search Mechanism

To be actually useful during an already long interview with possibly many provider searches, any search mechanism to be used for this search has to be fast and powerful, serving up search results almost instantaneously after an interviewer types in a search string. It also needs to be somewhat forgiving of typos in name or address of a provider as given by the respondent and allow for easy searches on name, phone number, and/or address.

Before we switched to Blaise 4.8, we did this search in SQL Server, using a “LIKE” search with wildcards. The speed of a search was adequate, but that mechanism was not able to deal with typos very well, especially if they were at the start of search words. It also needed separate searches for name, address, and phone number.

The switch to Blaise 4.8 and trigram lookups meant a big improvement in the success rate of provider searches. Because of the trigrams—three letter snippets—this search mechanism is much less influenced by a typo, and the way Blaise indexes those trigrams means the search results come up extremely fast. With a carefully constructed ‘search string’ for each record in the lookup data file, we can also assure that the interviewer can choose to enter parts of the provider name, address, phone number, or some combination of these without having to specify that in any way by designating a search category beforehand. We have seen much better outcomes since we switched, with a much higher percentage of searches finding a match, as opposed to adding verbatim provider info after an unsuccessful search.

5.1 Creating Localized Lookup Data Files

One way to try to include relevant providers for a specific interview is to use the respondent’s state: include all providers within that state and exclude all others. The problem with that approach is that people will cross state lines to get medical services, and for many people near state lines, that may well be their preferred option. The same disadvantage goes for the counties within states.

Instead, we wanted to ensure that the respondent’s address was more in the center of the area that would be covered by the lookup file. So, we started off with the respondent’s zip code (for non-US readers, that’s a postal code, much smaller than states or counties). Just including providers within that zip code would be too limiting, so we wanted to include all the providers within a wider circle around the respondent’s zip code.

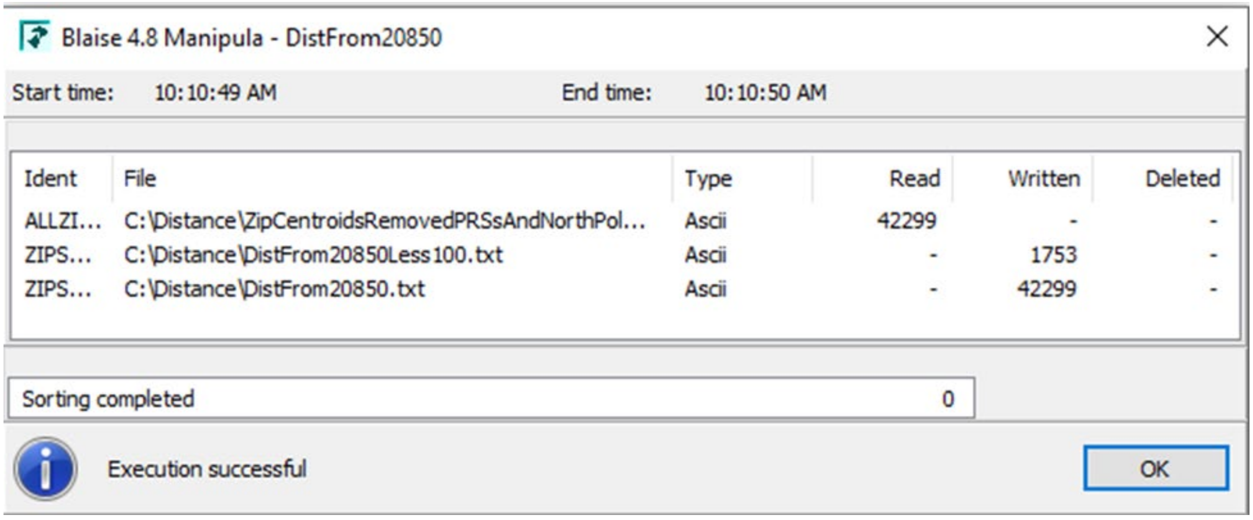
At the outset, we created circles with a radius of 100 miles around the zip code’s centroid (you can think of that as the center or average location in the zip code area). The list of zip codes in the US, and their centroid coordinates, is freely available. So, for each zip code, we calculated a list of zip codes that would fit within the circle with radius of 100 miles by calculating the distance from one centroid to another.

As an aside, Manipula is known for being very fast, especially in handling large text files. We had to double check the results when calculating the distance from zip code 20850 to all others—writing two

> back to Table of Contents

output files and sorting the first output file took all of one second. Figure 1 is a screenshot with that result, with start time and end time near the top.

Figure 1. 42,000 Distance Calculations in One Second



```
Blaise 4.8 Manipula - DistFrom20850

Start time:     10:10:49 AM                End time:     10:10:50 AM

Ident      File                                          Type        Read    Written   Deleted
ALLZI...   C:\Distance\ZipCentroidsRemovedPRSsAndNorthPol...   Ascii      42299        -         -
ZIPS...    C:\Distance\DistFrom20850Less100.txt          Ascii          -       1753         -
ZIPS...    C:\Distance\DistFrom20850.txt                 Ascii          -      42299         -

Sorting completed                                              0

Execution successful                                                        OK
```

Unfortunately for us, creating Blaise trigram lookup files takes a lot longer—not surprising when thinking about the work needed for creating, indexing, and managing the trigram index file.

The next issue was when and where to create these lookup files, and how to get them on interviewer laptops. There are more than 42,000 zip codes in the US and creating and storing them all takes up 20 TB (or 2 TB zipped), which is, of course, way too much data for storing on an interviewer's laptop. We can't create them during the interview either, as it can take up to 15 or 20 minutes—and lots of resources—to create some of the bigger lookup files. So, we are creating them on the interviewer laptops: right after the interviewer has picked up the transmission with the preload data for that case, the IMS checks whether lookup files for zip code and state are already on the laptop. If not, they get created at that point.

We tried to optimize the Manipula process that creates the lookups to minimize the time needed when interviewers get a big number of cases assigned at once. Somewhat to our surprise, we found that splitting the process in two separate Manipula setups was faster than using just one setup.

The annually growing file with NPI records is too big—close to seven million records at the moment—to fit into memory in a Manipula setup, where that would have been useful. So, using a two-step process, each one using a differently sorted file, we ended up with a combined process that was much faster than the one-step approach. This was much appreciated by field staff and home office staff who sometimes had to wait for the completion of this task.

We have also reduced the size of some of the biggest lookup files by basing the diameter of the circle around the zip code centroid on the urbanicity degree of the area (this is also freely available data). The thinking there is that there are many more medical providers in urban areas compared to rural areas, so respondents would typically not travel as far to see a provider. This means that instead of a 100-mile diameter, we can have a much smaller diameter in, say, New York City compared to a rural area in, for instance, Wyoming.

On the other hand, the process gets slightly more complicated by choosing to include certain providers in all lookup files, regardless of their zip code. These are what we call 'Centers of Excellence,' with

> back to Table of Contents

examples like the Mayo Clinic, Sloan Kettering, Johns Hopkins, and other well-known medical facilities. People may travel to these centers for consultation and/or treatment, including surgery. While these cases may be relatively rare, they may include extensive and expensive medical treatment. Including these facilities in every lookup file helps us to capture everything about these visits as accurately as possible.

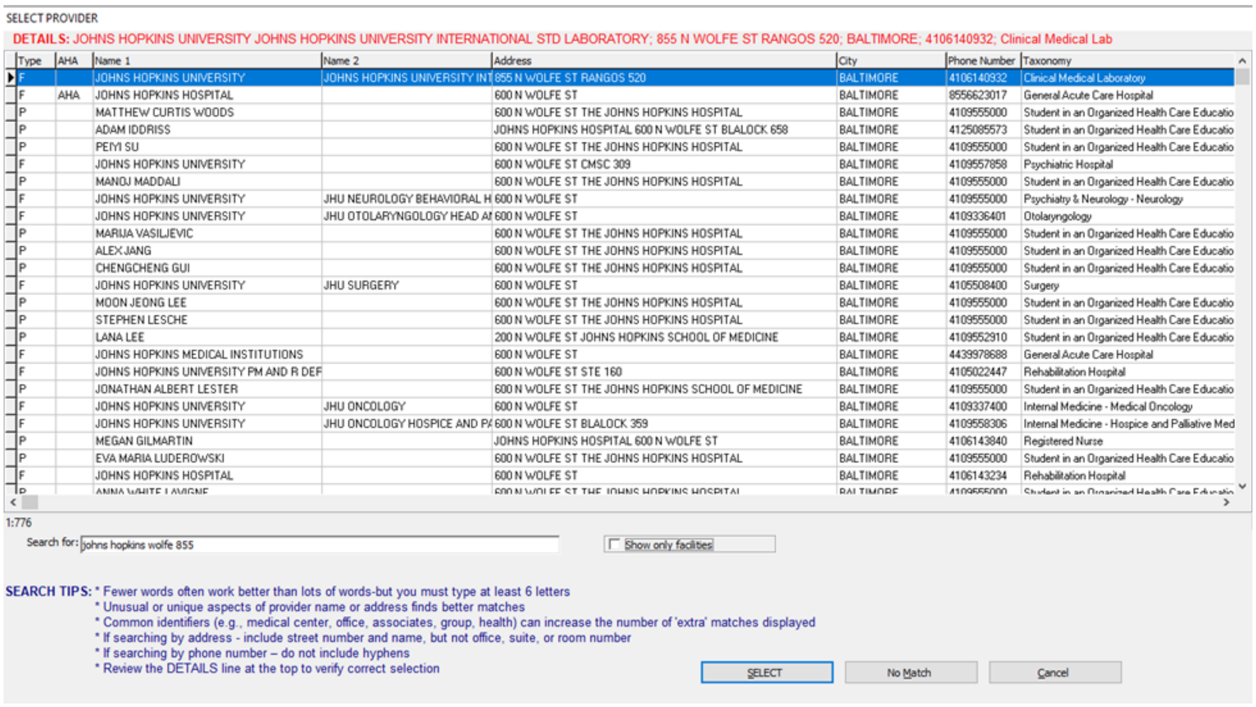
6. Presentation

Lookups, including trigram lookups, have been available in Blaise for decades, so Blaise users are likely to be very familiar with the look and feel of these screens. There are, however, some elements that can be tailored or added to make a lookup more user-friendly and more effective, as well.

We are using a Manipula dialog for the display of the lookup, which allows us to add a few elements that are useful to interviewers.

Figure 2 is a screenshot of a typical search.

Figure 2. Medical Provider Lookup Screen



SELECT PROVIDER

DETAILS: JOHNS HOPKINS UNIVERSITY JOHNS HOPKINS UNIVERSITY INTERNATIONAL STD LABORATORY; 855 N WOLFE ST RANGOS 520; BALTIMORE; 4106140932; Clinical Medical Lab

Type	AHA	Name 1	Name 2	Address	City	Phone Number	Taxonomy
F		JOHNS HOPKINS UNIVERSITY	JOHNS HOPKINS UNIVERSITY INT	855 N WOLFE ST RANGOS 520	BALTIMORE	4106140932	Clinical Medical Laboratory
F	AHA	JOHNS HOPKINS HOSPITAL		600 N WOLFE ST	BALTIMORE	8556623017	General Acute Care Hospital
P		MATTHEW CURTIS WOODS		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
P		ADAM IDDRISS		JOHNS HOPKINS HOSPITAL 600 N WOLFE ST BLALOCK 658	BALTIMORE	4125085573	Student in an Organized Health Care Educatio
P		PEIYI SU		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
F		JOHNS HOPKINS UNIVERSITY		600 N WOLFE ST CMSC 309	BALTIMORE	4109557858	Psychiatric Hospital
P		MANOJ MADDALI		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
F		JOHNS HOPKINS UNIVERSITY	JHU NEUROLOGY BEHAVIORAL H	600 N WOLFE ST	BALTIMORE	4109555000	Psychiatry & Neurology - Neurology
F		JOHNS HOPKINS UNIVERSITY	JHU OTOLARYNGOLOGY HEAD AI	600 N WOLFE ST	BALTIMORE	4109336401	Otolaryngology
P		MARIJA VASILJEVIC		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
P		ALEX JANG		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
P		CHENGCHENG GUI		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
F		JOHNS HOPKINS UNIVERSITY	JHU SURGERY	600 N WOLFE ST	BALTIMORE	4105508400	Surgery
P		MOON JEONG LEE		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
P		STEPHEN LESCHE		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
P		LANA LEE		200 N WOLFE ST JOHNS HOPKINS SCHOOL OF MEDICINE	BALTIMORE	4109552910	Student in an Organized Health Care Educatio
F		JOHNS HOPKINS MEDICAL INSTITUTIONS		600 N WOLFE ST	BALTIMORE	4439978688	General Acute Care Hospital
F		JOHNS HOPKINS UNIVERSITY PM AND R DEF		600 N WOLFE ST STE 160	BALTIMORE	4105022447	Rehabilitation Hospital
P		JONATHAN ALBERT LESTER		600 N WOLFE ST THE JOHNS HOPKINS SCHOOL OF MEDICINE	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
F		JOHNS HOPKINS UNIVERSITY	JHU ONCOLOGY	600 N WOLFE ST	BALTIMORE	4109337400	Internal Medicine - Medical Oncology
F		JOHNS HOPKINS UNIVERSITY	JHU ONCOLOGY HOSPICE AND P/	600 N WOLFE ST BLALOCK 359	BALTIMORE	4109558306	Internal Medicine - Hospice and Palliative Med
P		MEGAN GILMARTIN		JOHNS HOPKINS HOSPITAL 600 N WOLFE ST	BALTIMORE	4106143840	Registered Nurse
P		EVA MARIA LUDEROWSKI		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio
F		JOHNS HOPKINS HOSPITAL		600 N WOLFE ST	BALTIMORE	4106143234	Rehabilitation Hospital
P		ANNA WHITE LAVIGNE		600 N WOLFE ST THE JOHNS HOPKINS HOSPITAL	BALTIMORE	4109555000	Student in an Organized Health Care Educatio

1:776

Search for: johns hopkins wolfe 855

☐ Show only facilities

SEARCH TIPS: * Fewer words often work better than lots of words-but you must type at least 6 letters
* Unusual or unique aspects of provider name or address finds better matches
* Common identifiers (e.g., medical center, office, associates, group, health) can increase the number of 'extra' matches displayed
* If searching by address - include street number and name, but not office, suite, or room number
* If searching by phone number – do not include hyphens
* Review the DETAILS line at the top to verify correct selection

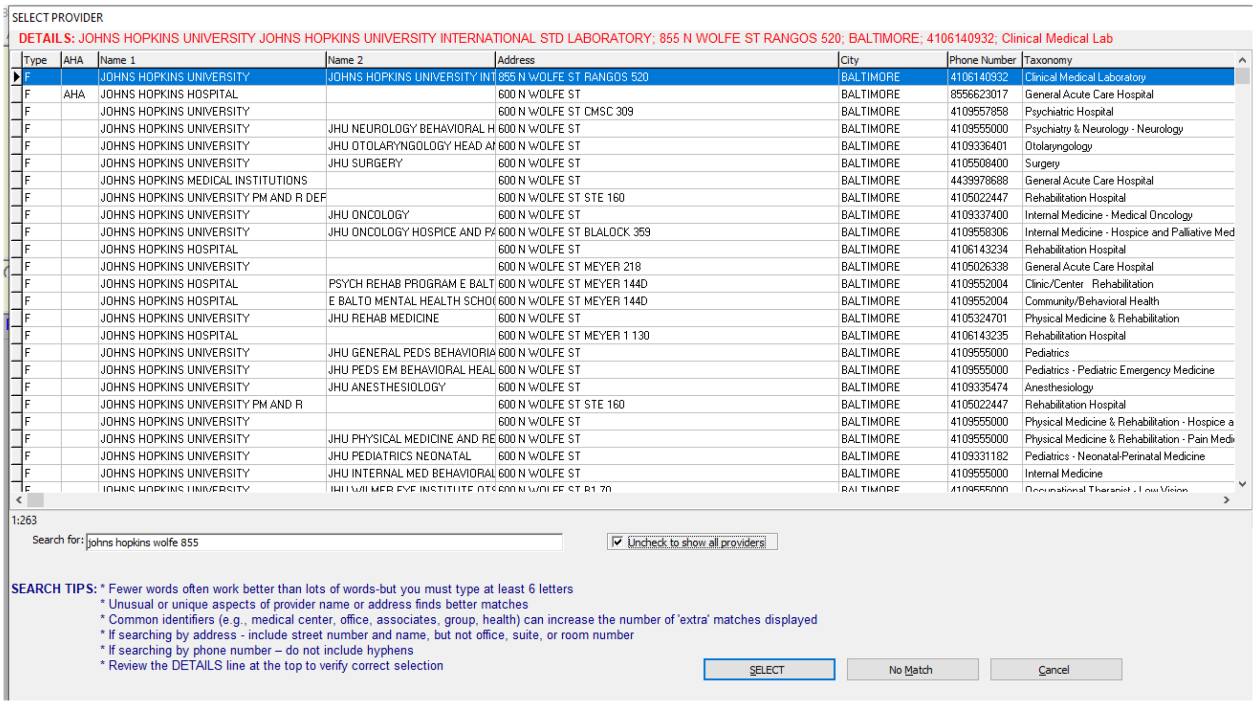
Note the red line at the top, which shows the detailed information of the currently selected line in the results grid. This ‘detail line’ gets updated automatically whenever a different line is selected and it allows the interviewer to easily see all the available info on that provider, even parts that might be cut off in the results grid.

There are search tips for interviewers near the bottom because it can really be a difficult task to help a respondent recall details of an event and names of doctors and facilities, even more so if the event happened several months ago.

> back to Table of Contents

We also included an option to filter the results and display only facilities, instead of doctors and facilities all mixed together. As you can see in Figure 3, we have a checkbox with the label “Show only facilities” next to the searchstring input line. When checked, we get the following:

Figure 3. Medical Provider Lookup Screen with a Filter to Show Only Medical Facilities



SELECT PROVIDER

DETAILS: JOHNS HOPKINS UNIVERSITY JOHNS HOPKINS UNIVERSITY INTERNATIONAL STD LABORATORY; 855 N WOLFE ST RANGOS 520; BALTIMORE; 4106140932; Clinical Medical Lab

Type	AHA	Name 1	Name 2	Address	City	Phone Number	Taxonomy
F		JOHNS HOPKINS UNIVERSITY	JOHNS HOPKINS UNIVERSITY INT	855 N WOLFE ST RANGOS 520	BALTIMORE	4106140932	Clinical Medical Laboratory
F	AHA	JOHNS HOPKINS HOSPITAL		600 N WOLFE ST	BALTIMORE	8556623017	General Acute Care Hospital
F		JOHNS HOPKINS UNIVERSITY		600 N WOLFE ST CMSC 309	BALTIMORE	4109557858	Psychiatric Hospital
F		JOHNS HOPKINS UNIVERSITY	JHU NEUROLOGY BEHAVIORAL H	600 N WOLFE ST	BALTIMORE	4109555000	Psychiatry & Neurology - Neurology
F		JOHNS HOPKINS UNIVERSITY	JHU OTOLARYNGOLOGY HEAD AI	600 N WOLFE ST	BALTIMORE	4109336401	Otolaryngology
F		JOHNS HOPKINS UNIVERSITY	JHU SURGERY	600 N WOLFE ST	BALTIMORE	4105508400	Surgery
F		JOHNS HOPKINS MEDICAL INSTITUTIONS		600 N WOLFE ST	BALTIMORE	4439978688	General Acute Care Hospital
F		JOHNS HOPKINS UNIVERSITY PM AND R DEF		600 N WOLFE ST STE 160	BALTIMORE	4105022447	Rehabilitation Hospital
F		JOHNS HOPKINS UNIVERSITY	JHU ONCOLOGY	600 N WOLFE ST	BALTIMORE	4109337400	Internal Medicine - Medical Oncology
F		JOHNS HOPKINS UNIVERSITY	JHU ONCOLOGY HOSPICE AND PA	600 N WOLFE ST BLALOCK 359	BALTIMORE	4109558306	Internal Medicine - Hospice and Palliative Med
F		JOHNS HOPKINS HOSPITAL		600 N WOLFE ST	BALTIMORE	4106143234	Rehabilitation Hospital
F		JOHNS HOPKINS UNIVERSITY		600 N WOLFE ST MEYER 218	BALTIMORE	4105026338	General Acute Care Hospital
F		JOHNS HOPKINS HOSPITAL	PSYCH REHAB PROGRAM E BALT	600 N WOLFE ST MEYER 144D	BALTIMORE	4109552004	Clinic/Center Rehabilitation
F		JOHNS HOPKINS HOSPITAL	E BALTO MENTAL HEALTH SCHO(	600 N WOLFE ST MEYER 144D	BALTIMORE	4109552004	Community/Behavioral Health
F		JOHNS HOPKINS UNIVERSITY	JHU REHAB MEDICINE	600 N WOLFE ST	BALTIMORE	4105324701	Physical Medicine & Rehabilitation
F		JOHNS HOPKINS HOSPITAL		600 N WOLFE ST MEYER 1 130	BALTIMORE	4106143235	Rehabilitation Hospital
F		JOHNS HOPKINS UNIVERSITY	JHU GENERAL PEDS BEHAVIORIA	600 N WOLFE ST	BALTIMORE	4109555000	Pediatrics
F		JOHNS HOPKINS UNIVERSITY	JHU PEDS EM BEHAVIORAL HEAL	600 N WOLFE ST	BALTIMORE	4109555000	Pediatrics - Pediatric Emergency Medicine
F		JOHNS HOPKINS UNIVERSITY	JHU ANESTHESIOLOGY	600 N WOLFE ST	BALTIMORE	4109335474	Anesthesiology
F		JOHNS HOPKINS UNIVERSITY PM AND R		600 N WOLFE ST STE 160	BALTIMORE	4105022447	Rehabilitation Hospital
F		JOHNS HOPKINS UNIVERSITY		600 N WOLFE ST	BALTIMORE	4109555000	Physical Medicine & Rehabilitation - Hospice a
F		JOHNS HOPKINS UNIVERSITY	JHU PHYSICAL MEDICINE AND RE	600 N WOLFE ST	BALTIMORE	4109555000	Physical Medicine & Rehabilitation - Pain Medi
F		JOHNS HOPKINS UNIVERSITY	JHU PEDIATRICS NEONATAL	600 N WOLFE ST	BALTIMORE	4109331182	Pediatrics - Neonatal-Perinatal Medicine
F		JOHNS HOPKINS UNIVERSITY	JHU INTERNAL MED BEHAVIORAL	600 N WOLFE ST	BALTIMORE	4109555000	Internal Medicine

1:263

Search for: johns hopkins wolfe 855

[✓ Uncheck to show all providers]

SEARCH TIPS: * Fewer words often work better than lots of words-but you must type at least 6 letters
* Unusual or unique aspects of provider name or address finds better matches
* Common identifiers (e.g., medical center, office, associates, group, health) can increase the number of 'extra' matches displayed
* If searching by address - include street number and name, but not office, suite, or room number
* If searching by phone number – do not include hyphens
* Review the DETAILS line at the top to verify correct selection

Note that all records now are facilities and that the label for the checkbox now reads “Uncheck to show all providers.”

This capability to filter search results is quite naturally built-in in Blaise 5, but in Blaise 4.8, it is a fairly well-hidden feature. Using it requires a decent amount of work, and if you want to do your own sorting of search results, it also requires a recalculation of trigram scores, as those are not passed on with the search results.

The searchstring that the interviewer types to try to select a provider can be accessed and stored for later analysis. When no match can be found in the lookup, the interviewer can select the “No Match” button and enter provider details verbatim.

7. Conclusions

Trigram lookups in Blaise work very well and fast, even with very large lookup files. This is as true in Blaise 5 as well as it was in Blaise 4. In Blaise 5, the additional flexibility for presentation and the ease of filtering search results are much appreciated.

> back to Table of Contents

Session Data Preservation and Migration—Problems and Solutions

Jason Ostergren and Helena Stolyarova, University of Michigan Institute for Social Research

The Health and Retirement Study (HRS) at the University of Michigan Institute for Social Research is a longitudinal study that originated in 1992, switched to conducting interviews using Blaise 4 in 2002, and moved to Blaise 5 in 2018. Among the challenges HRS has faced since the switch to Blaise 5 is how to handle Blaise 5 session data when an interview is not completed in one sitting. Session data is the working database that maintains the state of the instrument, including the values of temporary data and properties of fields. If that data is lost in a case where an interview was interrupted and has to be resumed later, significant problems can result if the instrument is dependent on session data to resume correctly, as is the case with HRS. There are a number of aspects to this problem, and HRS has tackled new and different ones in each of the three (2018, 2020, and 2022) waves of interviews since adopting Blaise 5. Data migration has been at the center of some of these issues—for example, HRS has had to incorporate careful attention to harmless changes into processes for updating instruments in the field. Most problems have been solved, sometimes with significant help from CBS, but with varying degrees of completeness—for example, it was thought that data migration issues had been solved in 2020, but it turned out that mode switches rendered that solution incomplete. Finally, HRS has determined that it would be useful to retain and preserve session data after interview completion, which has been partially solved as well. This paper will break down why session data has been important to HRS, as well as the various issues and solutions to the problems that have arisen—including testing tools—and what remains to be done.

1. Session Data

A brief discussion of the nature and purpose of session data is in order. Session data is the stored state information from the Blaise 5 session service, which handles active interview sessions. The runtime session database (shortened to "session database" here), where the session data is stored, is regularly updated with data from the active interview while in session. Additionally, the session database persists session data when an interview is interrupted. When an interrupted interview is resumed, provided there is a primary key, Blaise first checks to determine whether session data are available. If session data for the key are found in the session database, the interview is resumed where it left off, with all session data present, including auxiliary data and survey state. This process is at the heart of the issues discussed in this paper. There are a variety of ways that session data might be lost before an interview is resumed, and in a survey like HRS, this causes problems for any interview that is interrupted. Generally speaking, HRS needs session data to be preserved both between sessions and across new datamodel releases (within an HRS "wave" [e.g., HRS 2020]). There are a number of reasons that this is the case.

At first glance, session data may not appear to be a crucial component of the Blaise system for a user to understand. Indeed, in many use cases, there would be no need for a Blaise user to consider it at all. Session data-related issues arise mainly due to the ways external processes interact with Blaise interviews. A simple web survey that is deployed and runs without interruption or intervention until the end of its interview period would be unlikely to be impacted by these issues. The failure points that HRS has experienced occur in transitional processes. One such process happens when particular interviews are stored and transferred within our system—for example, between servers in the building and field laptops. Another such process happens when new versions of the HRS datamodel must be deployed, replacing old versions (rather than multiple versions running side by side).

> back to Table of Contents

2. HRS and Session Data

The other element that makes session data a concern to HRS is the complexity of the HRS survey instrument. HRS has design features, legacy code, and operational requirements that turn out to make session data important in a number of ways. A very simple instrument subject to the same transitional processes alluded to earlier may have some problems (e.g., resetting the position to the first question), but those would be among the more manageable ones HRS encounters. The risk was not immediately apparent to HRS. In fact, HRS has come across new problems connected with session data in each of the three "waves" since adopting Blaise 5.

Among the design features and legacy code found in HRS that make session data significant are the following.

HRS has reusable question series for estimation (referred to as "unfolding sequences," or just "unfoldings" hereafter) that are scattered throughout the instrument after questions where a respondent may be unsure about an important amount, such as the value of a house or pension. These are programmed as procedures for legacy reasons and, as such, are made up of temporary data—even the questions defined as Blaise Fields are actually treated as auxfields. Historically, it was possible to close a routing gate after each procedure was complete and preserve the final result. If an interviewer needed to get back in, there was a complicated method of doing so involving erasing answers and moving back and forth. In a theme that will repeat in this paper, the need to support web self-interviews required a change.

HRS also ran into a variety of issues connected with losing session data that were not HRS specific. For example, when resuming a case that had lost session data, the interview would revert to the first field on the route rather than the field at which the interview was suspended. This forced the interviewer—or worse, a self-interview respondent—to step through the dozens or hundreds of questions that had previously been answered. Another such problem was that formerly suppressed signals would be reset and require intervention again.

HRS also ran into some still-mysterious data-loss issues that likely result from complicated programming for arrayed data (in this case, data about children-in-law) reacting badly to the loss of session data upon resume. This highlights the need to consider session data in testing. HRS has a variety of custom tools for testing its instrument and dedicated staff to handle the testing. An enormous amount of time is spent testing each instrument to minimize flaws. However, until HRS discovered the kinds of session data problems described in this paper, there was no provision in those tools to simulate loss of session data; therefore, problems resulting from that eventuality (like the missing children-in-law) were not discovered before the field period began.

One last wrinkle is that there was initially no effort to preserve any session data after the interview was over. When problems were discovered later on, there was a desire to look at session data to troubleshoot and reconstruct missing data, much like is often done with audit trail data. Paradoxically, of course, even if the session data were preserved in general, it would have been partially missing from the problem cases. Nonetheless, while preventing loss of session data during suspend and resume was the highest priority, HRS realized that once that was solved, it would be desirable to retain session data even after the interview was complete for later reference.

3. HRS Session Data Vulnerabilities

To expand on the reasons mentioned earlier for the HRS instrument being vulnerable to loss of session data mid-interview, a little background is required. The design and programming of the HRS instrument,

> back to Table of Contents

which goes through a process of reevaluation and improvements in each two-year cycle, attempts to balance out a huge amount of content with extensive customization of flow, question text substitutions, and other tricks. The interview typically clocks in at more than two hours—sometimes much more—so a lot of attention is paid to complicated logic to minimize the number of questions and to an equally complicated amount of logic to make question text personalized for each respondent as much as possible. When the decision was made to add a web self-interview mode to the HRS instrument starting in 2018, a comprehensive reevaluation of all this logic was undertaken. For example, it was quickly determined that the self-interview needed to support respondents if they simply clicked next rather than answering a question. In other words, our interviewer-administered mode rarely allows empty answers—the interviewer may probe when instructed or assign DK/RF or other codes as needed. HRS determined that none of these options should be forced on self-interview respondents ("self Rs" from now on).

Furthermore, where previously HRS made extensive use of gates (using conditions that stored data from sequences and removed them from the flow) to close off precisely the kinds of sequences that are now causing problems in session data, now HRS strives to minimize such devices to prevent confusion among self Rs. The reason for this was mainly that having a "previous" button to allow the self R to have control over the ability to return to a previous question was important, but gates that removed previous questions from the flow would make the self R unable to locate their previous answers. Sometimes this could be handled with instructions to the self R, but it was determined that these should not be common occurrences anymore.

There were multiple aspects to addressing this issue. For example, HRS gates normally rely on the interviewer answering a gate "question." This question would have some instruction to the interviewer, even if there was no text to be read to the respondent. The interviewer would then know that once they answered the question, the previous sequence would be inaccessible (at least without some complicated process). Since HRS determined not to force any answers on self Rs and Blaise logic requires some catalyst to properly close a gate, HRS had to remove or replace gates with other mechanisms. Where HRS simply removed gates entirely, as in the case of the aforementioned unfoldings, the loss of session data would cause the code to be reevaluated and the final answers would be emptied out as a result because the session data they relied on was emptied out.

When HRS retained existing gates, it turned to other mechanisms that proved vulnerable to this problem, as well. One such mechanism is the use of the isVisited property to trigger a gate in place of a human-selected answer when, as the name implies, the field is visited (appears on the screen). Since some gates were still required, this was used in a handful of places, including some that turned out to require very complicated logic and even custom browser DEP code. The use of isVisited presented its own problems initially. HRS quickly discovered that it was losing this property data alongside the session data in 2018. We believe that in Blaise versions released after that time, it became possible to persist properties like isVisited by redeclaring them. However, HRS has not explicitly tested this in our process, and we are still under the impression that isVisited is precarious without preserving the session data. The point of all this is that if session data is lost and isVisited is lost with it, gates are unintentionally opened, and the previously gated logic is then reevaluated incorrectly because the gated logic also relied on now missing session data.

4. HRS 2018

With a lot of troubleshooting and experimentation, and with considerable help from CBS, HRS has made progress in chipping away at the kinds of problems outlined above. In 2018, HRS experienced widespread problems with loss of session data. In the case management system used for interviewer-administered cases, session data was being lost on every occasion when an interview was suspended and resumed. This

> back to Table of Contents

resulted in problems like data loss in unfoldings, as described above. This not only caused a loss of data, but also caused respondents to be reasked many questions they had answered before—or alternatively, it caused an employee to have to painstakingly reenter the previous answers from the audit trail, for example.

This problem turned out to be one of the easiest to solve and simply stemmed from inexperience with session data the first time out. In the case management system that was being used, the case files were always packaged and stored after a suspend and the working folder being cleared. When a case was resumed, those files were brought back to their previous locations. However, because there was not a general awareness of the importance of session data, those files were being deleted in this process. Once we realized this, it was a simple matter to retain the session data files, and this problem was solved.

Along these lines, one key takeaway from this paper is that session data should be considered a normal and necessary component of Blaise 5 operation. At the very least, care should be taken with whether it is being retained up until the point when an interview is truly completed.

5. Harmless Changes

Another concept that needs attention is that of harmless (or harmful) changes. This notion concerns data file compatibility (in particular, for the purposes of this paper, session data file compatibility) in a case where a survey needs to be replaced with an updated version of itself. While changes to a survey that break data file compatibility may be common during the development of a survey, once real data collection starts, serious problems can arise if compatibility is broken. Blaise uses a checksum to determine compatibility and may refuse to launch or install a survey if the checksum fails. To make things slightly more complicated, some changes that cause checksums not to match may still be harmless. For example, changing the number of fields in a survey breaks data file compatibility and alters the checksum, but the change can still be considered harmless if the change was due to fields being added rather than deleted. The Blaise 5 online help specifies a set of general rules governing this situation, as follows:

- No (relevant) items may be removed.
- Items can be added.
- Each item of the new collection must be a harmless extension of the related item (i.e., with the same name) of the old collection.

The online help also notes: “As a rule of thumb, extension of the data definition, such as an additional field, enlarging a string, or expanding answer categories, are fairly harmless.”

In practice, if one needed to harmlessly rename a field, for example, it would be necessary to retain the original field (while taking it off the route, probably with a keep statement) and add a new field with the changed name. Simply renaming the field would be a harmful change because it effectively removed that field from the instrument. In principle, there are ways to make almost any proposed change harmless with enough attention to detail.

6. HRS 2020

HRS was forced to begin thinking seriously about how to handle changes between versions during its second (2020) fielding of a Blaise 5 instrument. One of the (perhaps peculiar) features of HRS is that during the field period, which often lasts most of a year or even more than a year, HRS updates its instrument with fixes for problems identified in the field and with high-value changes requested by

> back to Table of Contents

investigators. As a result, HRS has a history of releasing multiple versions of its datamodel each wave (an average of one per month over a year is not surprising). Importantly, these are very similar datamodels, usually with logic or wording bugfixes, new Spanish translations, or added question sequences arising from unexpected events (e.g., COVID-19). In other words, they are conceptually identical for the most part and lack the kinds of wholesale rewrites of sections or sequences that can be made during the preproduction phase between field periods. Therefore, HRS has always assumed (and found in practice in our interviewer-administered-only Blaise 4 environment before 2018) that data could be easily transferred between these versions within a "wave." In particular, HRS assumed that there would be no problems when a case is suspended in one datamodel version but must be resumed in a later one (incidentally, the versions may not even be consecutive, since there is occasionally a long lag between suspend and resume, based on the respondent's schedule or wishes).

Unfortunately, HRS discovered that it was not possible to resume a case in a newer datamodel version that was suspended in a previous datamodel with session data intact. By this point, HRS had learned to preserve the session data files as described above, so this problem revealed itself only after having solved the previous one. What is more, it turned out that this newly discovered problem occurred even if there were no harmful changes between the two datamodels in question (as of Blaise release 5.10.6).

To take a step back, HRS at first thought that data incompatibility might be the cause and looked deeper into understanding harmless/harmful change concepts. As implied above, the key requirement for preserving session data between sessions in different datamodel releases is that the new datamodel must not have harmful changes. Because the rules governing harmful or harmless changes can be a little vague, it is necessary to resort to some form of testing or tool to detect harmful changes. For example, Blaise 5 ships with a sample Manipula script for detecting harmful changes, which can be found in the samples folder:

• \Documents\Blaise5\Samples\Specific Features\Manipula\HarmlessChanges\HarmlessChanges.bsol

This script compares two datamodels and reports whether it detects either no changes or only harmless changes, or, as a third possible outcome, it lists all harmful changes. HRS had previously tried this Manipula sample, but it did not work well due to a bug causing a listing of more than a thousand false positives. As a result, HRS was essentially guessing at whether changes were harmful and was not fully engaged in documenting this. When HRS began to run into these datamodel version update problems during the 2020 field period, CBS was called in to help and quickly provided a small code change to make this script work as intended for HRS. CBS also provided sample API code for building a tool to test datamodel updates with harmless changes, along roughly the following lines:

```
private void ApplyHarmless(string oldDMbdixFilename, string newDMbmixFilename)
{
        DataLinkAPI.IDataLink5 dl = DataLinkAPI.DataLinkManager.GetDataLink(oldDMbdixFilename) as
        DataLinkAPI.IDataLink5;
        dl.ApplyHarmlessChanges(newDMbmixFilename);
}
```

HRS was then able to verify compatibility between datamodels and was immediately presented with the next problem: session data was *always* treated as incompatible, regardless of the harmless change determination in the version of Blaise HRS was using for development at the time (5.10.6), as well as earlier versions.

In order to solve this problem, CBS provided a special unsupported 5.10.10 release to accommodate this need for the then upcoming start of HRS 2022 interviewing. Basically, in that version, the

> back to Table of Contents

“ApplyHarmlessChanges” would successfully run against the session database, allowing a suspended case to be resumed in a new datamodel version, provided that only harmless changes were present between the two. This capability should be a normal part of the Blaise 5 feature set from 5.13 on (note that HRS never tested these features in 5.11 or 5.12, but CBS had specifically mentioned 5.13 in this regard, so that would seem to be the best starting point), meaning updates that preserve session data are now a normal supported part of Blaise 5 operation. So far, our testing in 5.13 has borne this out. In addition to the API functionality, a number of options now (as of 5.13) appear in the Blaise 5 Server Manager when updating an existing datamodel that allow for this, as will be discussed later.

7. Testing Harmless Change Compatibility

This new capability (in 5.10.10 and 5.13.x) has allowed HRS to add functions to test session data migration between datamodels in our testing tools. So, to complement efforts by programmers to keep harmful changes in mind while programming (the first line of defense), and to allay any lingering doubt about the aforementioned Manipula script’s accuracy (which is the second line of defense), HRS can actually test suspending a case in one datamodel, updating the datamodel, and attempting to resume using the main HRS case testing tool. Additionally, the staff at SRO, who handle field operations, do a final test using tools with the aforementioned code. As a result, session data problems due to this set of issues have been minimized in the 2022 field period.

Following is an illustrative process (as of Blaise 5.10.10) for testing datamodel migration with harmless changes to preserve session data, which works in a normal server deployment:

1. Copy a preloaded .bdbx into the .bpkg of the initial datamodel (or equivalent process to handle preload).

2. In Server Manager, install the .bpkg of the initial datamodel.

3. In Server Manager, start in browser with normal arguments, test some “unfoldings” and signals, and suspend.

4. Run a tool with the ApplyHarmlessChanges API function as administrator against the deployed .bdix in the \Blaise5\Surveys folder and the .bmix from the new datamodel with the harmless changes.

5. In Server Manager, start in browser with normal arguments. It will then resume in the correct place in the new datamodel with session data intact.

The problem with updated datamodels that was described above had one particularly troublesome twist that will require some extra detail and an alternative set of testing steps to describe. HRS has been handling web self-interview and telephone or face-to-face interviewer-assisted cases using different systems so far, though efforts are being made to eventually merge everything into one. The web self-interview cases run from a server in a conventional way, but the interviewer-assisted cases run in a locked-down laptop environment in Windows DEP standalone mode. A lot of extra steps were involved in making datamodel updates work in that environment.

Following is an illustrative process (as of Blaise 5.10.10) for testing datamodel migration with harmless changes to preserve session data, which works in Standalone mode:

1. Copy a preloaded .bdbx into the .bpkg of the initial datamodel (or equivalent process to handle preload).

2. Copy this .bpkg of the initial datamodel into the same folder as the DEP (and its associated .dlls).

> back to Table of Contents

3. Run the special DEP with the commandline argument -RunMode:ThickClient (and other normal arguments), test some "unfoldings" and signals, and suspend.

4. Run a tool with the ApplyHarmlessChanges API function against the deployed .bdix and the .bmix from the new datamodel with the harmless changes.

5. Copy the three files that are updated by the ApplyHarmlessChanges tool from the deploy folder into the same preloaded .bpkg from Step 2.

6. Run the special DEP with the commandline argument -RunMode:ThickClient (and other normal arguments). It will then resume in the correct place in the new datamodel with session data intact.

8. HRS 2022

In the 2022 field period, HRS encountered another new twist on session data preservation. This was connected with a new style of mode switch being supported in 2022. In this approach, cases could be passed from the standard web server deployment into the offline standalone laptop environment. To make this approach work for HRS, SRO uses a heavily modified custom DEP, based on CBS examples and assistance. The sync process in this custom DEP presents issues with preserving session data. In a variation on the previously discussed suspend-resume issue, HRS had cases that switched from web-based self-interview to telephone interview from an offline laptop, and many of these cases had been started before the switch. It was discovered that starting the case prior to the switch caused many calculations and assignments to happen, even if it was suspended on the first screen, and those were reflected in the .bdbx that was being transferred, causing problems. If the case was resumed using only the .bdbx without the session data in that situation, you would get some corrupted data. For example, the aforementioned problem where children-in-law got deleted cropped up as a result of this.

A number of discussions with CBS have resulted in a possible fix coming down the road in Blaise 5.14 or 5.15. Just like the previous changes concerning preserving session data during datamodel updates solved that problem in the HRS 2022 field period, HRS is hopeful that these upcoming fixes will eliminate one more cause of session-data-related difficulties in the HRS 2024 field period.

One clear trend in all of this is that each set of Blaise fixes have unmasked new deeper issues with HRS and session data over time, but it is also the case that the magnitude of the problem keeps shrinking wave-on-wave. Whereas the first issue impacted potentially thousands of cases, the subsequent issue may have affected hundreds of cases, and the most recent is impacting perhaps less than a hundred. Additionally, new versions are bringing more features to address these issues. For example, the Blaise 5 Server Manager presents more options when installing surveys over previously existing ones in Blaise 5.13. Some of those will be discussed below. Overall, our hope would be that questions about what happens to session data will eventually permeate every part of the system.

9. Preserving Session Data After Completion

One last concern that fits by virtue of being about session data but is not a bug or unsolved issue of any sort is the preservation of session data after a case is completed. So far, this paper has discussed mainly preservation of session data on suspend and resume, first within the same datamodel, then across different datamodels, and finally through the custom DEP sync process. All of those were concerned with successfully completing the interview when multiple sessions were involved. This additional issue of long-term preservation of session data is for all interviews. To some extent, what HRS wants from this is already handled by audit trail data—that is, the ability to go back later and review a case that had

> back to Table of Contents

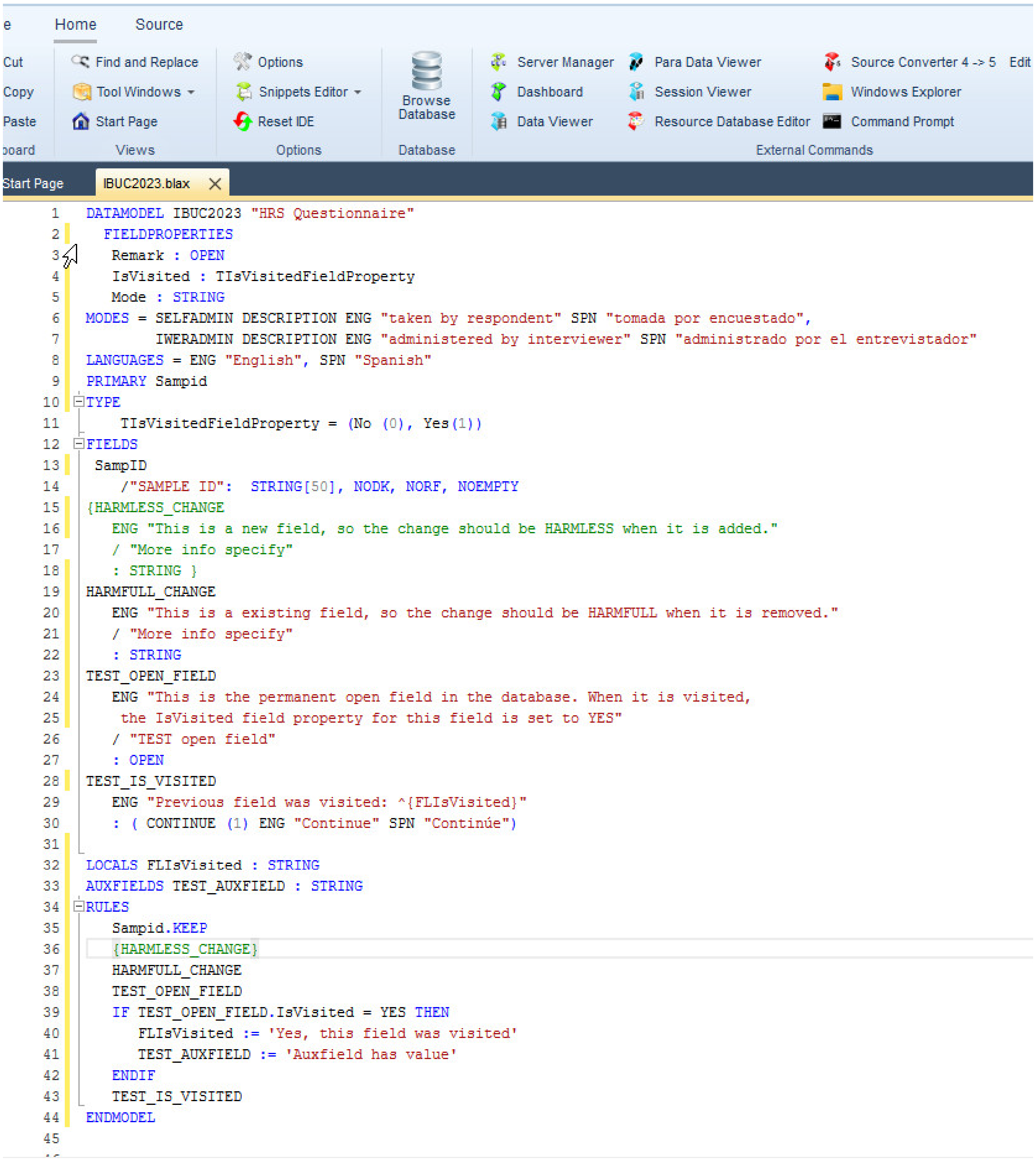
problems, determine what went wrong, and recover missing answers, as well as other related needs. Having session data available for later review would expand on that since it contains *all* the working data, such as auxfields. This could make later investigations easier by providing a more direct way to see what might have caused a particular problem.

HRS did not realize that session data was deleted upon case completion until well into the 2020 field period. In response, SRO was able to develop an approach for HRS that uses SQL Server triggers (the self-administered mode of HRS stores data in SQL Server) to copy the session data to another database right before any delete command is executed. This has worked well for the parts of HRS 2022 using SQL Server. In the future, CBS has indicated that new options may be added to natively prevent deletion of session data, if desired.

> back to Table of Contents

10. Examples

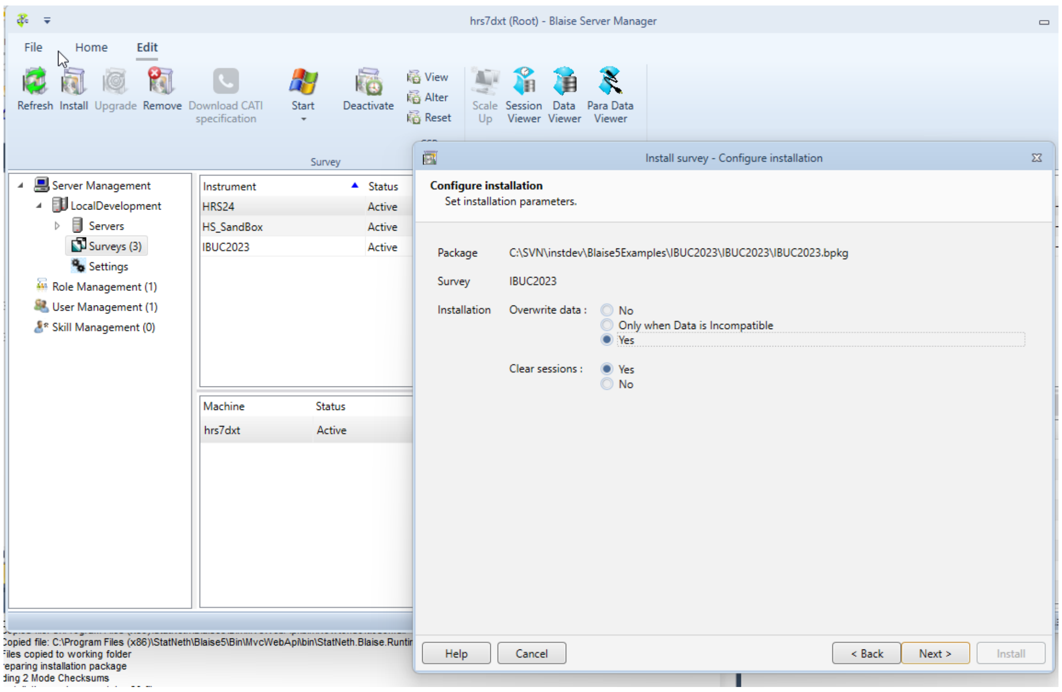
To illustrate harmful/harmless changes, we use a simple datamodel. Note that there is a field HARMLESS_CHANGE that is off the route for now. We plan to add it later on to demonstrate that this change is harmless.



```
DATAMODEL IBUC2023 "HRS Questionnaire"
  FIELDPROPERTIES
   Remark : OPEN
   IsVisited : TIsVisitedFieldProperty
   Mode : STRING
MODES = SELFADMIN DESCRIPTION ENG "taken by respondent" SPN "tomada por encuestado",
        IWERADMIN DESCRIPTION ENG "administered by interviewer" SPN "administrado por el entrevistador"
LANGUAGES = ENG "English", SPN "Spanish"
PRIMARY Sampid
TYPE
    TIsVisitedFieldProperty = (No (0), Yes(1))
FIELDS
 SampID
    /"SAMPLE ID":  STRING[50], NODK, NORF, NOEMPTY
{HARMLESS_CHANGE
   ENG "This is a new field, so the change should be HARMLESS when it is added."
   / "More info specify"
   : STRING }
HARMFULL_CHANGE
   ENG "This is a existing field, so the change should be HARMFULL when it is removed."
   / "More info specify"
   : STRING
TEST_OPEN_FIELD
   ENG "This is the permanent open field in the database. When it is visited,
    the IsVisited field property for this field is set to YES"
   / "TEST open field"
   : OPEN
TEST_IS_VISITED
   ENG "Previous field was visited: ^{FLIsVisited}"
   : ( CONTINUE (1) ENG "Continue" SPN "Continúe")

LOCALS FLIsVisited : STRING
AUXFIELDS TEST_AUXFIELD : STRING
RULES
   Sampid.KEEP
   {HARMLESS_CHANGE}
   HARMFULL_CHANGE
   TEST_OPEN_FIELD
   IF TEST_OPEN_FIELD.IsVisited = YES THEN
      FLIsVisited := 'Yes, this field was visited'
      TEST_AUXFIELD := 'Auxfield has value'
   ENDIF
   TEST_IS_VISITED
ENDMODEL
```

> back to Table of Contents

After installing and deploying this survey, we are able to view session data from Blaise Server Manager. At the first installation, we choose the options to overwrite the data and clear sessions.



Instrument	Status
HRS24	Active
HS_SandBox	Active
IBUC2023	Active

Machine	Status
hrs7dxt	Active

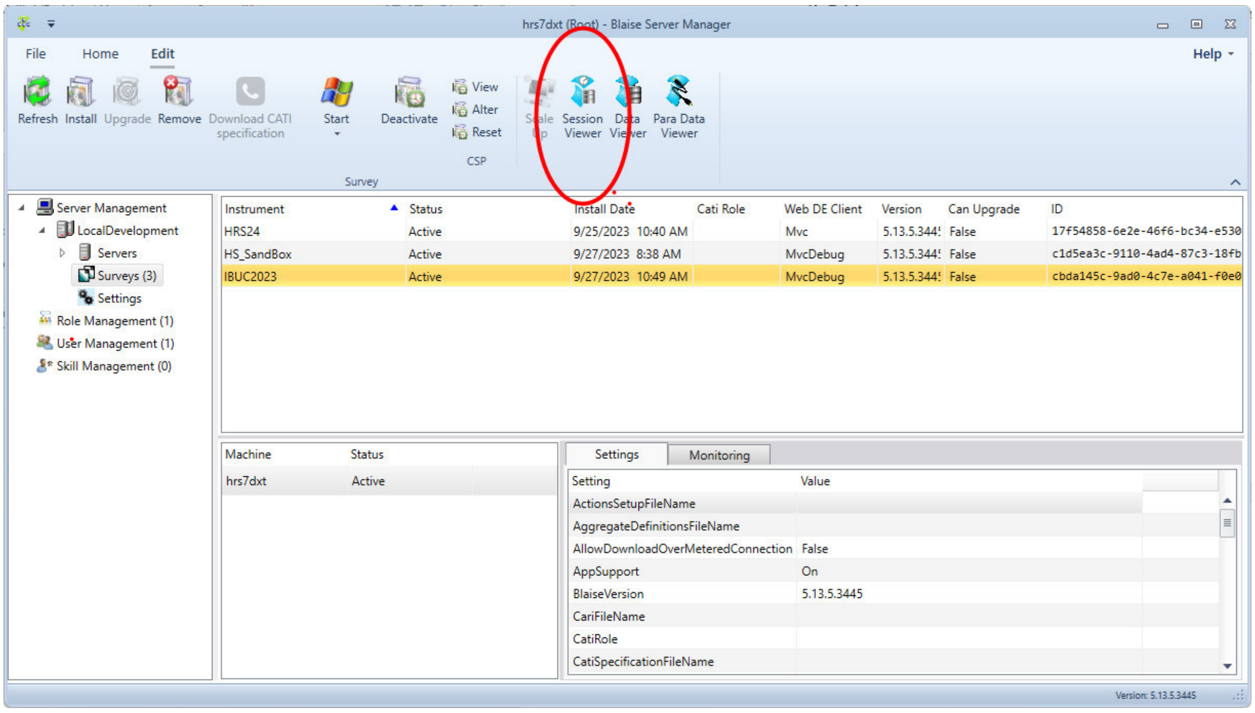
Install survey - Configure installation

Configure installation
Set installation parameters.

Package	C:\SVN\instdev\Blaise5Examples\IBUC2023\IBUC2023\IBUC2023.bpkg	
Survey	IBUC2023	
Installation	Overwrite data :	No [✓ Yes] Only when Data is Incompatible
	Clear sessions :	[✓ Yes] No

```
Copied file: C:\Program Files (x86)\StatNeth\Blaise5\Bin\MvcWebApi\bin\StatNeth.Blaise.Runtir
Files copied to working folder
reparing installation package
ding 2 Mode Checksums
```

After answering a series of questions, we are able to view the data in fields and auxfields through Session Viewer. You can also do it from the Blaise Control Center.



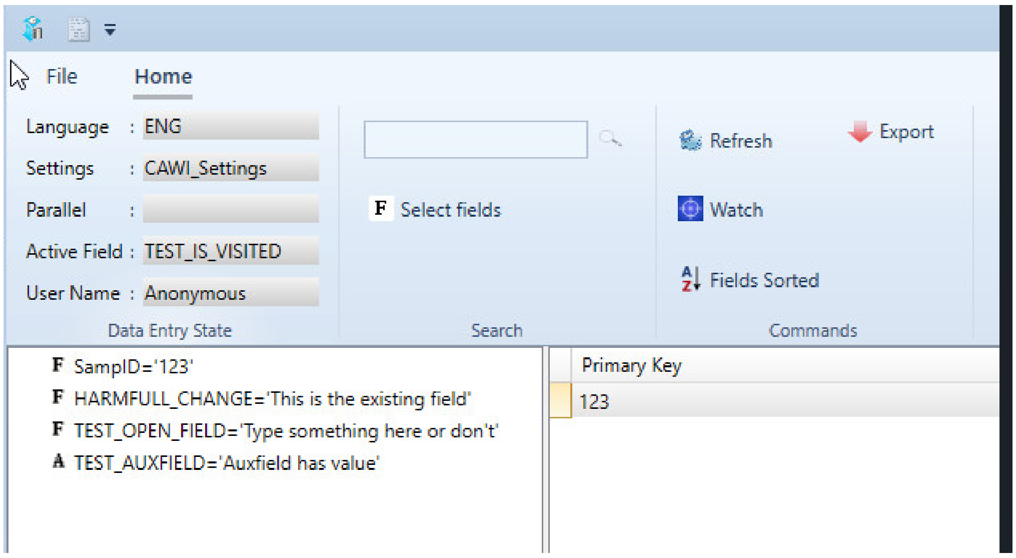
Instrument	Status	Install Date	Cati Role	Web DE Client	Version	Can Upgrade	ID
HRS24	Active	9/25/2023 10:40 AM		Mvc	5.13.5.344!	False	17f54858-6e2e-46f6-bc34-e530
HS_SandBox	Active	9/27/2023 8:38 AM		MvcDebug	5.13.5.344!	False	c1d5ea3c-9110-4ad4-87c3-18fb
IBUC2023	Active	9/27/2023 10:49 AM		MvcDebug	5.13.5.344!	False	cbda145c-9ad0-4c7e-a041-f0e0

Machine	Status
hrs7dxt	Active

Setting	Value
ActionsSetupFileName	
AggregateDefinitionsFileName	
AllowDownloadOverMeteredConnection	False
AppSupport	On
BlaiseVersion	5.13.5.3445
CariFileName	
CatiRole	
CatiSpecificationFileName	

> back to Table of Contents

Provided that your survey has a Primary Key, defined in your datamodel, you can view your session data:



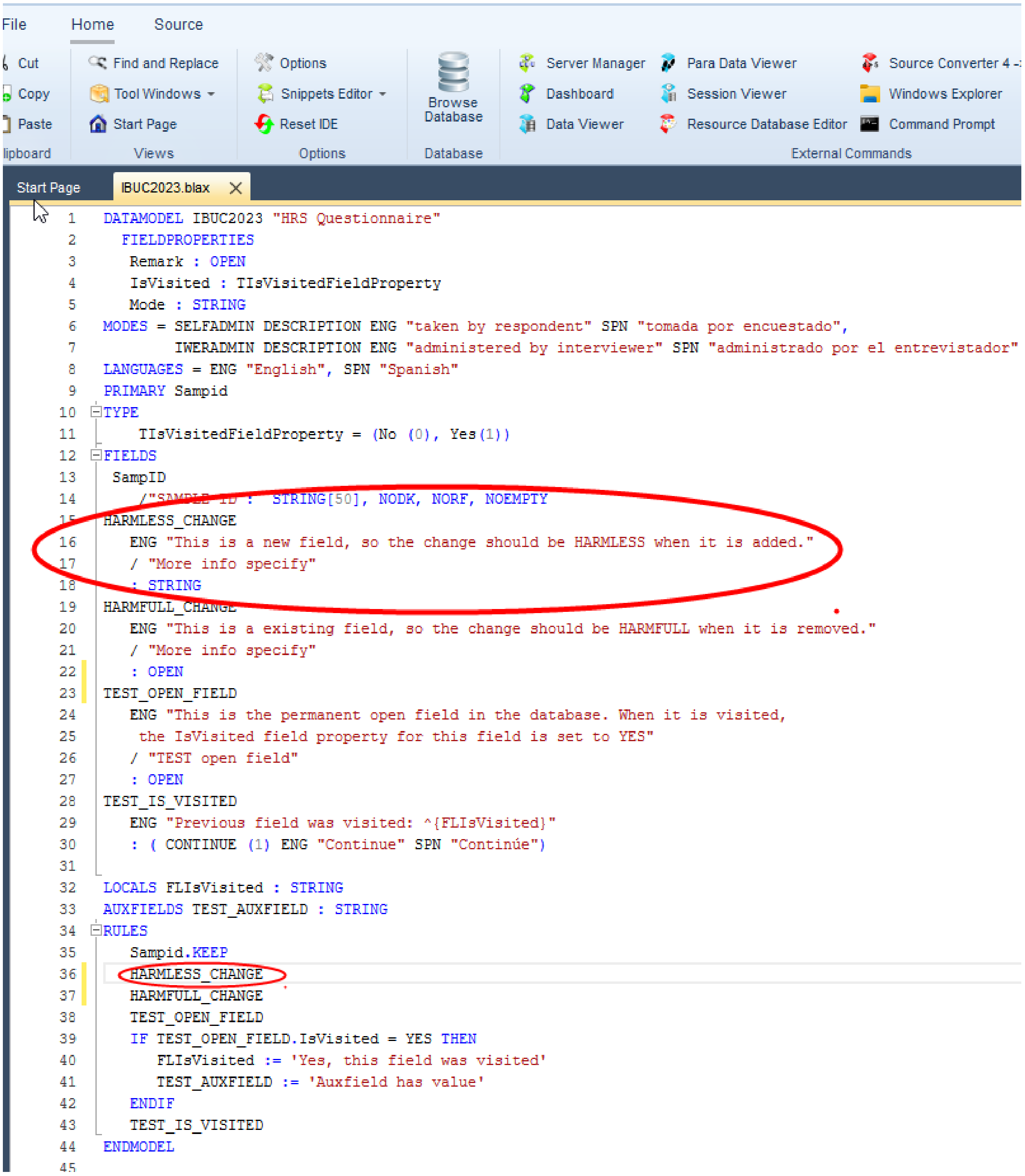
Language : ENG
Settings : CAWI_Settings
Parallel :
Active Field : TEST_IS_VISITED
User Name : Anonymous

F SampID='123'
F HARMFULL_CHANGE='This is the existing field'
F TEST_OPEN_FIELD='Type something here or don't'
A TEST_AUXFIELD='Auxfield has value'

Primary Key
123

> back to Table of Contents

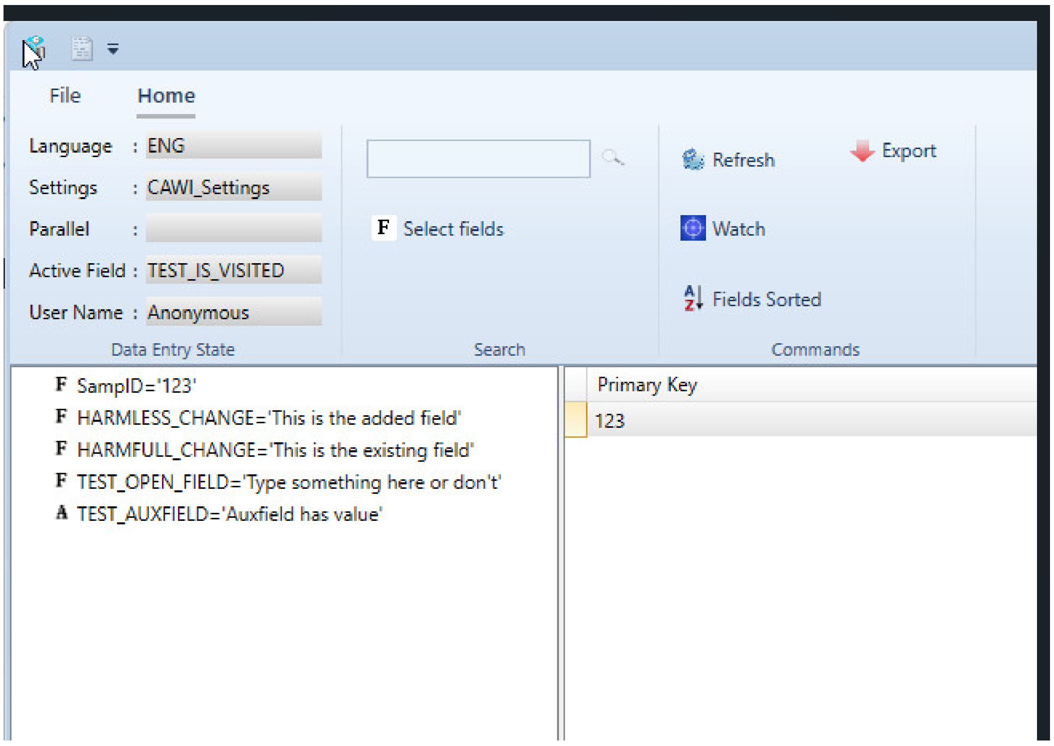
Now, suppose we want to add a new field to a datamodel that is already deployed in the field.



```
DATAMODEL IBUC2023 "HRS Questionnaire"
  FIELDPROPERTIES
   Remark : OPEN
   IsVisited : TIsVisitedFieldProperty
   Mode : STRING
MODES = SELFADMIN DESCRIPTION ENG "taken by respondent" SPN "tomada por encuestado",
        IWERADMIN DESCRIPTION ENG "administered by interviewer" SPN "administrado por el entrevistador"
LANGUAGES = ENG "English", SPN "Spanish"
PRIMARY Sampid
TYPE
    TIsVisitedFieldProperty = (No (0), Yes(1))
FIELDS
 SampID
    /"SAMPLE ID :  STRING[50], NODK, NORF, NOEMPTY
HARMLESS_CHANGE
   ENG "This is a new field, so the change should be HARMLESS when it is added."
   / "More info specify"
   : STRING
HARMFULL_CHANGE
   ENG "This is a existing field, so the change should be HARMFULL when it is removed."
   / "More info specify"
   : OPEN
TEST_OPEN_FIELD
   ENG "This is the permanent open field in the database. When it is visited,
    the IsVisited field property for this field is set to YES"
   / "TEST open field"
   : OPEN
TEST_IS_VISITED
   ENG "Previous field was visited: ^{FLIsVisited}"
   : ( CONTINUE (1) ENG "Continue" SPN "Continúe")

LOCALS FLIsVisited : STRING
AUXFIELDS TEST_AUXFIELD : STRING
RULES
   Sampid.KEEP
   HARMLESS_CHANGE
   HARMFULL_CHANGE
   TEST_OPEN_FIELD
   IF TEST_OPEN_FIELD.IsVisited = YES THEN
      FLIsVisited := 'Yes, this field was visited'
      TEST_AUXFIELD := 'Auxfield has value'
   ENDIF
   TEST_IS_VISITED
ENDMODEL
```

> back to Table of Contents

This change would be harmless. After installing a new survey over the old one, the new field is on the route and the session data is preserved. Note in the following screenshot that the auxfield data is also preserved. Prior to Blaise 5.10.10 this would not have worked.



Language : ENG
Settings : CAWI_Settings
Parallel :
Active Field : TEST_IS_VISITED
User Name : Anonymous

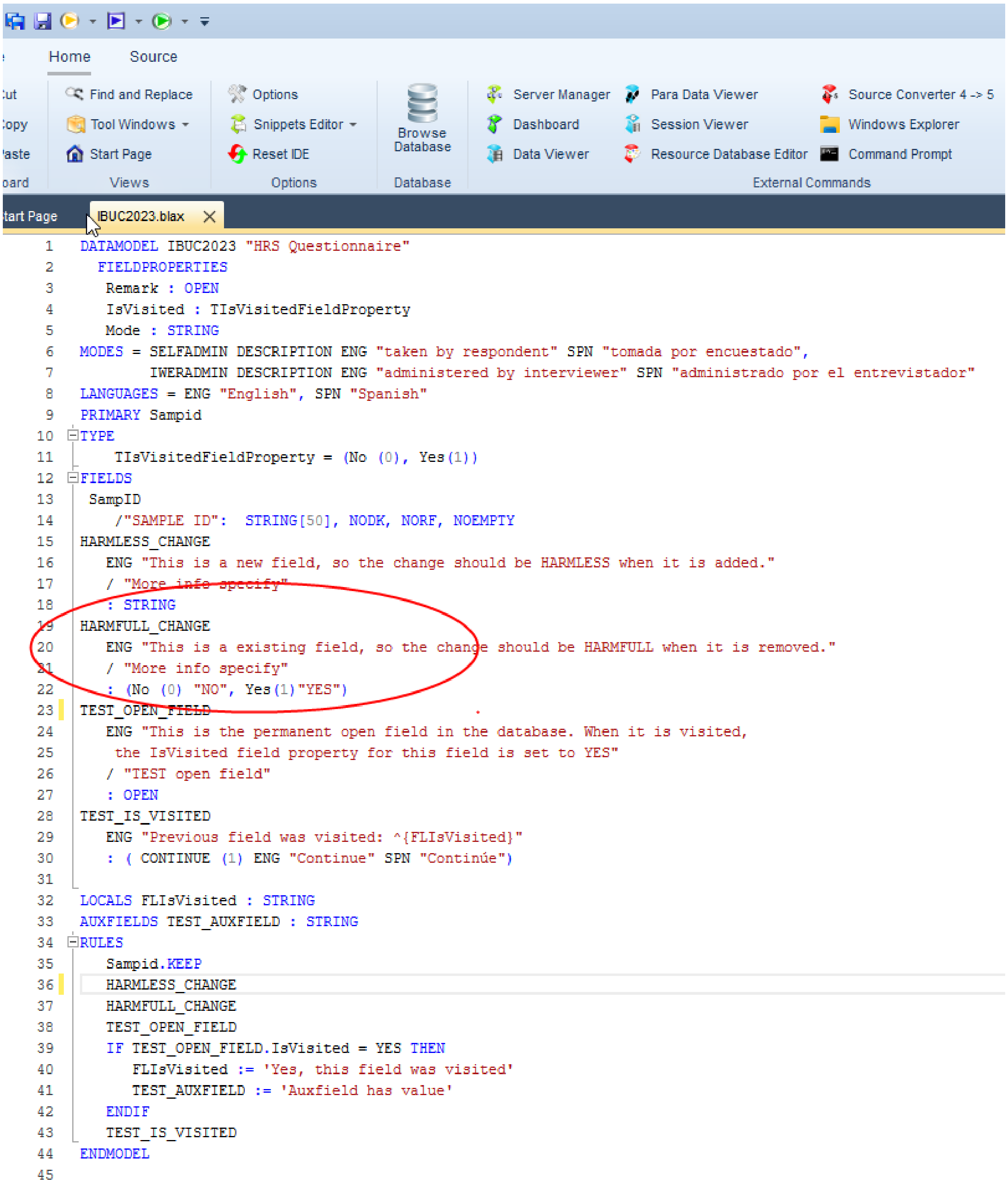
Data Entry State

F SampID='123'
F HARMLESS_CHANGE='This is the added field'
F HARMFULL_CHANGE='This is the existing field'
F TEST_OPEN_FIELD='Type something here or don't'
A TEST_AUXFIELD='Auxfield has value'

Primary Key
123

> back to Table of Contents

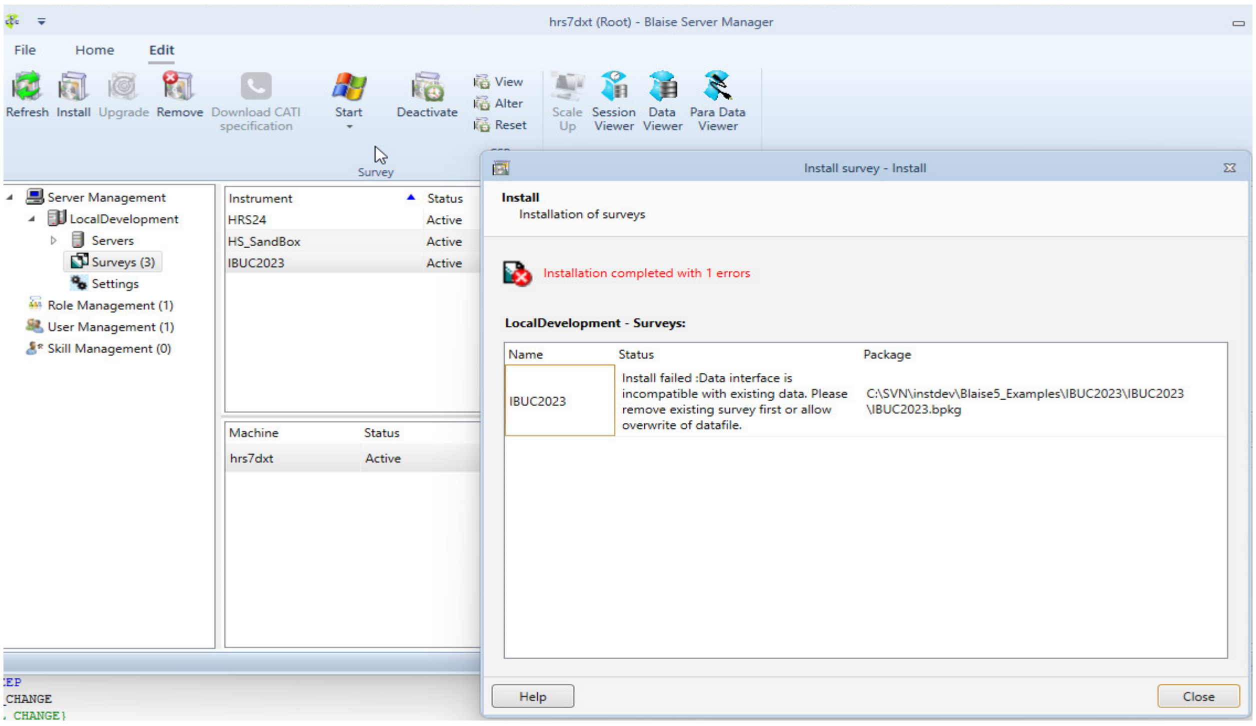
Now, suppose there is need to change a field type in the survey that is already deployed. Here, we are changing the type from OPEN to ENUMERATION for HARMFULL_CHANGE field.



```
DATAMODEL IBUC2023 "HRS Questionnaire"
  FIELDPROPERTIES
   Remark : OPEN
   IsVisited : TIsVisitedFieldProperty
   Mode : STRING
MODES = SELFADMIN DESCRIPTION ENG "taken by respondent" SPN "tomada por encuestado",
        IWERADMIN DESCRIPTION ENG "administered by interviewer" SPN "administrado por el entrevistador"
LANGUAGES = ENG "English", SPN "Spanish"
PRIMARY Sampid
TYPE
    TIsVisitedFieldProperty = (No (0), Yes(1))
FIELDS
 SampID
    /"SAMPLE ID":  STRING[50], NODK, NORF, NOEMPTY
HARMLESS_CHANGE
   ENG "This is a new field, so the change should be HARMLESS when it is added."
   / "More info specify"
   : STRING
HARMFULL_CHANGE
   ENG "This is a existing field, so the change should be HARMFULL when it is removed."
   / "More info specify"
   : (No (0) "NO", Yes(1)"YES")
TEST_OPEN_FIELD
   ENG "This is the permanent open field in the database. When it is visited,
    the IsVisited field property for this field is set to YES"
   / "TEST open field"
   : OPEN
TEST_IS_VISITED
   ENG "Previous field was visited: ^{FLIsVisited}"
   : ( CONTINUE (1) ENG "Continue" SPN "Continúe")

LOCALS FLIsVisited : STRING
AUXFIELDS TEST_AUXFIELD : STRING
RULES
   Sampid.KEEP
   HARMLESS_CHANGE
   HARMFULL_CHANGE
   TEST_OPEN_FIELD
   IF TEST_OPEN_FIELD.IsVisited = YES THEN
      FLIsVisited := 'Yes, this field was visited'
      TEST_AUXFIELD := 'Auxfield has value'
   ENDIF
   TEST_IS_VISITED
ENDMODEL
```

> back to Table of Contents

Since the type of the field had been changed, we were unable to install the survey over the existing one without overwriting the old session data. An attempt to do so will result in the following error message:



Instrument	Status
HRS24	Active
HS_SandBox	Active
IBUC2023	Active

Machine	Status
hrs7dxt	Active

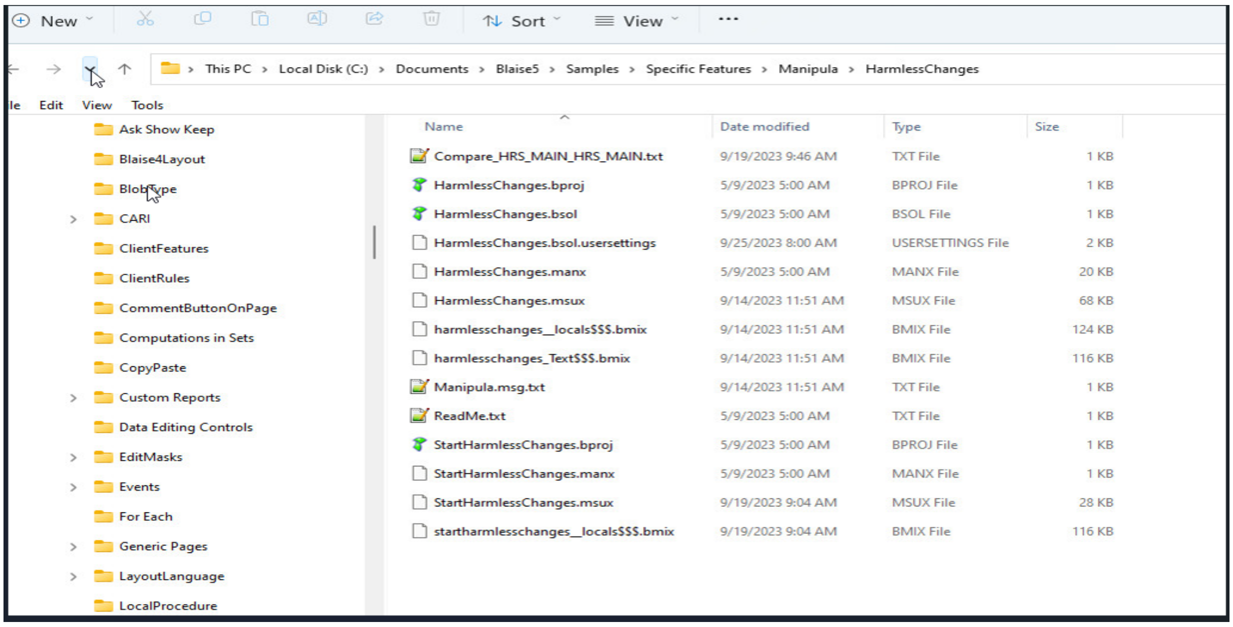
Install
Installation of surveys

Installation completed with 1 errors

LocalDevelopment - Surveys:

Name	Status	Package
IBUC2023	Install failed :Data interface is incompatible with existing data. Please remove existing survey first or allow overwrite of datafile.	C:\SVN\instdev\Blaise5_Examples\IBUC2023\IBUC2023\IBUC2023.bpkg

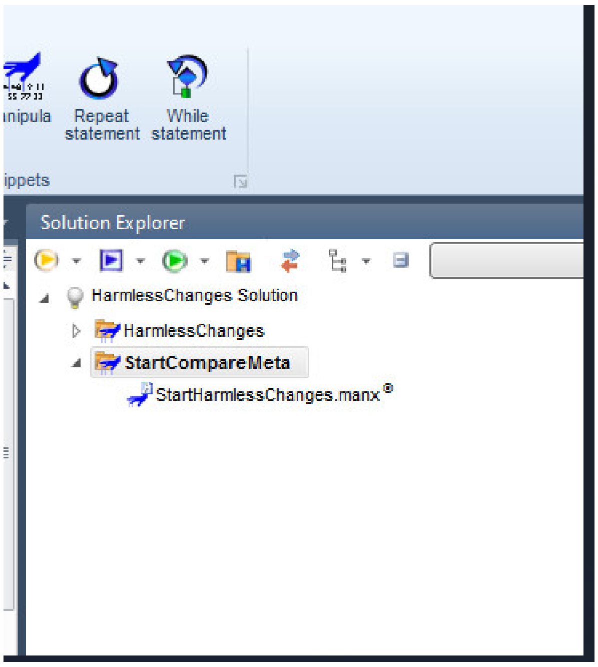
Given the complexity of HRS and the number of datamodels released into the field after the data collection has started, we always run a harmless change tool to check for harmful changes. This useful tool provided by the Blaise team is located here:



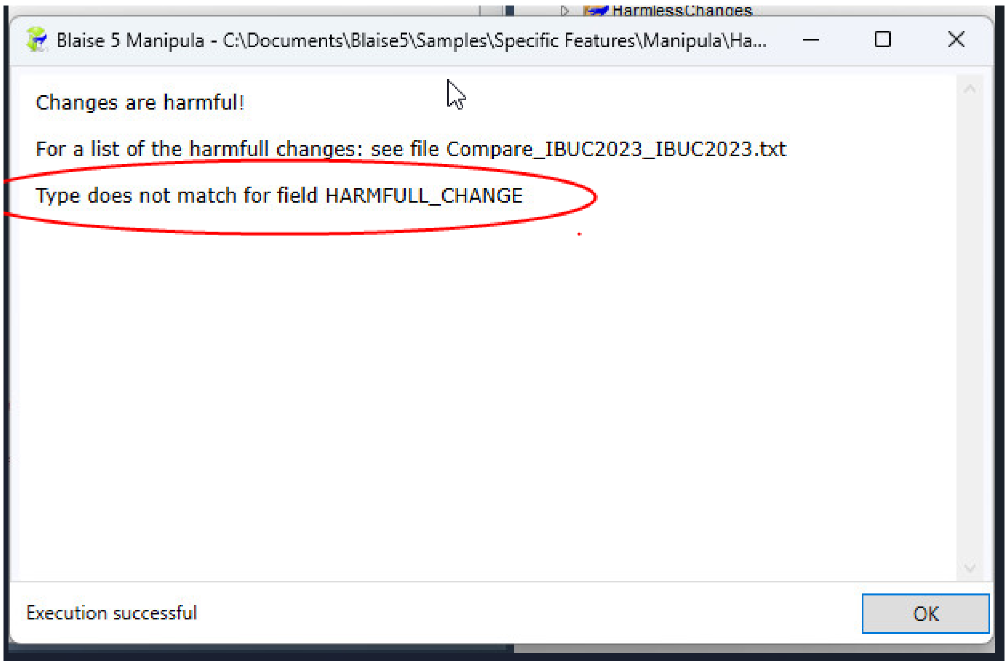
Name	Date modified	Type	Size
Compare_HRS_MAIN_HRS_MAIN.txt	9/19/2023 9:46 AM	TXT File	1 KB
HarmlessChanges.bproj	5/9/2023 5:00 AM	BPROJ File	1 KB
HarmlessChanges.bsol	5/9/2023 5:00 AM	BSOL File	1 KB
HarmlessChanges.bsol.usersettings	9/25/2023 8:00 AM	USERSETTINGS File	2 KB
HarmlessChanges.manx	5/9/2023 5:00 AM	MANX File	20 KB
HarmlessChanges.msux	9/14/2023 11:51 AM	MSUX File	68 KB
harmlesschanges__locals$$$.bmix	9/14/2023 11:51 AM	BMIX File	124 KB
harmlesschanges_Text$$$.bmix	9/14/2023 11:51 AM	BMIX File	116 KB
Manipula.msg.txt	9/14/2023 11:51 AM	TXT File	1 KB
ReadMe.txt	5/9/2023 5:00 AM	TXT File	1 KB
StartHarmlessChanges.bproj	5/9/2023 5:00 AM	BPROJ File	1 KB
StartHarmlessChanges.manx	5/9/2023 5:00 AM	MANX File	1 KB
StartHarmlessChanges.msux	9/19/2023 9:04 AM	MSUX File	28 KB
startharmlesschanges__locals$$$.bmix	9/19/2023 9:04 AM	BMIX File	116 KB

> back to Table of Contents

Compile and run HarmlessChange.bsol. You will be asked to provide the locations of the old and the new .bmix files that need to be compared to detect if there are harmful changes.



In our example, the report is generated, warning of the field type mismatch in the datamodels.



Blaise 5 Manipula - C:\Documents\Blaise5\Samples\Specific Features\Manipula\Ha...
OK

Changes are harmful!

For a list of the harmfull changes: see file Compare_IBUC2023_IBUC2023.txt

Type does not match for field HARMFULL_CHANGE

Execution successful

> back to Table of Contents

To remedy the problem, HRS uses the following techniques:

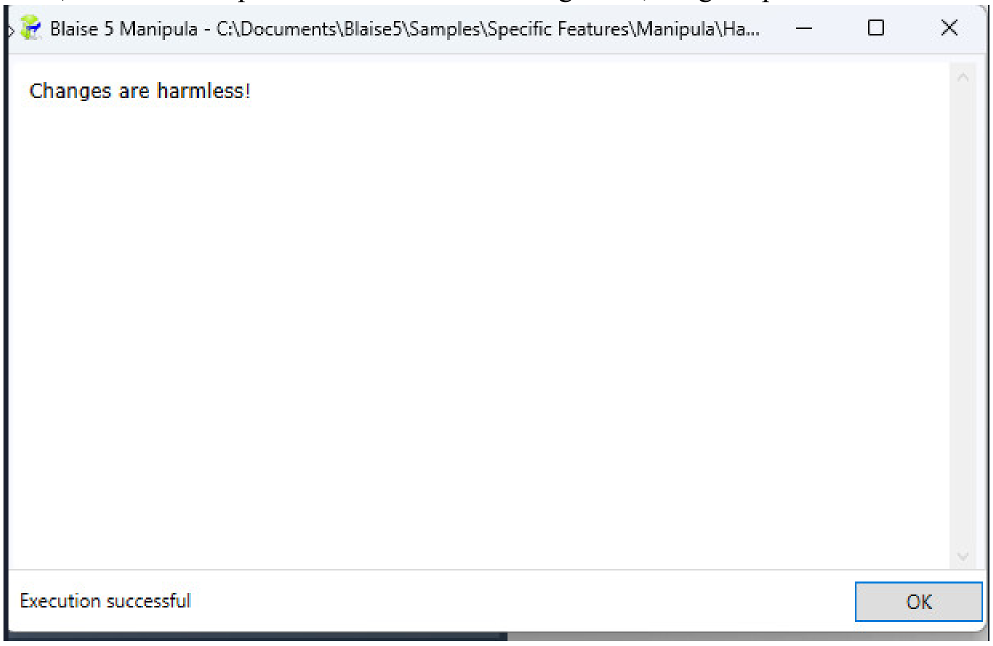
1. We create a new field that has the desired type without commenting out the old one.

```
{keep the old field in the datamodel}
HARMFULL_CHANGE
   ENG "This is a existing field, so the change should be HARMFULL when it is removed."
   / "More info specify"
:OPEN
 {Create a new field with the desired type}
HARMFULL_CHANGE_NEW
   ENG "This is a existing field, so the change should be HARMFULL when it is removed."
   / "More info specify"
   : (No (0) "NO", Yes(1)"YES")
```

2. We keep the old field on the route but put an IF-THEN statement that is never TRUE around it.

```
RULES
   Sampid.KEEP
  {Keep the old field on the route, but so that it is never asked}
   IF 1  = 2 THEN
      HARMLESS_CHANGE
   ENDIF
   HARMFULL_CHANGE_NEW
   HARMFULL_CHANGE
```

Now, when we compile and run HarmlessChange.bsol, we get a positive result!



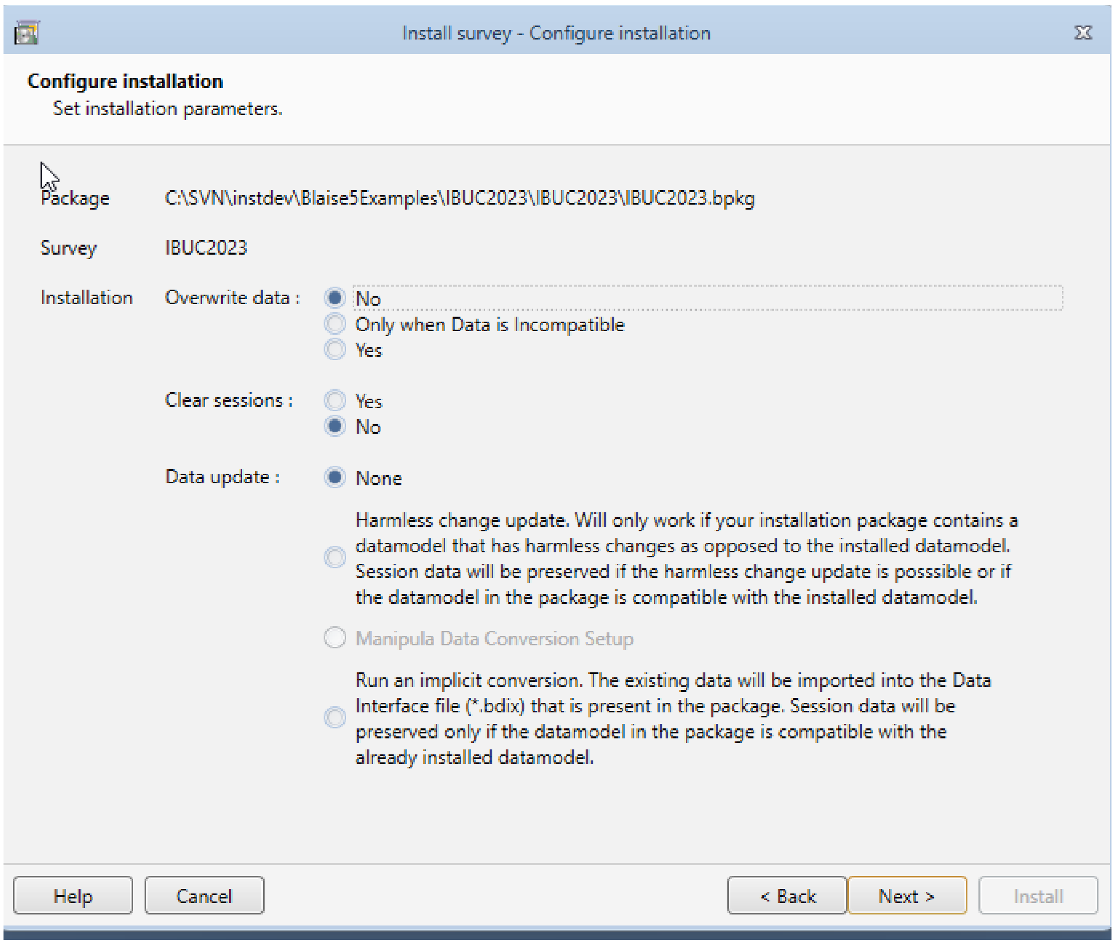
Blaise 5 Manipula - C:\Documents\Blaise5\Samples\Specific Features\Manipula\Ha...

```
Changes are harmless!
```

Execution successful

> back to Table of Contents

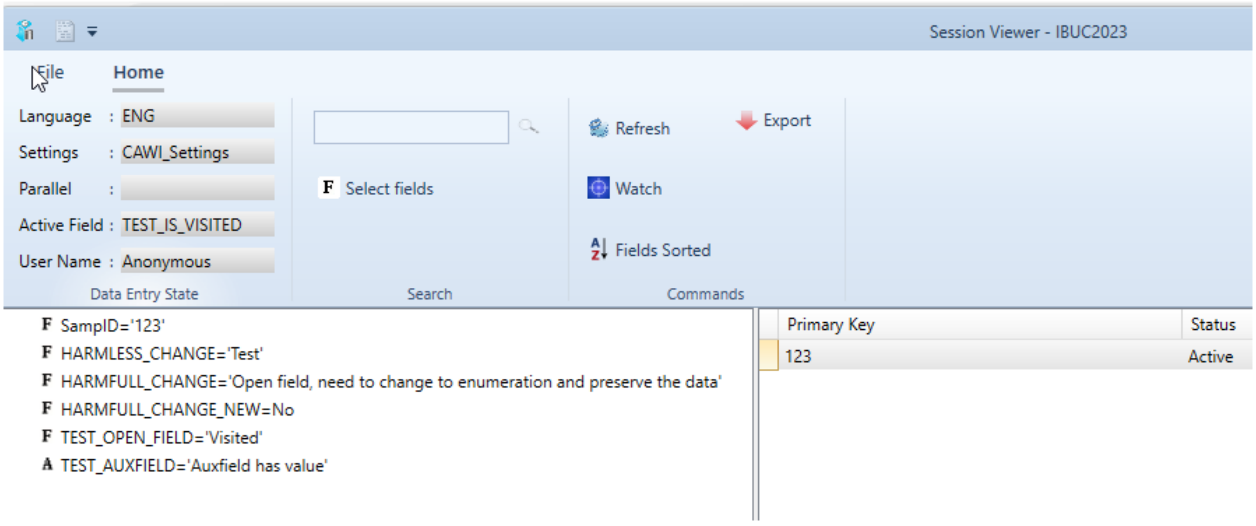
When we install the new survey over the old one, we use the following options to preserve the session data:


Install survey - Configure installation

Configure installation
Set installation parameters.

Package	C:\SVN\instdev\Blaise5Examples\IBUC2023\IBUC2023\IBUC2023.bpkg	
Survey	IBUC2023	
Installation	Overwrite data :	[✓ No] Only when Data is Incompatible Yes
	Clear sessions :	Yes [✓ No]
	Data update :	[✓ None]
		Harmless change update. Will only work if your installation package contains a datamodel that has harmless changes as opposed to the installed datamodel. Session data will be preserved if the harmless change update is posssible or if the datamodel in the package is compatible with the installed datamodel.
		Manipula Data Conversion Setup
		Run an implicit conversion. The existing data will be imported into the Data Interface file (*.bdix) that is present in the package. Session data will be preserved only if the datamodel in the package is compatible with the already installed datamodel.

When viewing the session data for the new survey, we can see that the old session data is preserved.


Session Viewer - IBUC2023

Language	ENG
Settings	CAWI_Settings
Parallel	
Active Field	TEST_IS_VISITED
User Name	Anonymous

F SampID='123'
F HARMLESS_CHANGE='Test'
F HARMFULL_CHANGE='Open field, need to change to enumeration and preserve the data'
F HARMFULL_CHANGE_NEW=No
F TEST_OPEN_FIELD='Visited'
A TEST_AUXFIELD='Auxfield has value'

Primary Key	Status
123	Active

> back to Table of Contents

11. Appendix—Source Code

```
DATAMODEL IBUC2023 "HRS Questionnaire"
  FIELDPROPERTIES
             Remark : OPEN
             IsVisited : TIsVisitedFieldProperty
             Mode : STRING
MODES = SELFADMIN DESCRIPTION ENG "taken by respondent" SPN "tomada por encuestado",
                  IWERADMIN DESCRIPTION ENG "administered by interviewer" SPN "administrado por el
entrevistador"
LANGUAGES = ENG "English", SPN "Spanish"
PRIMARY Sampid
TYPE
    TIsVisitedFieldProperty = (No (0), Yes(1))
FIELDS
 SampID
    /"SAMPLE ID": STRING[50], NODK, NORF, NOEMPTY
HARMLESS_CHANGE
             ENG "This is a new field, so the change should be HARMLESS when it is added."
             / "More info specify"
             : STRING
{keep the old field in the datamodel}
HARMFULL_CHANGE
             ENG "This is a existing field, so the change should be HARMFULL when it is removed."
             / "More info specify"
:OPEN
 {Create a new field with the desired type}
HARMFULL_CHANGE_NEW
             ENG "This is a existing field, so the change should be HARMFULL when it is removed."
             / "More info specify"
             : (No (0) "NO", Yes(1)"YES")

TEST_OPEN_FIELD
             ENG "This is the permanent open field in the database. When it is visited,
    the IsVisited field property for this field is set to YES"
             / "TEST open field"
             : OPEN
TEST_IS_VISITED
             ENG "Previous field was visited: ^{FLIsVisited}"
             : ( CONTINUE (1) ENG "Continue" SPN "Continúe")

LOCALS FLIsVisited : STRING
AUXFIELDS TEST_AUXFIELD : STRING
RULES
   Sampid.KEEP
   HARMLESS_CHANGE
  {Keep the old field on the route, but so that it is never asked}
```

> back to Table of Contents

```
   IF 1 = 2 THEN
      HARMFULL_CHANGE
  ENDIF
   HARMFULL_CHANGE_NEW
   TEST_OPEN_FIELD
   IF TEST_OPEN_FIELD.IsVisited = YES THEN
      FLIsVisited := 'Yes, this field was visited'
      TEST_AUXFIELD := 'Auxfield has value'
   ENDIF
   TEST_IS_VISITED
ENDMODEL
```

> back to Table of Contents

Design Considerations for Web and CAPI Multimode Using Blaise 5

Todd Flannery and David Simpson, Westat

Presenter: Todd Flannery

1. Introduction

Differences in modes like CAPI and CAWI present some unique challenges related to data collection and storage for a single instrument. For example, the web may require token authentication that is not needed in a secure disconnected tablet environment. In Blaise 5, we have tools that allow us to present question text and create unique routes that are based on either the mode or are perhaps device specific for the web. This paper describes several of these methods used to allow synchronized code updates between multiple modes by using prepared directives along with other specialized techniques to develop the code base.

For some projects, both the web and disconnected CAPI are used in order to collect instrument data from sampled respondents. To accomplish this, we have designed and developed an instrument in Blaise 5 that allows us to handle difference in the modes. In order to maintain comparability of the data, we need to ensure that the text of the questions and responses are consistent, but each mode has some unique considerations that also need to be addressed.

2. Some Initial Considerations

Time is not always on your side. We made the decision early in the design stage that our modes were different enough to forego trying to align the data model structures perfectly. To this end, the data and paradata can be separated into web or CAPI when they are being collected. Although this approach results in a greater amount of work in terms of the data alignment on the back end, it is less constraining on the data model design and development.

Web design requires more focus on accessibility controls, security, and maintaining mode-specific templates and layouts that can be used on a variety of devices and browsers. There are other CAPI-specific controls like touchscreen use, electronic signatures, or text-to-speech that may require specialized code like actions setups that may not be available via a client-server configuration. Additionally, since a disconnected CAPI instrument uses the Windows Data Entry Program, consideration needs to be given to developing layouts for a non-browser–based instrument. Although many differences in template design by mode can be effectively seen in the layout design, each mode (and browser) should be tested via data entry to detect errors in the layouts. Several elements of web versus CAPI design are depicted in Figure 1.

3. Code Methods—Use of Conditional Defines

Conditional defines (Figure 2) can be used to maintain separate project settings while having each project use the same files, and we can then allow the same text, fields, rules, or other Blaise controls to be shared among the projects. If we align the projects to specific modes, we can then update things like text and guarantee comparability over multiple modes. For our study, we have a mode (or project) for CAPI and several modes aligned with the different web servers (development, testing, production, etc.) When we build the BDIX files and associated objects like SQL tables or SQLite database files, they are associated with the mode that is defined for the specific project. This allows us to open the project to only view the routes, definitions, and layouts for the defined mode. For our project, the focus on maintaining consistent text translates into keeping the question texts and response texts outside of these conditions, whereas the

> back to Table of Contents

mode-specific text roles, procedures, blocks, fields, routes, or layouts may be exclusive to the mode in which the build is occurring (Figure 3).

Figure 1. Web Design and CAPI Design Elements

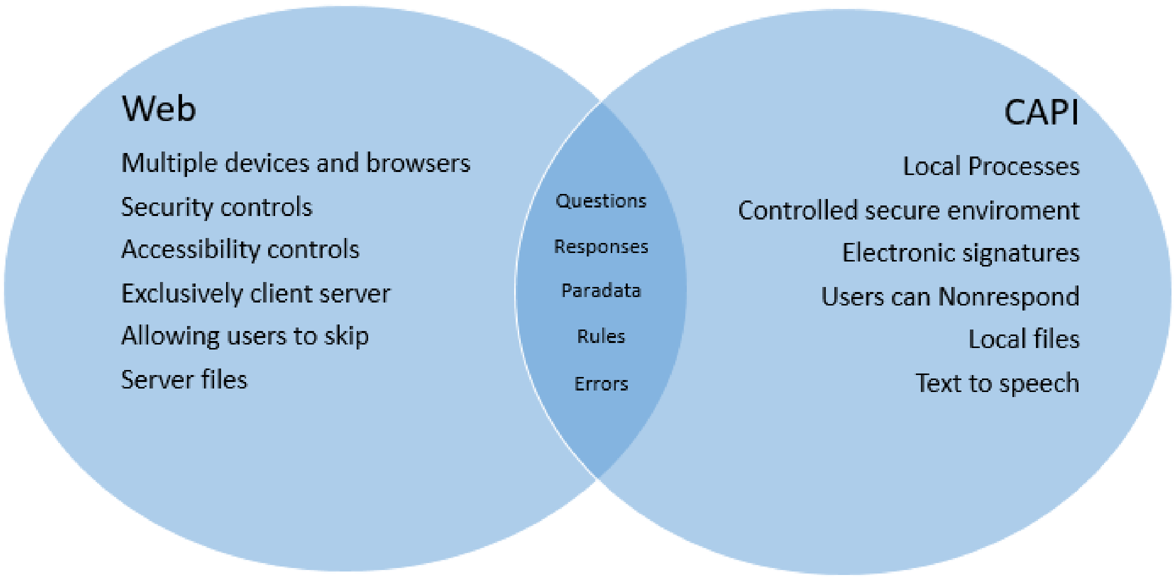
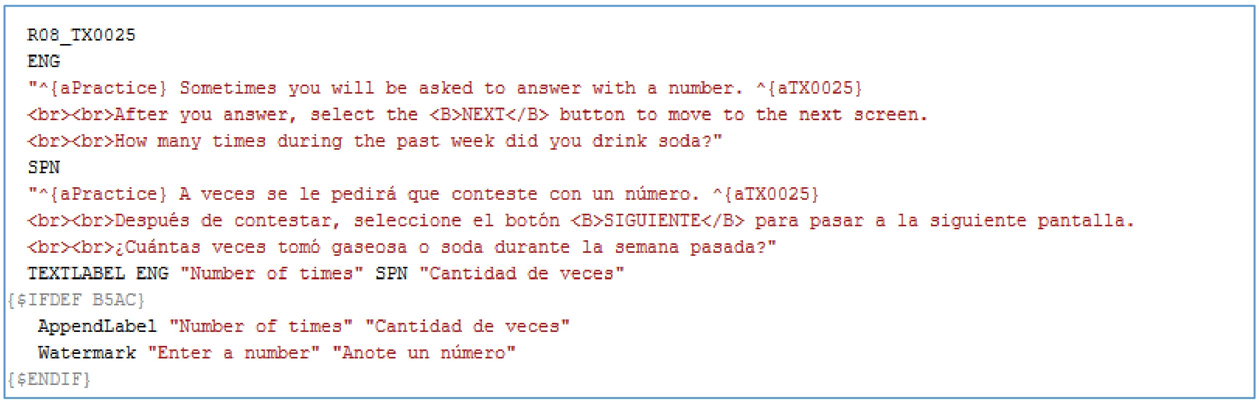


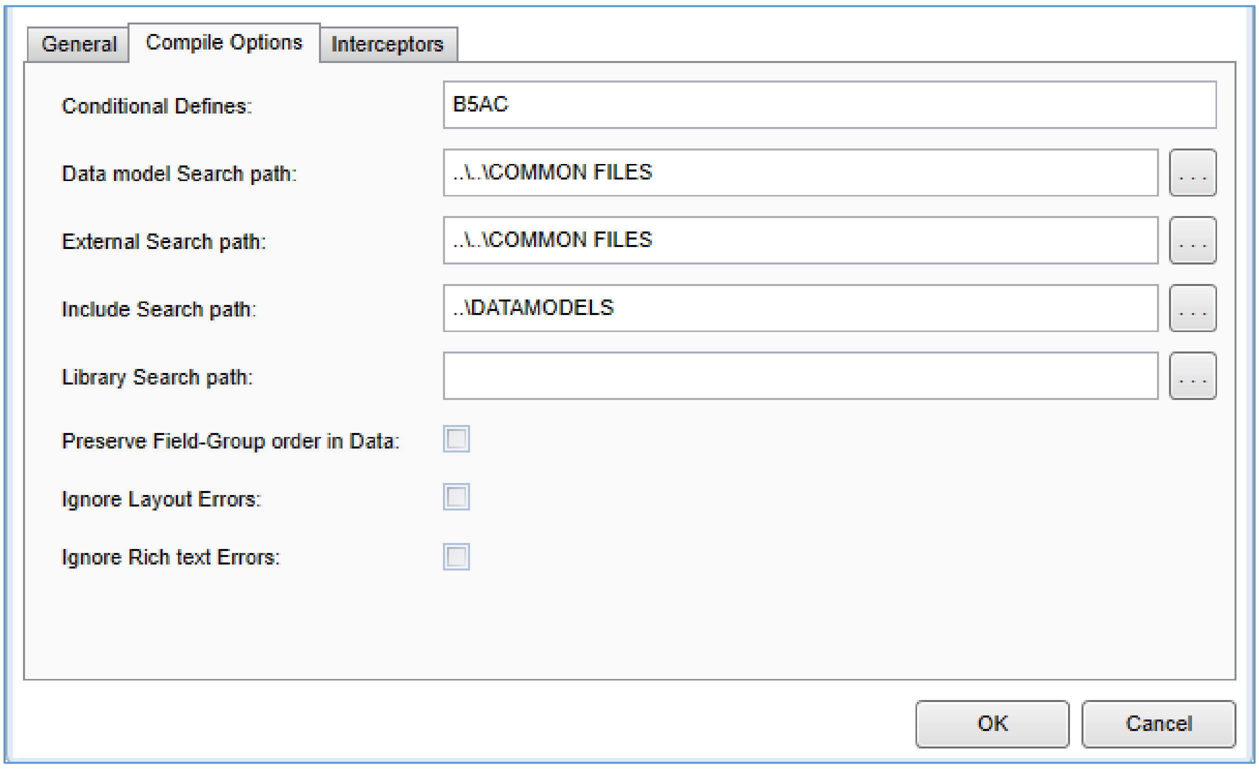
Figure 2. Use of Conditional Directive for a Non-Web Screen



```
  R08_TX0025
  ENG
  "^{aPractice} Sometimes you will be asked to answer with a number. ^{aTX0025}
  <br><br>After you answer, select the <B>NEXT</B> button to move to the next screen.
  <br><br>How many times during the past week did you drink soda?"
  SPN
  "^{aPractice} A veces se le pedirá que conteste con un número. ^{aTX0025}
  <br><br>Después de contestar, seleccione el botón <B>SIGUIENTE</B> para pasar a la siguiente pantalla.
  <br><br>¿Cuántas veces tomó gaseosa o soda durante la semana pasada?"
  TEXTLABEL ENG "Number of times" SPN "Cantidad de veces"
{$IFDEF B5AC}
   AppendLabel "Number of times" "Cantidad de veces"
   Watermark "Enter a number" "Anote un número"
{$ENDIF}
```

> back to Table of Contents

Figure 3. Definition for the Non-Web Build Mode of a Project



Conditional Defines:	B5AC
Data model Search path:	..\..\COMMON FILES
External Search path:	..\..\COMMON FILES
Include Search path:	..\DATAMODELS
Library Search path:	
Preserve Field-Group order in Data:	☐
Ignore Layout Errors:	☐
Ignore Rich text Errors:	☐

4. Web-Specific Considerations

For the web, some additional controls are required to be able to prevent security breaches into the study data or maintain compliance with accessibility standards (508, WCAG). Token authentication involves passing an expiring unique identifier (i.e., the token), like a GUID, into a Blaise instrument to be verified against an external data source before it expires. This ensures that access to the study data is only allowed from a known source, such as a landing page for a website. Since the source of the Blaise instrument is completely controlled in a disconnected mode, there is no requirement to token authenticate for CAPI, so we can exclude that section of code based on the mode. Blaise 5 help provides documentation for accessibility requirements settings and has the example "Visually Impaired" to demonstrate features associated with accessibility. Depending on project requirements, these controls may be mode-specific, so we can use the conditional defines to define our metadata based on these mode requirements.

5. CAPI-Specific Considerations

5.1 Action Setups

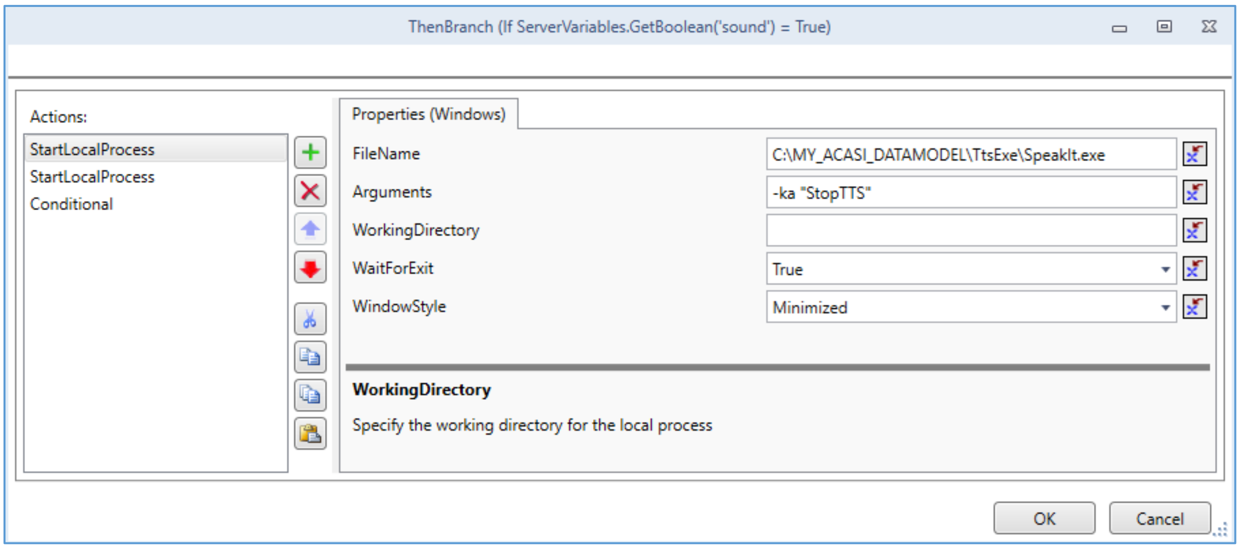
Blaise 5 allows use of one action setup per project to shell to either Manipula or non-Blaise processing, like an application to collect e-signatures. This method has been very useful in our project to allow some nonstandard Blaise behaviors like looping back to an earlier field on the route without first triggering an error message. If your user is a respondent, behaviors like these can be simpler to administer than training the user to react to error messaging. Since Blaise limits these action setups to thick client mode, some actions that are desired for the web need to use alternative methods to do this additional processing.

> back to Table of Contents

5.2 StartLocalProcess

We find the StartLocalProcess method via expressions in the resource database to be very useful in disconnected mode, as an alternative to using an action setup because the StartLocalProcess method, unlike Action Setup, does not require a page refresh (Figure 4). The action can call an external program to perform multiple actions. This in turn allows some processing to be done outside of conventional Blaise behaviors while a user may have scrolled down to the bottom of a page (a page refresh would by default show the top of the page).

Figure 4. Using StartLocalProcess in the Events Editor Instead of Action Setup to Call an External Process and Send Some Parameters



ThenBranch (If ServerVariables.GetBoolean('sound') = True)

Actions:
- StartLocalProcess
- StartLocalProcess
- Conditional

Properties (Windows)

FileName	C:\MY_ACASI_DATAMODEL\TtsExe\SpeakIt.exe
Arguments	-ka "StopTTS"
WorkingDirectory	
WaitForExit	True
WindowStyle	Minimized

WorkingDirectory

Specify the working directory for the local process

6. Both Web and CAPI Considerations

6.1 Texts

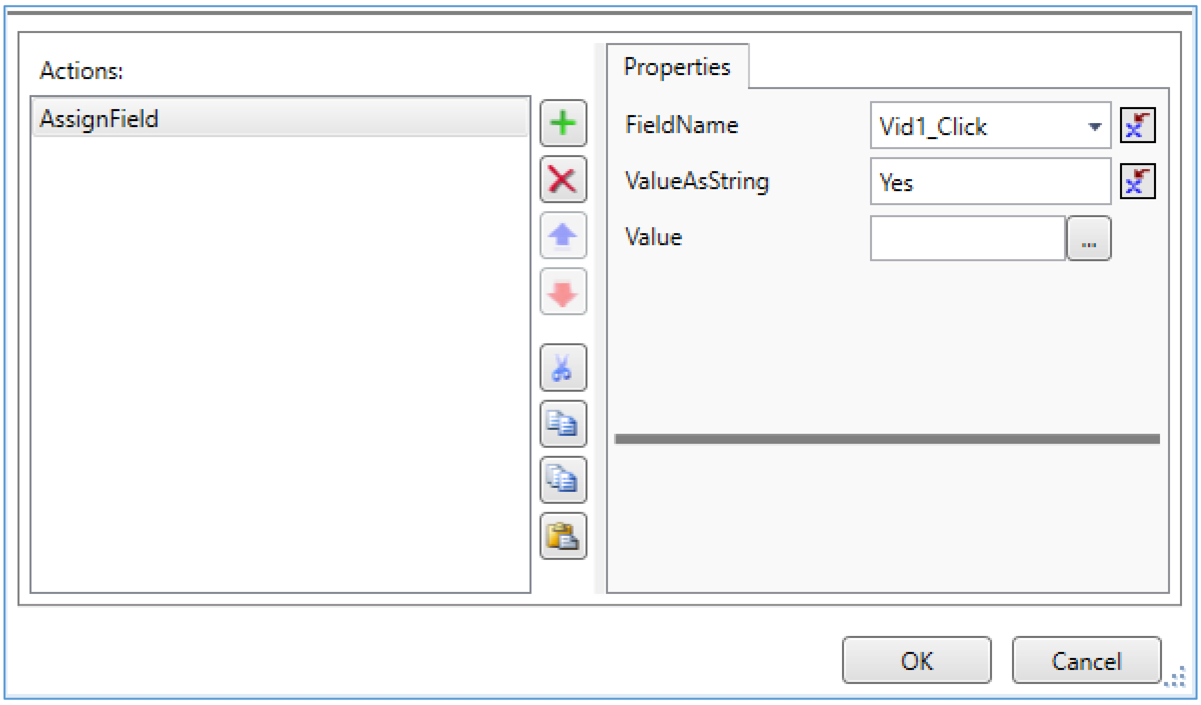
Since the content of the questions and responses is critical to the data quality, we need to ensure that any edits to these texts are simultaneously applied to both modes. The easiest way to satisfy this requirement is to have a single source for both modes. So, aside from mode-specific text (e.g., CAPI interviewer instructions), we share text across all modes for instrument questions and responses. Additional text roles that do not involve question or response text, like alt text for accessibility via the web or CAPI-interviewer error messages for CAPI mode, can be specific to the mode at build time.

6.2 Using Expressions and Assignments in the BLRD

To be able to capture user behaviors that trigger data assignments, we can use expressions or assignments in the events editor (Figure 5). This allows us to confirm whether an image or video was clicked on and can determine routes or error messages.

> back to Table of Contents

Figure 5. Assigning to Field Reference Based on a Click Event



Actions:

AssignField

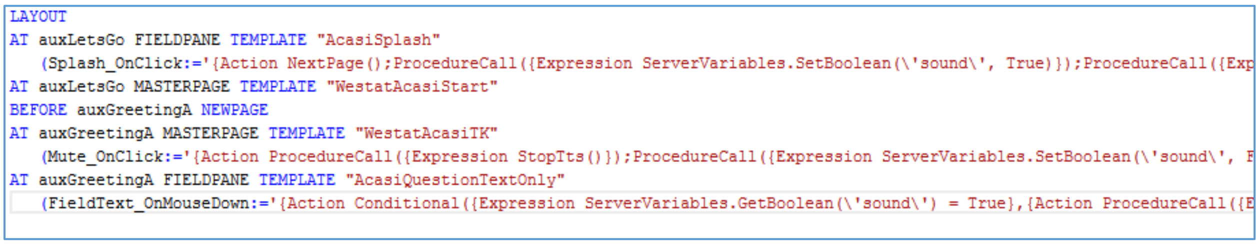
Properties

FieldName	Vid1_Click
ValueAsString	Yes
Value	

6.3 Use of Server Variables

Server variables can also be useful to allow assignments based on user behaviors, like clicking on a "Next" button instead of responding to something on the page or initiation of some ACASI events. These values can also be passed to procedures in an action setup for thick client configurations (Figure 6).

Figure 6. Using Server Variables to Store Boolean Values Used in Expressions.



```
LAYOUT
AT auxLetsGo FIELDPANE TEMPLATE "AcasiSplash"
   (Splash_OnClick:='{Action NextPage();ProcedureCall({Expression ServerVariables.SetBoolean(\'sound\', True)});ProcedureCall({Exp
AT auxLetsGo MASTERPAGE TEMPLATE "WestatAcasiStart"
BEFORE auxGreetingA NEWPAGE
AT auxGreetingA MASTERPAGE TEMPLATE "WestatAcasiTK"
   (Mute_OnClick:='{Action ProcedureCall({Expression StopTts()});ProcedureCall({Expression ServerVariables.SetBoolean(\'sound\', F
AT auxGreetingA FIELDPANE TEMPLATE "AcasiQuestionTextOnly"
   (FieldText_OnMouseDown:='{Action Conditional({Expression ServerVariables.GetBoolean(\'sound\') = True},{Action ProcedureCall({E
```

7. Data ALIGNMENT and Handling Field Updates

The desire to design and develop separate data structures that use the same texts leads to some additional complexities when transferring data between modes or merging data for delivery. For example, in order to maximize the response rate, some studies will allow a web user to skip past a field without responding, whereas CAPI mode requires the respondent to enter a nonresponse value of "don't know" or "refused" before progressing in the route. Since these definitions may contain different values, you must consider how to store, transfer, or deliver the data.

> back to Table of Contents

8. Results

Overall, results have been successful in the approach to use a single code base for text and have the metadata structures and layouts designed and developed based on the mode. Additional time is required to process and analyze test results, since the data are dependent on the mode. The experience has enriched our knowledge base of each mode and will inform future decisions regarding how to adapt to multimode requirements to more efficiently combine or separate the design and development for each mode.

9. Limitations of the Approach

One of the benefits or drawbacks of the approach is that conditional directives can appear anywhere in the code base, such as field definitions, rules, or layouts. So maintaining or discerning where modes differ between the built data models can be complex without strong source code management. Additionally, since the conditions themselves drive the build process but are not part of it, making sure that all of the "{$IFDEF}", "{$ELSE}", and "{$ENDIF}" statements can be burdensome. To assist with this task, it is helpful to know that an enhanced search of the number of $ENDIFs necessarily needs to be the sum of $IFDEFs and $IFNDEFs. To be sure, trying to track down missing conditional define statements can be difficult.

Given ample time for development and design, a single data model that allows data in either mode may be optimal, since the approach for using separate data structures by mode also requires significant commitment to merging data as a back-end task. So, given that design elements like layouts or web-based controls will need to be treated separately, consideration needs to be given as to whether the additional task of maintaining separate data is desired as well, especially if your project is intended to be fully multimode, such as when a respondent is given the opportunity to start in one mode, save and exit, and then finish in another mode.

10. Where to Go from Here

- Greater specificity of controls for the modes: Refining how to apply the conditional directives involves some trial and error to achieve optimal behavior and storage of the data in the data models. As we continue to learn more about how the design of web instruments improves our data collection capabilities, we can work to adapt how each mode can be made more efficient.
- As time allows, align the data. Ideally, one database shared among modes would allow for real-time multimode sharing of the data. Achieving this goal also eliminates the need for back-end processing to align the data, as well as the need to do multi-instrument synchronized releases. An intermediate step may be to have a server-based process to copy data from mode to mode, but this requires some lag time to handle the copying.
- Refinement of the processing: We apply many concepts from the exploration of multimode instrument development to include actions that occur outside of Blaise, enabling greater flexibility in our handling of on-screen behaviors and actions. Additionally, we can continue to apply some of the techniques like the use of server variables or adding events to enhance web processing to allow greater ease of use for users.
- We can continue to refine template design to allow the use of parameters in the templates themselves in order to have a consistent appearance across modes wherein a single design change can be applied to templates that are used in either mode.

> back to Table of Contents

11. Conclusions

Design and development for multimode instrumentation requires significant planning in determining which approach regarding database structure, code base, resources, and layouts is exclusive to each mode. Given substantial time to plan this work, a single database and codebase could be achievable, but complex studies rarely provide for this benefit. To be able to collect high-quality data via the web and disconnected modes, the approach to utilize these techniques of retaining a single code base for text and allowing for differences in data structure and layouts has proven beneficial in circumstances where complete harmonization of the data between modes is not possible in a single structure.

> back to Table of Contents

Advanced Editing: Integrating Blaise with a Management System

Peter Stegehuis and Seth Benson-Flannery, Westat

Presenter: Peter Stegehuis

1. Introduction

During the stage of cleaning interview data, it is good practice to use the same Blaise datamodel as used in the field for reasons of consistency and efficiency. This paper will touch on the challenges presented by special demands during this phase. Some of the challenges are in the Blaise datamodel and the data itself, for instance, the need for additional checks and being able to reach "behind walls" set during the field interview. We also look at how best to present data issues to staff in an easy-to-use user interface, how to automatically categorize interviewer comments (Blaise remarks), and how to integrate this editing system within a comprehensive Home Office System.

2. What Is There to Clean?

2.1 A Different Focus

During a field interview, especially for a longer and more complex survey, the focus is, of course, on getting high-quality responses, but also on ensuring the continued engagement of the respondent. To keep the interview going, and not frustrate respondents or give them an easy opportunity to stop their cooperation, interviewers and the (Blaise) interview program need to work well without significant pauses. It puts emphasis on good interviewer training on the one hand, and on a well-designed and well-functioning instrument on the other.

For the design, that could mean choosing to not add too many checks, balancing the need for the highest data quality with the need to not slow down the interview or frustrate the respondent to the point where they stop their cooperation. For the CAPI instrument, it means things have to work well and flow well, without glitches or long pauses.

This means that during the cleaning phase, some additional Blaise soft checks may be applied to scrutinize situations that purposefully were not flagged with the respondent during the interview.

In very complex instruments, there may be many places where variables get assigned a value behind the scenes, meaning not in rules that will get reevaluated, which should be undone in case the interviewer backs up and changes answers in a way that leads to a different route through the questionnaire. This cleaning up can get very complicated, and it may be a realistic option to do this cleaning up after completion of the interview, in the data cleaning phase, especially if it is a rare scenario and the data that is not being cleaned up has no impact on the remainder of the interview with the respondent.

These are a few examples of additional checks that can be applied during data cleaning, either by adding checks in the Blaise rules that are only active during the data cleaning phase or by running separate Manipula setups on the Blaise data when loading the case into the data cleaning system and creating new data cleaning issues for the case when needed.

2.2 Categories of Issues

In the data cleaning phase, the focus shifts to fixing specific issues. Issues to be looked at during the data cleaning phase may roughly be divided into four categories:

a) Interviewer comments/Blaise remarks

> back to Table of Contents

b) Known issues, like additional checks in the cleaning stage
c) New problems with the fielded CAPI questionnaire
d) Issues coming from the HelpDesk, reported by field interviewers after (partially) completing an interview
e) Issues discovered during the data cleaning phase by a Data Quality Control (DQC) Data Technician

Interviewer comments can be very useful, but at the same time they are very time consuming to handle, so we have developed some special ways of dealing with them. During data collection, instead of the standard Blaise remark screen, we bring up a Manipula dialog that asks for a category before the interviewer can make the comment itself, and then when the category is known, we remind the interviewer what specific information to include. This was described in a separate 2018 IBUC paper (Stegehuis, 2018).

We store the comment in a separate Blaise data file—with the category—but still use the Blaise remark paperclip for visibility of and easy access to the comment.

When the interview data gets received at Home Office, our DQC process will send these comments through a Natural Language Process to parse the comment itself and—separate from the category assigned by the interviewer—assign the most likely three categories for the comment. These categories are stored with the comment in the DQC issue that gets created and will be visible during data cleaning, so it is easier to assign the right specialist Data Technician to deal with the issue.

For category b), we have a folder where we have Manipula setups ready to run against all incoming data and run the necessary checks. Each issue that gets flagged during this process gets turned into its own data cleaning item in the SQL Server database that will guide the DQC stage.

In case any problems are found in the data, for instance, based on newly found CAPI program issues, as in category c) above, new Manipula setups can be added to the same folder. The overall system is set up to execute all Manipula setups in the folder, and flexibility lets us check all incoming cases automatically. Any discovered issues from these setups will be recorded in the DQC SQL Server database. This flexibility allows us to react very quickly to any new problems that may be found, even during the field period itself.

HelpDesk reports and issues discovered by a DQC Data Technician, categories d) and e) above, may be turned into DQC issues as well, so they can be addressed quickly.

3. Datamodel Changes

As mentioned before, one of the strengths of Blaise is that the same code can be used both for the original field interview and for the data cleaning stage. The two big advantages of that approach to cleaning are:

- Any changes made during the data cleaning stage will undergo the same routing and rules checking as the original interview, ensuring that no new data problems are introduced during the cleaning stage.
- Re-use of the Blaise code for this stage is a big cost saver compared to creating and maintaining a separate code base that uses different software, especially for complex surveys and for panel surveys.

> back to Table of Contents

However, there are differences in how we'd want to work with the Blaise application based on the difference in what the main goal is: completing an interview from start to finish versus fixing one issue, or maybe a few issues, in that entire case.

So, we do want to have one code base but also have slightly different behavior in field interviews than when used at the Home Office for data cleaning purposes. There are different ways to achieve that, but the way we have implemented this is by assigning a value to a datamodel-level auxfield on the command line when a data cleaning "interview" session gets started.

We will highlight just a few behavior changes that this use of the command line parameter enables.

3.1 Walls

The first main section of our questionnaire establishes the household composition, and at the end of the section we know, based on preload and the answered questions, who will be part of the remainder of the questionnaire. At the end of this section, we have programmatically erected a "wall," so that interviewers cannot back up after proceeding past it. This is a common strategy to ensure that the data collected in the main interview will not be invalidated by an interviewer backing up and changing the household composition later on.

In the data cleaning phase, however, it may be important to have access to those first questions, for instance, to change a typo in a person's name or correct a birth date or age. Determining whether a section gets an "ASK" or a "KEEP" in Blaise language based on an auxfield value is easy enough.

The challenge here is what needs to happen at the end of that first section, the code in the wall, if you will. In the field interview, this is the spot where the person roster, plus all that needs to be carried over from any preload data, gets put into place. During the cleaning phase, we do not want to overwrite that person roster (for the same reason we don't allow backing up during the field interview), so we have to check at the wall whether any changes were made during the data cleaning session that would have a negative impact. This is complicated code but is needed to ensure the quality of the data. If all is fine, the DQC Data Technician can proceed; otherwise, a warning will be put up detailing the problematic circumstance that has arisen and a decision needs to be made by project staff about what needs to be done.

3.2 Skipping a Section

An easier change in instrument routing during the data cleaning phase occurs when a section can be skipped as a whole, because it is outside the bounds of DQC staff to make changes there. An example of this is when we ask household members for signatures or to complete additional questionnaires, either online or on paper. This is only relevant during the field interview, and we do not need nor want DQC staff to ever make any changes here, so we use the auxfield parameter value in the datamodel rules to skip such sections.

4. Data Flow

A somewhat simplified step-by-step description of the data flow and the place for data cleaning within it:

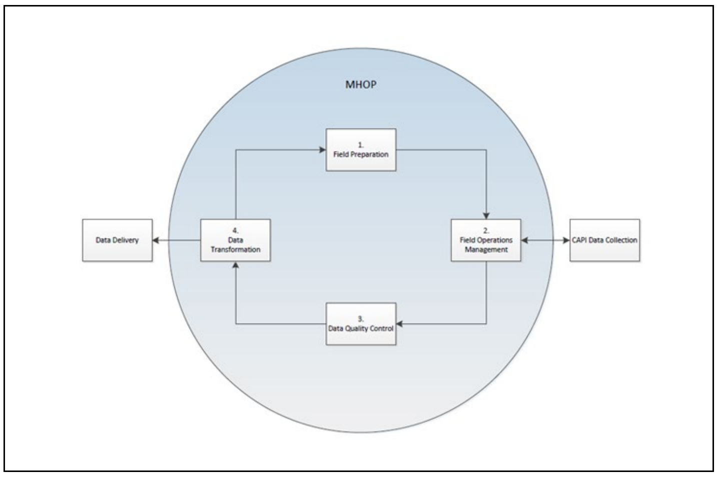
- Data transmissions with completed interviews come in from the field
- Daemon automatically picks up those transmissions and divides parts of the data to deposit them in the right place on the network (e.g. management data, CARI data, Blaise interview data)
- Load Blaise data for new cases into a consolidated database to always have the Blaise data exactly as it was before any data cleaning

> back to Table of Contents

- Run any interviewer comments (Blaise remarks) through a special process to determine the most likely category for each of them, making it easier for work distribution later
- Run any necessary checks on cases with Manipula setups

- Log any issues, either from these checks or from interviewer comments into the SQL Server database that is the backbone of our DQC (the "cleaning stage")

Figure 1. High-Level Overview of Data Flow for Our Survey



Of course, some cases will have issues that need to be cleaned, or at least looked at, and others will not need any additional scrutiny at this stage. In our experience, approximately three-quarters of cases sail through, meaning they have no interviewer comments and no issues from additional checks or from HelpDesk tickets.

Data Technicians work on the issues for the cases that do need attention using the Blaise CAPI instrument, with a special command-line parameter that is used to alter the DEP behavior for this stage where needed:

```
IF auxEditModeToggle = Yes THEN
  auxSlashQ:= auxSlashQ + 'EditMode=Yes;'
ENDIF
auxCommandLineString := auxCommandLineString + auxSlashQ + ' /X'
AuxRslt := EDIT(auxCommandLineString)
```

A straightforward skip of a section in this EditMode, or cleaning mode, would simply become:

```
IF EditMode <> Yes THEN
    RF_Main
ELSE
    RF_Main.KEEP
ENDIF
```

Note that we are using an auxfield for this designation. The value does not get saved at the end of the session, so that the desired behavior—interviewing or cleaning—gets set for just the current session and not for all future sessions.

5. Home Office System Integration

On interviewer laptops, we have an Interviewer Management System (IMS) that shows the field interviewers their assigned cases and tasks, one of which is starting the Blaise interview. When data gets transmitted back to the Home Office, parts of the data package go in different directions: Blaise CARI data goes one way, management data from the IMS goes to a central SQL Server database, and the Blaise CAPI data will go yet another way—to be loaded overnight into our DQC system.

> back to Table of Contents

Just like the IMS is controlling the field interviewers' options and tasks, the Home Office System is controlled by the management data in SQL Server. Subprocesses, like DQC, have their own SQL Server tables and statuses for tasks, but the overall control lies with the central system. This includes the overall status for each case, as well as status for additional items like separately collected (online or paper) forms.

The DQC system runs its own tasks and also checks the consistency of cases and status within the larger system.

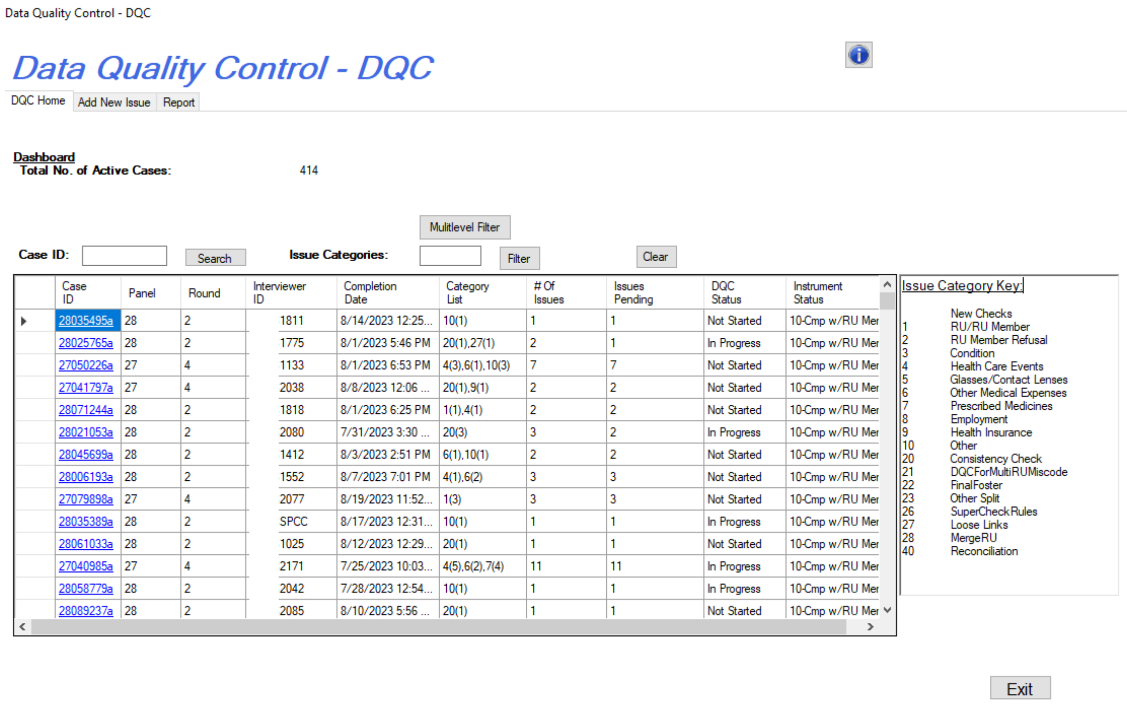
6. The User Interface

The DQC Data Technicians and DQC Supervisors start their work using our C# application that serves as the graphical user interface. It shows their workload based on role, and for each assigned issue, they can see the status overview plus a detail screen with the history of the issue and if needed, they can start the Blaise instrument from there.

After a Blaise data entry session, we run the same Manipula setups that were executed during the original loading of the case to ensure no new issues were introduced. They can close out issues and when a case has no more issues left can clear it for further processing.

When the user first starts the C# program, they will see the cases available to them.

Figure 2. DQC Overview Screen



Dashboard
Total No. of Active Cases: 414

Case ID:
Issue Categories:

Case ID	Panel	Round	Interviewer ID	Completion Date	Category List	# Of Issues	Issues Pending	DQC Status	Instrument Status
28035495a	28	2	1811	8/14/2023 12:25...	10(1)	1	1	Not Started	10-Cmp w/RU Mer
28025765a	28	2	1775	8/1/2023 5:46 PM	20(1),27(1)	2	1	In Progress	10-Cmp w/RU Mer
27050226a	27	4	1133	8/1/2023 6:53 PM	4(3),6(1),10(3)	7	7	Not Started	10-Cmp w/RU Mer
27041797a	27	4	2038	8/8/2023 12:06 ...	20(1),9(1)	2	2	Not Started	10-Cmp w/RU Mer
28071244a	28	2	1818	8/1/2023 6:25 PM	1(1),4(1)	2	2	Not Started	10-Cmp w/RU Mer
28021053a	28	2	2080	7/31/2023 3:30 ...	20(3)	3	2	In Progress	10-Cmp w/RU Mer
28045699a	28	2	1412	8/3/2023 2:51 PM	6(1),10(1)	2	2	Not Started	10-Cmp w/RU Mer
28006193a	28	2	1552	8/7/2023 7:01 PM	4(1),6(2)	3	3	Not Started	10-Cmp w/RU Mer
27079898a	27	4	2077	8/19/2023 11:52...	1(3)	3	3	Not Started	10-Cmp w/RU Mer
28035389a	28	2	SPCC	8/17/2023 12:31...	10(1)	1	1	In Progress	10-Cmp w/RU Mer
28061033a	28	2	1025	8/12/2023 12:29...	20(1)	1	1	Not Started	10-Cmp w/RU Mer
27040985a	27	4	2171	7/25/2023 10:03...	4(5),6(2),7(4)	11	11	In Progress	10-Cmp w/RU Mer
28058779a	28	2	2042	7/28/2023 12:54...	10(1)	1	1	In Progress	10-Cmp w/RU Mer
28089237a	28	2	2085	8/10/2023 5:56 ...	20(1)	1	1	Not Started	10-Cmp w/RU Mer

Issue Category Key:

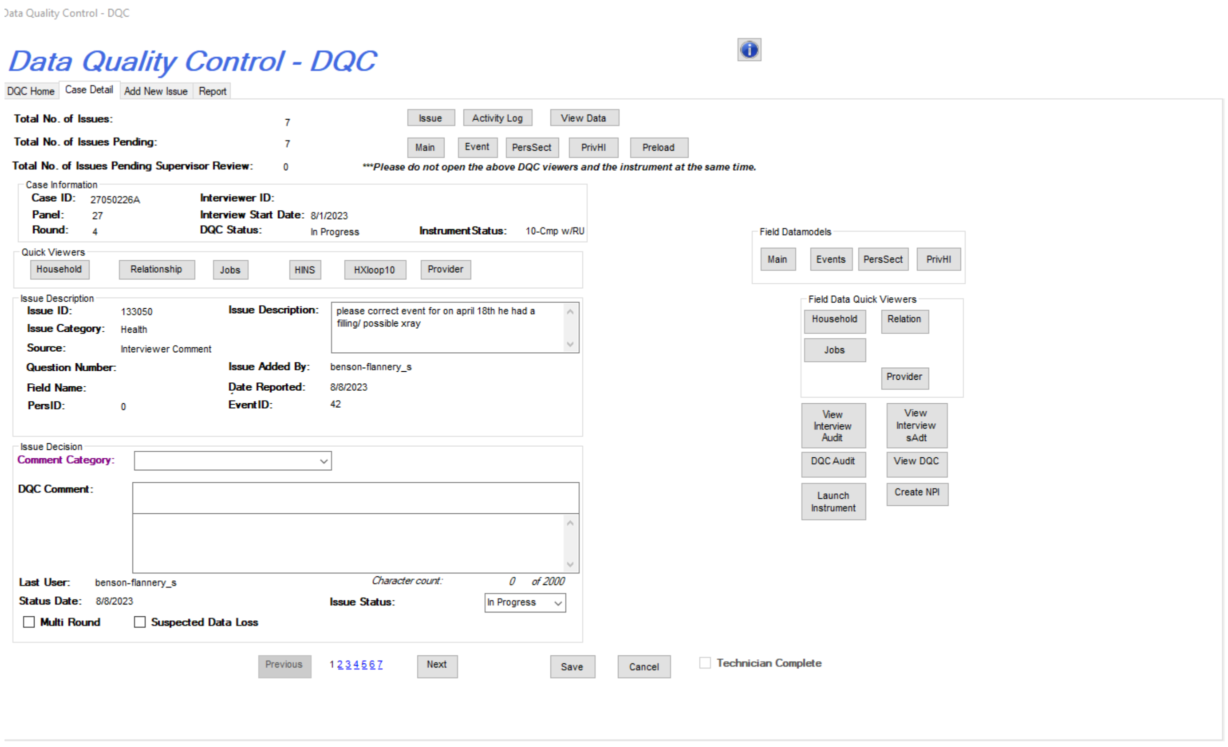
	New Checks
1	RU/RU Member
2	RU Member Refusal
3	Condition
4	Health Care Events
5	Glasses/Contact Lenses
6	Other Medical Expenses
7	Prescribed Medicines
8	Employment
9	Health Insurance
10	Other
20	Consistency Check
21	DQCForMultiRUMiscode
22	FinalFoster
23	Other Split
26	SuperCheckRules
27	Loose Links
28	MergeRU
40	Reconciliation

After selection of the case, the user will be taken to the Case Details Screen, as seen below. As you can see, this page allows the user to see any information we have about this specific case. In addition to the data provided on the page, the user can launch the Blaise instrument via a button click. The user may also select a data viewer to see the case data in its entirety or a predetermined subset of the data.

> back to Table of Contents

This screen is also where the user will enter any information about the case they feel is relevant to the issue. They can also record that all work on the case has been completed.

Figure 3. DQC Case Detail Screen



Data Quality Control - DQC

Data Quality Control - DQC

DQC Home | Case Detail | Add New Issue | Report

Total No. of Issues: 7

Total No. of Issues Pending: 7

Total No. of Issues Pending Supervisor Review: 0

****Please do not open the above DQC viewers and the instrument at the same time.*

Case Information

Case ID: 27050226A **Interviewer ID:**
Panel: 27 **Interview Start Date:** 8/1/2023
Round: 4 **DQC Status:** In Progress **InstrumentStatus:** 10-Cmp w/RU

Issue Description

Issue ID: 133050 **Issue Description:** please correct event for on april 18th he had a filling/ possible xray
Issue Category: Health
Source: Interviewer Comment
Question Number: **Issue Added By:** benson-flannery_s
Field Name: **Date Reported:** 8/8/2023
PersID: 0 **EventID:** 42

Issue Decision

Comment Category:

DQC Comment:

Last User: benson-flannery_s *Character count: 0 of 2000*
Status Date: 8/8/2023 **Issue Status:** In Progress

☐ **Multi Round** ☐ **Suspected Data Loss**

☐ Technician Complete

7. Conclusions

The use of Blaise for data cleaning as well as the original field interviewing can lead to a very powerful and efficient system, fitting seamlessly between fielding of cases and following tasks like data delivery and creating preload data for the next round of field interviewing.

Using scheduled automatic processes on (virtual) daemon machines, we can have a steady stream of new cases to work for our DQC Data Technicians and create a very efficient process using the same Blaise code for field interviewing and data cleaning.

8. References

Stegehuis, P., & Westat. (2018, October 22). *A different approach to Blaise remarks*. 18th International Blaise Users Conference, Baltimore.

> back to Table of Contents

Using Paradata to Evaluate the Effect of Changes in the Small Screen Layout

Elise Alstad and Erdal Kilicdogan, Statistics Norway

1. Abstract

With Blaise 5 multimode capabilities, we can create dynamic and adjustable layouts for different screens—including layout for small screens—which is important because more respondents are using smart phones to respond to our surveys. Understanding the best design practices for small screens becomes essential as we aim to improve our surveys on mobile and to use a mobile-first approach when designing new surveys. A significant challenge when adapting web surveys initially designed for pen and paper or PC is to adjust questions in a group that are presented in a grid table to small screens. As grid tables are unsuitable for small screens, we usually change the presentation of the group questions to item-by-item questions on small screens. The Quality of Life Survey (QLS) is a web survey that uses grid tables to present question groups on PC, and between 2022 and 2023, we attempted to improve the small screen layout for these question groups. We transitioned from a pagination design, where each question is presented on a single page, to a scrolling design, where several questions are on the same page. Hoping to have reduced respondent burden and improved user experience by transitioning from pagination to scrolling, we will assess this hypothesis by comparing paradata indicators of QLS 2022 and 2023. We have focused on the following indicators in our analysis: (1) previous page actions, (2) response times, (3) breakoff rates, and (4) route errors. We find lower rates of previous page clicks, route errors, and breakoffs among mobile respondents in the 2023 QLS compared to the 2022 QLS. These results could indicate that the scrolling layout might be more suitable for small screens than the pagination layout. However, we see that we need to do controlled A/B testing of the layout to arrive at a solid conclusion, and we discuss how we aim to conduct these tests in the future.

2. Introduction

At Statistics Norway, we see that a majority of web responses are completed on small screens in various web surveys for individual persons (Pettersen & Engvik, 2022; Holmøy & Rossbach, 2021). Consequently, we aim to have a mobile-first approach when designing new surveys and to enhance our current web surveys for mobile use. Mobile phones may introduce other sources of survey error, which are not as prominent on PC. For instance, on small screens, response burden may increase if respondents have to scroll and zoom to see the questions, or if respondents have to handle small slider scales or radio items to select their responses. Therefore, it is important that the questionnaire and the layout are designed such that it is easy for the respondent to respond to the questions on small screens.

Our team has experienced that adapting our web surveys to small screens can be challenging if the web surveys use several large grid tables. Grid table is a format widely used to present a group of questions with the same answer categories, as it is an efficient use of space and text. But grids tables are unsuitable for small screens because they take up too much space, requiring the user to zoom and scroll. Additionally, the radio buttons in the grid tables are too small for small screens to comply with accessibility standards. It has been recommended to transform grid questions to item-by-item questions for mobile (Vehovar et al., 2022), but also on PC, to ensure comparability between mobile and PC respondents (Revilla et al., 2017). At Statistics Norway, we have, for most web surveys, adjusted the format for grid questions into item-by-item questions on small screens.

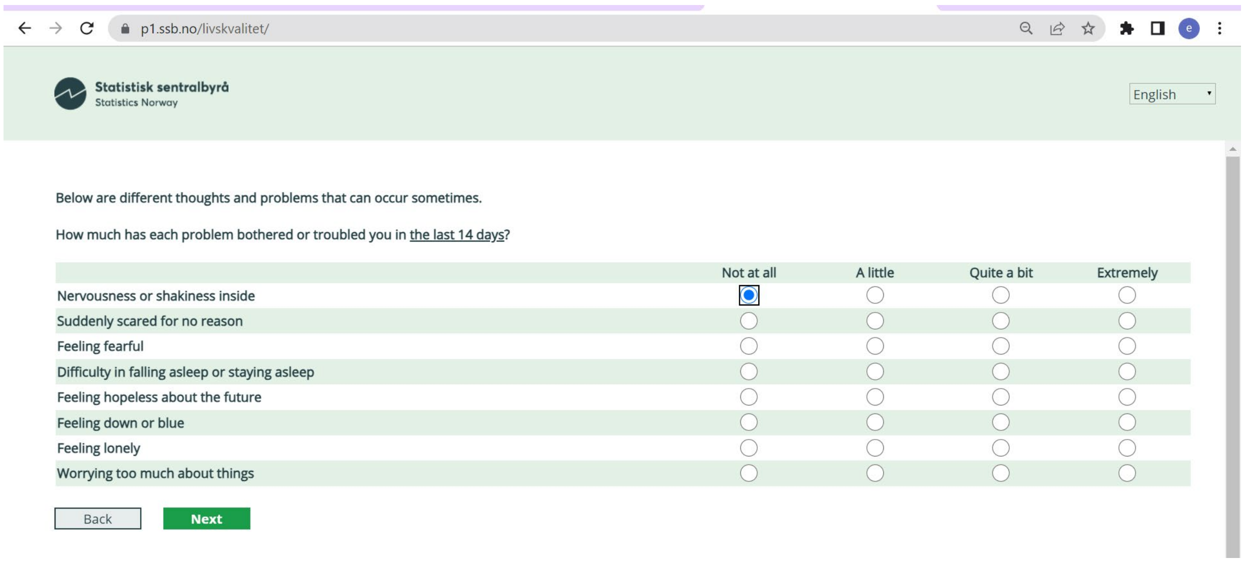
The Quality of Life Survey (QLS), which has been conducted yearly since 2020, uses several grid tables. Out of 175 question fields in the QLS survey questionnaire, 55 question fields are presented using grid tables in the PC layout. There are a total of 9 grid tables for 9 question groups in the PC layout of the

> back to Table of Contents

survey. These question fields are in groups, but they are presented as item-by-item questions and not in a grid table in the small screen layout, and we will therefore, for the sake of ease, refer to these specific questions of the QLS questionnaire as the “question groups” in this article. Between 2022 and 2023, we attempted to improve the small screen layout of the question groups by changing the presentation from a pagination design to a scrolling design. We utilized paradata from the Blaise audit trail to evaluate the effect of changing the layout.

Prior to the 2023 QLS, we assessed how we could improve the small screen layout of the question groups. Figure 1 shows a question group presented in a grid table in the large screen layout. The same question group in the small screen layout was presented in an item-by-item format with a pagination design and one item per page, as shown in Figures 2, 3, and 4. We suspected that the pagination layout might be unideal for the question groups because the introductory group question text was only visible on the page with the first question in the group. If respondents forget the group question text in the middle of the question group, it could be a substantial response burden to go back through previous pages to see the group question text. We therefore evaluated how we could improve the presentation of the question groups in the small screen layout.

Figure 1. Screenshot of Quality of Life Survey 2022 and 2023, Chrome Browser



Below are different thoughts and problems that can occur sometimes.

How much has each problem bothered or troubled you in the last 14 days?

	Not at all	A little	Quite a bit	Extremely
Nervousness or shakiness inside	◉	○	○	○
Suddenly scared for no reason	○	○	○	○
Feeling fearful	○	○	○	○
Difficulty in falling asleep or staying asleep	○	○	○	○
Feeling hopeless about the future	○	○	○	○
Feeling down or blue	○	○	○	○
Feeling lonely	○	○	○	○
Worrying too much about things	○	○	○	○

We did a review of the literature to explore mobile alternatives for grid questions. A common mobile alternative to the grid table presentation is item-by-item in a scrolling layout, where all the group questions are presented on the same page. De Bruijne and Wijnant (2014) compared different alternative layouts for grid questions and saw a shorter completion time for a scrolling layout compared to a pagination layout. Similarly, Mavletova and Couper (2014) found that a scrolling layout seemed to be more suitable than pagination when assessing breakoff rates, reports of technical problems, and respondent rating of the questionnaire.

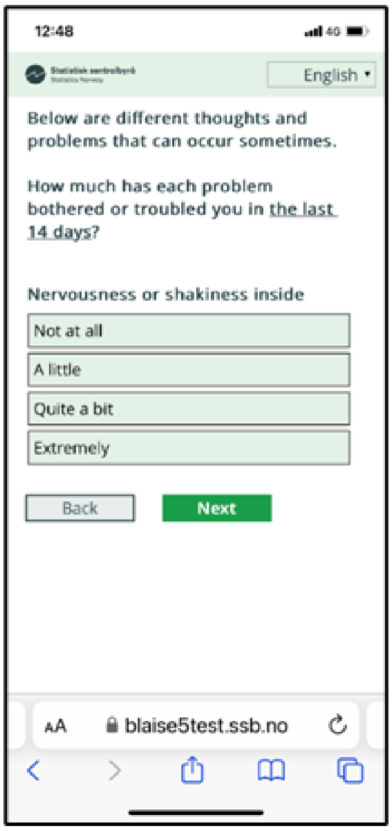
Additionally, we tested scrolling, pagination, and other layout alternatives within the team. Most of the team members were positive towards changing the small screen layout to a scrolling presentation, as they found it easier and more preferable to navigate by scrolling rather than having to navigate by using the “Next” and “Previous” page buttons. Thus, we decided to adjust the small screen layout of the question groups from pagination to a scrolling layout in the 2023 QLS (see Figures 5, 6, and 7). As emphasized by Cheung et al. (2016), the layout of a survey can substantially influence how respondents perceive and

> back to Table of Contents

respond to the questionnaire. To assess if the scrolling layout is more suitable for small screens than the pagination layout, we assess four indicators that aim to measure response burden using paradata from the Blaise audit trail; namely, we compare the level of (1) previous page actions, (2) response times, (3) breakoff rates, and (4) route errors received between 2022 and 2023 to see if adjusting the small screen layout could reduce response burden and improve user experience on mobile.

Figures 2–4 show the Quality of Life Survey 2022 in the iPhone 11 Safari browser.

Figure 2. View When Entering the Page for the First Question, Which Includes the Group Question Text



Below are different thoughts and problems that can occur sometimes.

How much has each problem bothered or troubled you in the last 14 days?

Nervousness or shakiness inside

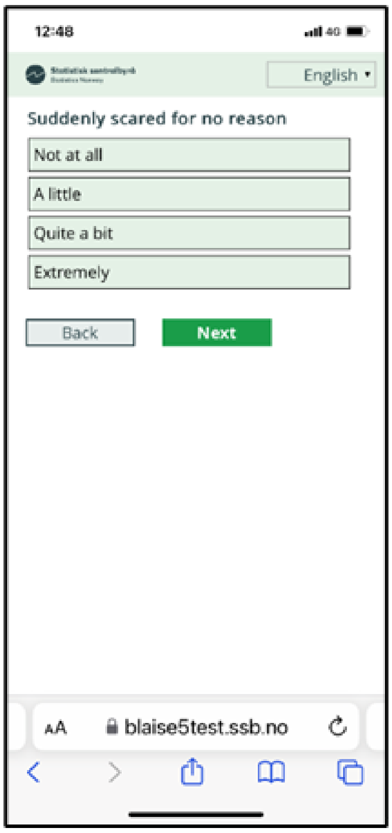
Not at all

A little

Quite a bit

Extremely

Figure 3. View of the Next Page



Suddenly scared for no reason

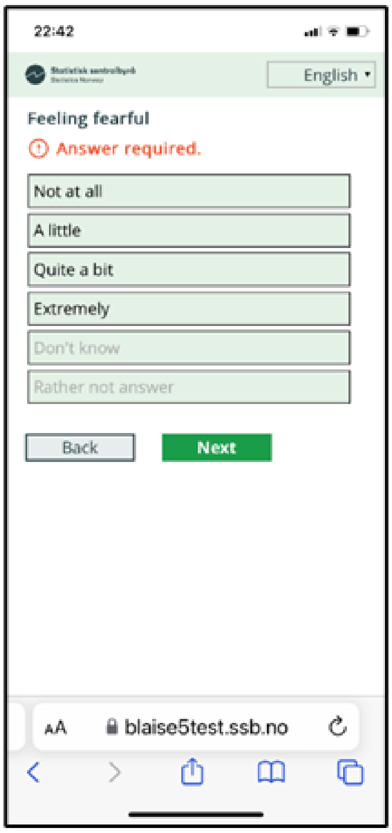
Not at all

A little

Quite a bit

Extremely

Figure 4. View of Third Group Question and the Route Error That Arises When the Respondent Attempts to Move to the Next Page without Responding to the Question



Feeling fearful

Answer required.

Not at all

A little

Quite a bit

Extremely

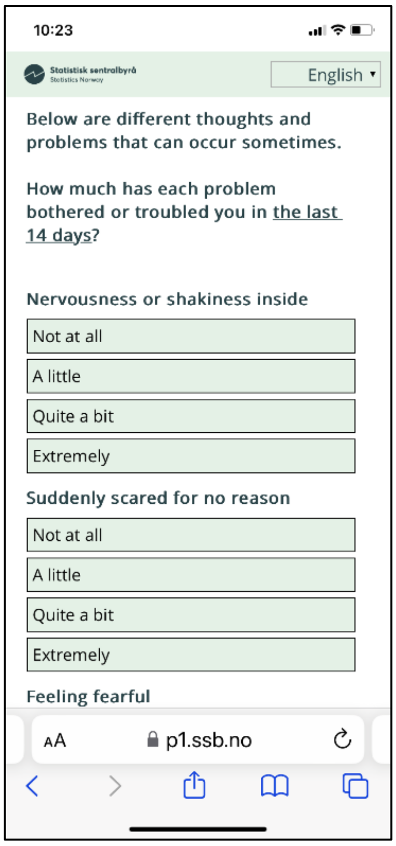
Don't know

Rather not answer

> back to Table of Contents

Figures 5–7 show the Quality of Life Survey 2023 in the iPhone 11 Safari browser.

Figure 5. Immediate View When Entering the Page


10:23

Statistisk sentralbyrå
Statistics Norway

English

Below are different thoughts and problems that can occur sometimes.

How much has each problem bothered or troubled you in the last 14 days?

Nervousness or shakiness inside

- Not at all
- A little
- Quite a bit
- Extremely

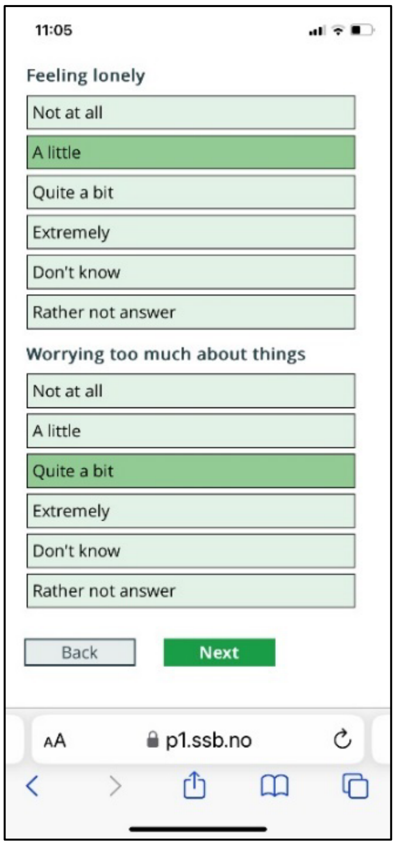
Suddenly scared for no reason

- Not at all
- A little
- Quite a bit
- Extremely

Feeling fearful

AA p1.ssb.no

Figure 6. View in the Middle of the Page When Two Answers Are Selected


11:05

Feeling lonely

- Not at all
- [✓ A little]
- Quite a bit
- Extremely
- Don't know
- Rather not answer

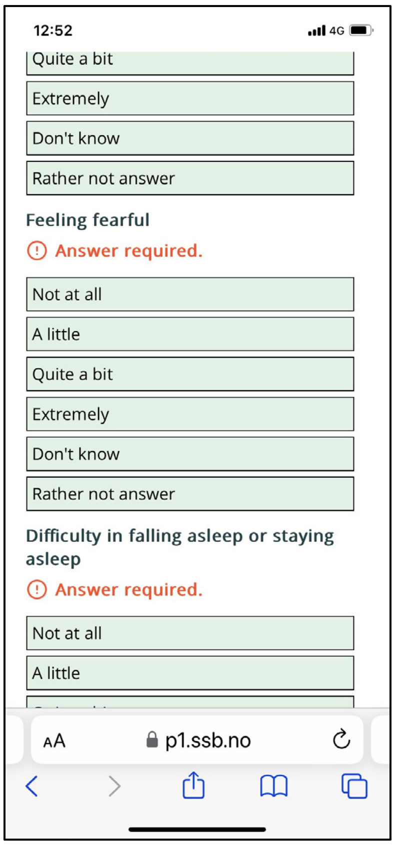
Worrying too much about things

- Not at all
- A little
- [✓ Quite a bit]
- Extremely
- Don't know
- Rather not answer

Back Next

AA p1.ssb.no

Figure 7. View in the middle of the Page When Two Unanswered Questions Trigger a Route Error


12:52

- Quite a bit
- Extremely
- Don't know
- Rather not answer

Feeling fearful

Answer required.

- Not at all
- A little
- Quite a bit
- Extremely
- Don't know
- Rather not answer

Difficulty in falling asleep or staying asleep

Answer required.

- Not at all
- A little

AA p1.ssb.no

3. Paradata Indicators

Paradata from the Blaise audit trail can provide useful insights into how respondents move through the questionnaire and can be used to identify issues respondents experience. Paradata from the audit trail show the respondents' actions in the survey and the timestamp for each action. Thus, by using paradata, we can gain a greater understanding in areas of concern and learn how to improve survey questionnaires, potentially resulting in higher response rates, more accurate responses, and enhanced user experience. In this article, we use paradata to assess specific indicators that aim to measure response burden. We will look at (1) previous page actions, (2) response time, (3) breakoff rates, and (4) route errors. By assessing these indicators on activity within the question groups with the changed layout, we evaluate if changing from a pagination layout to a scrolling layout can influence how the respondent interacts with the survey and if it can help to reduce the response burden.

3.1 Previous Page

Previous page can be an especially useful indicator, as it can illustrate issues with the flow of the survey and identify problems with survey questions. Qualitative user tests at Statistics Norway have found that respondents go to the previous page for various reasons; namely, if the question is perceived as too

> back to Table of Contents

similar to the previous one, the respondent may go back again to the previous question to understand the difference between the current and the previous question. Additionally, respondents go to the previous page if they recognize, when presented with a question, that the previous question on another page was answered incorrectly. Importantly, respondents may click to the previous page because the current question is missing some information, and the respondent returns to the previous page to retrieve the missing information.

In the pagination layout, the group question text is shown only on the page with the first field, and the following group fields are missing important information. This may cause the respondent to return to the previous page, resulting in a higher number of previous page clicks for respondents in the pagination layout than in the scrolling layout. We can find the previous page actions in the Actions variable in the audit trail data. By counting the number of "PreviousPage" actions within the "Action" variable for each respondent, we can find the average number of times respondents clicked previous page within the question groups.

Hypothesis 1: Respondents have a higher frequency of previous page actions in the pagination layout (2022) than in the scrolling layout (2023).

3.2 Response Time

Response time is frequently used to assess respondent behavior and to infer results regarding the quality of the responses respondents provide. For instance, very short completion time is associated with careless responding (Leiner, 2019). On the other hand, a long response time may suggest respondents experience difficulties answering the questions; namely, Antoun et al. (2017) found that mobile respondents spent more time responding to questions, and at the same time, the answers they provided were less accurate. Thus, long response time can be related to a greater response burden and can indicate that respondents experience difficulties.

Time used to scroll versus navigating between pages could provide different completion times. Scrolling layout has been found to have shorter completion time compared to pagination layout (de Bruijne & Wijnant, 2014; Mavletova & Couper, 2014). A possible explanation for this is that it is quicker to scroll than to click to the previous or next page, and each page must also be loaded. Thus, we would expect to see lower completion time in 2023 with the scrolling layout. If we were to see higher response time for questions that were changed in the pagination layout, it may indicate that respondents experienced higher response burden.

Hypothesis 2: Response time is lower in the scrolling layout (2023) than in the pagination layout (2022).

We estimate the total response time for the question groups where the layout was changed using the TimeStamps in the paradata. Both rounds of the QLS had AutoEnter on the mobile buttons, such that the respondent is automatically and immediately moved to the next item after providing a response, either by moving to the next page or by auto focusing the view on the next question on the page.

3.3 Breakoff

Breakoff refers to when respondents initiate a survey but do not complete the entire questionnaire. A survey breakoff that occurs before completion or before a predetermined limit of the number of questions to be answered is often categorized as nonresponse. Researchers point out that respondent-related or survey-related aspects can cause increasing breakoff rates. For example, lack of motivation, higher task difficulty, and technical problems are some reasons which can increase the probability of breakoff (Steinbrecher et al., 2015).

> back to Table of Contents

Analyzing breakoff rate is an important assessment of survey quality and user experience. If respondents feel the response burden is very high or experience issues, they could decide to not complete the survey. On mobile, the respondents may become frustrated when they have to scroll and zoom to see the question text or if the question text is hard to read on the small screen (Cheung et al., 2016). Vehovar et al. (2022) compared the breakoff rates of different layouts for mobile grid alternatives and saw a higher breakoff rate in the pagination layout in comparison to a scrolling layout and other layout alternatives. We therefore expect that the pagination layouts will result in higher breakoff rates than the scrolling layouts. We consider breakoff rate an important indicator because a difference in breakoff rates may indicate that changing the layout can reduce the response burden. In our analysis, we assess breakoff rates by finding the number of respondents that break off the survey within the question groups with the changed layout.

Hypothesis 3: The breakoff rate in the pagination layout (2022) is higher than in the scrolling layout (2023).

3.4 Error Messages

One possible disadvantage with a scrolling design is that respondents may be more likely to miss a question when there are several questions on one page. For instance, de Bruijne and Wijnant (2014) found an indication of more item nonresponse in the scrolling layout than in the pagination layout; however, the difference was not significant. Mavletova and Couper (2014) did not find any difference in item nonresponse when comparing pagination and scrolling layouts.

In the QLS, response is required to move to the next page, and it is therefore not relevant to assess item nonresponse. Respondents who missed a question would receive a route error informing them to reply to the question before moving on to the next page (see Figure 7). Thus, if it was easier to miss a question in the scrolling layout, we can determine this with the frequency of route errors experienced. Moreover, in the scrolling layout, if respondents tried to move to the next page when they had missed a question, they would have had to scroll back up to reply to the question and then scroll down to move to the next page. This may have been a substantial response burden, especially in question groups with many fields. In the pagination layout, as each item is on a single page, feedback is more immediate if the respondent misses a question.

To examine if it was easier to miss a question in the scrolling layout, we assess the average number of route error messages received per respondent and the share of respondents who received at least one route error. From the audit trail, we find route errors using “Route” in the variable “ErrorKind.”

Hypothesis 4: There are more frequent experiences of route errors and a higher share of respondents receiving route errors at least once in the scrolling layout (2023) than in the pagination layout (2022).

4. Methodology

The Quality of Life (QLS) survey has a sample size of 40,000 people each year, and the sample is drawn to be representative of the Norwegian population aged 18 and above. Using paradata and defining complete response as responding to the last question in the survey, the response rate was 38% in 2022 and 45% in 2023. Between the two years, the questionnaire remained very much the same, except for the change in the small screen layout. However, the 2022 QLS and the 2023 QLS used different contact methods, which may have contributed to the increased response rate in 2023. In 2022, we sent the respondents the invitation to the survey and the link to the questionnaire via both email and SMS (Pettersen & Engvik, 2022). In 2023, we transitioned to using Altinn, the digital mail inbox where public agencies can communicate with individuals and businesses. In Altinn, individuals will, for example, receive information about their tax returns. Users are required to use two-factor authentication to log in to Altinn. In 2023, the invitation to the survey and the link to the questionnaire were sent in Altinn.

> back to Table of Contents

Respondents would get notifications about receiving an Altinn letter from Statistics Norway and reminders about the survey via both SMS and email, but the link to the survey was only sent in Altinn.

By comparing mobile respondents between the two years, this is essentially an observational study that will have limitations regarding self-selection effects. In particular, the contact channel where respondents receive a direct link to the questionnaire might influence the choice of device. For example, de Bruijne and Wijnant (2014) and Mavletova and Couper (2014) found that sending the survey invitation and the link to the questionnaire via SMS led to a higher share of respondents choosing mobile compared to sending them via email. Thus, changing the channel where respondents received the direct link to the survey could influence whether they choose to complete the survey on mobile or on PC. In the QLS 2022, when respondents received the link via SMS and email, 72% of completed surveys were done on mobile, while only 54% were done on mobile in 2023 (see Table 1). As the different contact channels may influence the respondents' choice of device, it may also influence the composition of the net sample on mobile. We see a difference in the composition of the mobile respondents between the two years, especially regarding age, as shown in Table 1. Respondent-related factors could have an impact on the measures we are analyzing; for instance, Zhang et al. (2014) found a relationship between speeding and straight-lining for people with lower education levels, and also for gender. Thus, the different mode of contact between the years could lead to different people choosing to use mobile to respond to the survey, resulting in two different mobile samples between the years. We attempt to reduce confounding effects by conducting exact matching on demographic variables, which we will describe further in the Data Preparation section. But the limitation of not conducting a randomized controlled experiment is something we recognize and will discuss further in the Limitations section.

> back to Table of Contents

Table 1. QLS 2022 and 2023 Descriptive Statistics of Sample

Year	2022 (Pagination Layout)			2023 (Scrolling Layout)			2022 vs 2023 (Pagination vs Scrolling)
Category	**N**	**Completions (% of N)**	**Mobile Completions (% of Completions)**	**N**	**Completions (% of N)**	**Mobile Completions (% of Completions)**	**X2 (df)**
Gender							13.434*** (df=1)
Male	20,385	36	65	20097	43	47	
Female	19,615	40	79	19903	47	64	
Age							179.710*** (df=4)
18-24	4,209	31	85	4,243	40	76	
25-44	13,694	34	81	13,760	43	68	
45-66	13,975	45	68	13,854	52	49	
67-79	5,814	42	60	5,902	45	37	
80+	2,307	19	56	2,241	20	32	
Education							20.006*** (df=3)
Elementary school or lower	9,028	24	80	8,810	30	66	
Upper secondary school	15,467	36	73	15,445	45	55	
University/ University college	13,724	51	69	1,3724	57	53	
Not stated	1,781	18	71	1,226	29	51	
Total	40,000	38	72	40,000	45	55	

*p < 0.05; **p <0.01 ***p<0.001.

5. Data Preparation

We removed sessions with contradicting screen size and layout, as well as respondents with missing information in LayoutSetName. We also excluded respondents that replied to the survey with two different LayoutSetNames. As the interest of the analysis is mobile devices, we will only examine respondents with a small screen layout, and therefore exclude PC respondents.

When assessing breakoff rates, we include all respondents who started the survey by responding to the first question. For the other indicators, we aim to assess activity on the question groups, and we therefore filter to include respondents who responded to at least one of the fields in the middle of the page. This is to ensure that the respondents included in the analysis responded to all the selected question groups we assess. These samples are the foundation of the matching process, which we describe later.

Additionally, we filtered on the middle questions in the question groups because of missing paradata. We were informed by the Blaise support team that missing paradata could occur because of the workings of the Safari browser. As the Safari browser does not always focus on items being clicked and as the audit trail relies on focus being triggered, actions from the Safari browser can be missing. We have not been able to effectively establish if a respondent is unaffected by missing paradata, as absence of activity is hard to determine. Because a substantial share of respondents uses the Safari browser and the missingness does not apply to all observations using this browser, we did not want to exclude all Safari browsers from

> back to Table of Contents

the analysis. Still, we noticed a pattern for missing paradata: activity on the first item was registered but not on the subsequent fields within a page. Therefore, we only included respondents who had responded to at least one of the fields in the middle of the page. As part of the sensitivity analysis, which we discuss in the Results and Data Analysis section, we also ran the analysis while excluding Safari browsers to see if this influenced the results.

As we mentioned in the Methodology section, this is an observational study, and the composition of the mobile samples might be different between the two years. We therefore conducted exact matching to adjust for possible confounding effects due to the different contact methods. The objective of the matching was to construct a subsample of the mobile respondents where the demographic composition would be identical between the two years. To do the exact matching, we stratified the sample of mobile respondents based on the categorical demographic variables gender, education, and age. The categories are shown in Table 4 in the Appendix. The stratification provided us with a total of 40 strata, yet we only used 36 of them because not all strata contained mobile respondents in both 2022 and 2023. In each stratum, we found which year had the smallest number of mobile respondents in the stratum and used that as the benchmark sample size for the stratum. Then for each stratum, we selected a random sample from the year with the highest number of mobile respondents in the stratum. For the breakoff rate analysis, the foundation for the matching process was respondents who answered the first question in the survey. For the other indicators, we selected respondents who responded to all the group questions with the changed layout. After the matching process for breakoff rates, we ended up with 9,640 respondents, using 82% of the mobile respondents from 2022 and 96% from 2023. The matching process for the other indicators left us with 8,371 respondents, using 83% of the mobile respondents from 2022 and 94% from 2023 (see Tables 4 and 5 in Appendix).

For each analysis, we excluded respondents with extreme values from the effective sample. To calculate response time, we used the TimeStamps in the paradata and calculated the time it took from one observed activity until the next activity for each respondent, and then summed up the total time of activities within the question groups for each respondent. We excluded activities that were completed 15 minutes or longer after the previous action, to exclude time spent on not answering the survey. Respondents with the top 5% longest response times were excluded. When counting the frequency of previous page clicks and route errors received per respondent, we removed the top 1% of respondents to avoid extreme values.

As we are interested in the group questions where the layout changed, the basis of the analysis is respondent activity within the 51 fields in the 8 question groups that had the layout changed from pagination to scrolling. Therefore, the results of the indicators are related to activity in the group questions with the changed layout, and not the whole survey.

5.1 Results and Data Analysis

Table 2. Paradata Indicators of Mobile Respondents

Year	2022 (Pagination Layout)			2023 (Scrolling Layout)			2022 vs 2023 (Pagination vs Scrolling)	
Indicator	**Average**	**Std**	**N**	**Average**	**Std**	**N**	**Min - Max**	**p-value**
Previous Page actions (count)	0.92	1.50	8,224	0.07	0.34	8,370	0 – 8	***
Route Errors (count)	1.40	1.97	8,223	0.21	0.65	8,369	0-11	***
Response time (seconds)	342	125	7,984	343	127	7,920	26-724	Not significant

*p < 0.05; **p <0.01 ***p<0.001, Mann-Whitney U-test.

> back to Table of Contents

In 2022, the average times respondents clicked previous page was 0.92 (see Table 1), so we can roughly say that respondents, on average, clicked to the previous page once, but not all respondents clicked to the previous page. The average number of previous pages per respondent within the question group dropped from 0.92 in 2022 to 0.07 in 2023, which supports Hypothesis 1. Naturally, with more pages in the pagination layout than in the scrolling layout, we would expect to see more previous page actions. But considering that having to click previous page increases the response burden, the decrease may suggest improved user experience in the scrolling layout. While we do not have data to see if respondents scrolled up and down on the scrolling layout, we use response time to indicate the total time it took respondents to navigate between the questions. Should the scrolling layout provide better navigation and possibly reduced response burden, we should also expect to see a reduced response time. But contrary to Hypothesis 2, response time did not decrease in 2023 with the scrolling layout. The response time in the question groups was essentially the same between the years. Even though respondents received fewer route errors and did fewer previous page clicks in 2023 than respondents in 2022, the response time was not shorter.

Table 3. Paradata Indicators of Mobile Respondents

Year	2022 (Pagination Layout)		2023 (Scrolling Layout)		2022 vs 2023 (Pagination vs Scrolling)
Indicator	Count (% of N)	N	Count (% of N)	N	p-value (df)
Experienced at least one Route Error	4,664 (56%)	8371	1,227 (15%)	8,371	*** X2 (df= 1)
Breakoff	471 (4.9%)	9640	357 (3.7%)	9,640	*** X2 (df= 1)

*p < 0.05; **p <0.01 ***p<0.001.

Interestingly, the average number of times respondents received route errors decreased from 1.40 in 2022 to 0.21 in 2023 (Table 2). Similarly, the share of respondents receiving at least one route error within the survey question groups was substantially reduced, from 56% of respondents having at least one route error in 2022 to 15% in 2023. The lower share of experienced route errors in the scrolling layout compared to the pagination layout contradicts Hypothesis 4. In fact, respondents were less likely to miss questions in the scrolling layout than in the pagination layout.

When analyzing breakoff rates within the questions with the changed layout, we see a similar trend as with the route error. Out of the respondents that started the survey, 4.9% broke off the survey in the question groups in 2022, while only 3.7% did the same in 2023, supporting Hypothesis 3. This can imply that respondents experienced higher task difficulty in the pagination layout than in the scrolling layout. Seeing that respondents experienced less route errors and were less likely to break off in 2023 than in 2022, it can indicate that respondents experienced lower response burden and higher willingness to complete the questions in the scrolling layout than with the pagination layout.

To test if the difference between years on the indicators we analyze would be due to changing the layout and not due to other factors, we completed some sensitivity analyses with different model inputs. Comparing PC respondents between 2022 and 2023, there is no significant difference between the years for any of the indicators assessed. Moreover, to check that our analysis was not affected by missing paradata, we ran the analysis while excluding sessions from the Safari browser, which is where the missingness occurs, and saw the same significant patterns as when including sessions from the Safari browser. Importantly, we assessed activity on all other questions where the layout was unchanged between the two years, and we saw the same significant patterns: more frequent previous page clicks and route errors received, lower breakoff rate in 2023 compared to 2022, and even significantly lower response time in 2023.

> back to Table of Contents

5.2 Limitations

According to our assumption that any changes between the years in the paradata indicators we analyzed would be due to the layout, we should not see the same pattern in the indicators on questions with no change in the layout. However, when we ran the sensitivity analysis, we still saw the same pattern: reduced frequency of previous page clicks, less experience of route error, and lower breakoff rate among mobile respondents in 2023 but on questions where the layout was the same as in 2022. Thus, the difference in the indicator's measures cannot be isolated to changes in the layout. This limitation is caused by our study design and may be exacerbated due to the different contact channels. We attempted to control for confounding effects by conducting exact matching on sociodemographic variables. Still, it is not possible to control for all possible confounding effects related to the choice of device.

Moreover, contact methods may influence the context and environment for when and how respondents reply to a survey; namely, distracting environments and multitasking while responding to a survey may reduce commitment and ability to concentrate on the questions (de Bruijne & Oudejans, 2015). There might be reason to suspect that the change from sending the survey link via SMS and email to using Altinn can influence when and how respondents reply to the survey. As Altinn is a place where respondents expect to receive important communication from public authorities, they might wait until they are in a familiar environment without distractions before they open the Altinn letter from Statistics Norway. In comparison, when receiving the questionnaire link directly in an SMS or an email, respondents might be more likely to open the survey in distracting environments or on the go. If more mobile respondents were in less distracting environments and in a different mindset when replying to the 2023 QLS, this can have a confounding effect on the indicators assessed here. Thus, we recognized from doing the sensitivity analysis that we would need to do randomized controlled experiments to determine the effect of changes in the layout.

> back to Table of Contents

6. Discussion

We hypothesized that the pagination layout in 2022 would have longer response times, as other research found that response time in a scrolling layout was comparatively shorter (de Bruijne & Wijnant, 2014; Mavletova & Couper, 2014). However, the response time in the question groups remained the same in 2022 and 2023. Considering that respondents received more route errors and returned more frequently to the previous page in the pagination layout, but the response time is the same, this suggests that even if these actions may have broken the survey flow, it did not have an impact on the response time. However, in the pagination design, we can clearly identify when respondents return to the group question text on the first page, while in the scrolling layout, we cannot see if respondents have to scroll to the top of the page to see the text, thus breaking the survey flow.

We saw that the frequency of previous page clicks and route errors per respondent decreased between 2022 to 2023 when changing the layout of the question groups from pagination to scrolling. Similarly, breakoff rates within the questions groups were lower in 2023 compared to 2022. Using route errors as an indicator for how easy it was to miss questions, we found that in 2023, fewer respondents experienced a route error at least once and the average number of time respondents received route errors decreased, which may indicate that the scrolling layout is not more prone to making respondents miss a question. Our results showing reduced frequency of route errors, previous page clicks, and lower breakoff rates from 2022 to 2023 may suggest that the scrolling layout is more suitable for small screens. However, as we have pointed out in the Limitations section, our study design is not a randomized controlled experiment, and we can therefore not be sure whether the improvements are due to the changes in the layout.

Still, we found this analysis useful to start utilizing paradata, and we have gained a more thorough understanding of how we should conduct assessments in the future. Mobile phones are increasingly becoming more popular, and we ought to understand how to improve user experience on small screens and ensure high data quality from the responses. We have started to think about how we can conduct A/B testing to assess small adjustments in the layout using paradata. Importantly, as there are various alternatives for presenting grid questions on mobile (see Vehovar et al., 2022), we hope to conduct future experiments with different layouts and improvements using the knowledge learned from this analysis. Regarding improving the layout further for a better user experience, we would like to assess how respondents experience AutoEnter in our surveys, as well as differentiating the presentation of multiple-choice buttons and single response answer buttons.

In terms of improving how we analyze paradata, we hope to develop more indicators to assess user experience and survey quality. This article focused mainly on the objective response burden of the survey, but an important consideration is to assess subjective user experience and the data quality of the responses. Indicators for data quality include straight-lining, careless responding through random responding, inconsistent answers, inaccurate responses, and response order effects. For instance, when assessing response order effects, Liu and Cernat (2018) found that the mean shifted in the direction to the responses on the left of the screen for mobile respondents for 9- and 11-point scales. We would therefore like to assess how questions with different scales and the number of questions in one group can be influenced differently when adjusting the layout.

> back to Table of Contents

7. References

Antoun, C., Couper, M. P., and Conrad, F. G. (2017). “Effects of mobile versus PC web on survey response quality: A crossover experiment in a probability web panel”, *Public Opinion Quarterly*, *81*(S1), pp.280-306.

Cheung, G., Piskorowski, A., Wood, L. and Peng, H. (2016). “Using Survey Paradata”, Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam

de Bruijne, M., and Oudejans, M. (2015). “Online surveys and the burden of mobile responding”, *Survey Measurements: Techniques, Data Quality and Sources of Error*, pp.130-145.

de Bruijne, M., and Wijnant, A. (2014). “Improving response rates and questionnaire design for mobile web surveys”, *Public Opinion Quarterly*, *78*(4), pp.951-962.

Holmøy, A. and Rossbach, K. (2021) “IKT i husholdningene 2020” (Notater 2021/4). Statistisk sentralbyrå. https://www.ssb.no/teknologi-og-innovasjon/artikler-og-publikasjoner/_attachment/445604?_ts=177a4b71958

Leiner, D. J. (2019). “Too fast, too straight, too weird: Non-reactive indicators for meaningless data in internet surveys”, *Survey Research Methods*, *13*(3), pp.229-248.

Liu, M., and Cernat, A. (2018). “Item-by-item versus matrix questions: A web survey experiment”, *Social Science Computer Review*, *36*(6), pp.690-706.

Mavletova, A., and Couper, M. P. (2014). “Mobile web survey design: scrolling versus paging, SMS versus e-mail invitations”, *Journal of Survey Statistics and Methodology*, *2*(4), pp.498-518.

Pettersen, A and Engvik, M. (2022) “Livskvalitetsundersøkelsen 2022” (2022/35). Statistisk sentralbyrå. https://www.ssb.no/sosiale-forhold-og-kriminalitet/levekar/artikler/livskvalitetsundersokelsen-2022.dokumentasjonsnotat

Revilla, M., Toninelli, D., and Ochoa, C. (2017). “An experiment comparing grids and item-by-item formats in web surveys completed through PCs and smartphones”, *Telematics and Informatics*, *34*(1), pp.30-42.

Steinbrecher, M., Roßmann, J., and Blumenstiel, J. E. (2015). “Why do respondents break off web surveys and does it matter? Results from four follow-up surveys”, *International Journal of Public Opinion Research*, *27*(2), pp.289-302. https://doi.org/10.1093/ijpor/edu025

Vehovar, V., Couper, M. P., and Čehovin, G. (2022). “Alternative layouts for grid questions in PC and mobile web surveys: An experimental evaluation using response quality indicators and survey estimates”, *Social Science Computer Review.* https://doi.org/10.1177/08944393221132644

Zhang, C., and Conrad, F. (2014). “Speeding in web surveys: The tendency to answer very fast and its association with straightlining”, *Survey Research Methods*, *8*(2), pp.127-135. https://doi.org/10.18148/srm/2014.v8i2.5453

> back to Table of Contents

8. Appendix

Table 4. QLS 2022 and 2023 Sample Matching for Previous Page Actions, Route Error, and Response Time

Year	**2022 (Pagination layout)**	**2023 (Scrolling layout)**	**2022/2023 Matched samples**
Category	**N (% of Total)**	**N (% of Total)**	**N (% of Total)**
Gender			
Male	4,329 (43%)	3,586 (40%)	3,133 (40%)
Female	5,703 (57%)	5,311 (60%)	5,058 (60%)
Age			
18-24	1,082 (11%)	1,221 (14%)	1,056 (13%)
25-44	3,564 (36%)	3,767 (42%)	3,410 (41%)
45-66	3,958 (39%)	3,078 (35%)	3,078 (37%)
67-79	1,232 (12%)	722 (8%)	720 (9%)
80+	196 (2%)	109 (1%)	107 (1%)
Education			
Elementary school or lower	1,674 (17%)	1,659 (19%)	1,450 (17%)
Upper secondary school	3,720 (37%)	3,386 (38%)	3,086 (37%)
University/University college	4,426 (44%)	3,680 (41%)	3,680 (44%)
Not Stated	212 (2%)	172 (2%)	155 (2%)
Total	10,032	8,897	8,371

Table 5. QLS 2022 and 2023 Sample Matching for Breakoff Rates

Year	**2022 (Pagination layout)**	**2023 (Scrolling layout)**	**2022/2023 Matched samples**
Category	N (% of Total)	N (% of Total)	N (% of Total)
Gender			
Male	5,183 (44%)	4,111 (41%)	3,916 (41%)
Female	6,584 (56%)	5,907 (59%)	5,724 (59%)
Age			
18-24	1,383 (12%)	1,465 (15%)	1,355 (14%)
25-44	4,339 (37%)	4,317 (43%)	4,052 (42%)
45-66	4,446 (38%)	3,334 (33%)	3,334 (35%)
67-79	1,364 (12%)	780 (8%)	778 (8%)
80+	235 (2%)	122 (1%)	121 (1%)
Education			
Elementary school or lower	2,193 (19%)	2,028 (20%)	1,905 (20%)
Upper secondary school	4,293 (36%)	3,761 (36%)	3,516 (36%)
University/University college	4,961 (42%)	4,007 (42%)	4,003 (42%)
Not Stated	320 (3%)	222 (2%)	216 (2%)
Total	11,767	10,018	9,640

> back to Table of Contents

Table Layouts for Editing

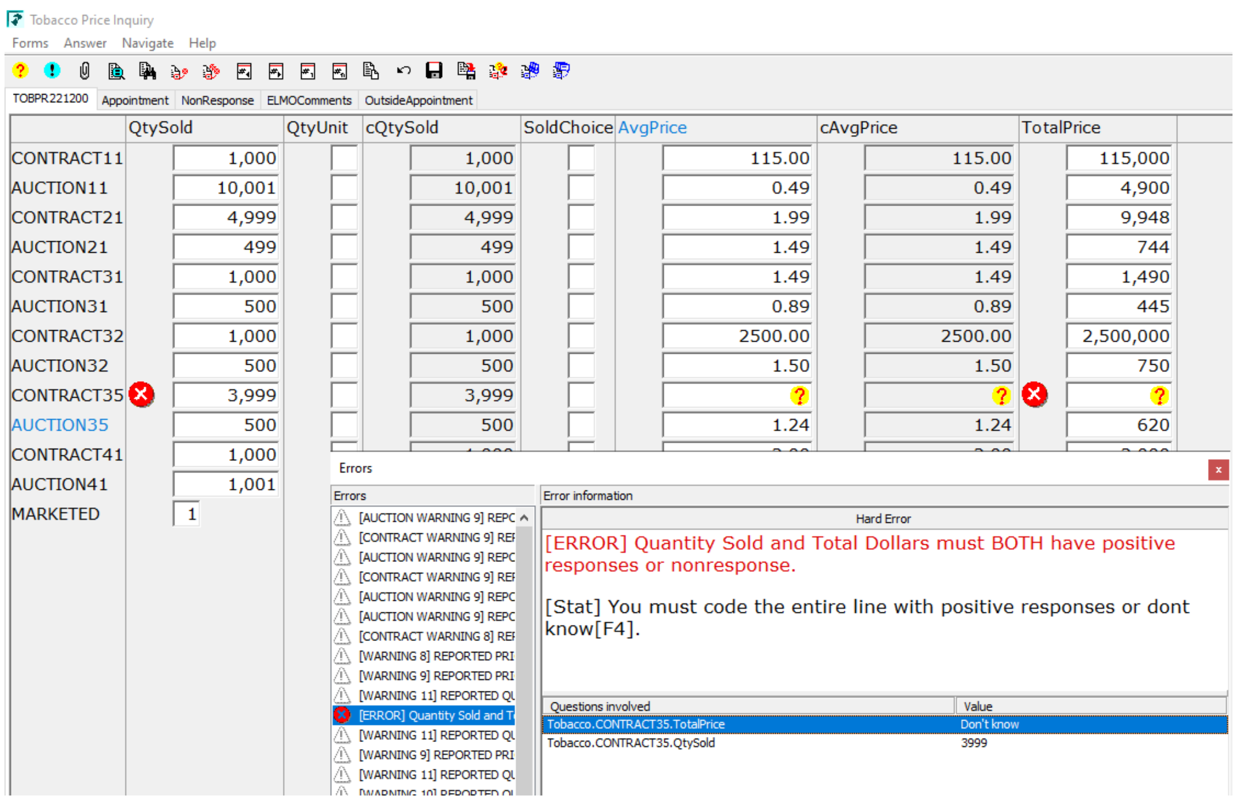
Charles Less, United States Department of Agriculture – NASS

Tables are a versatile tool for organizing and presenting data. They are used to store, sort, filter, and analyze data. Tables are used to present research findings, compare and contrast data, and summarize complex information. Navigating the intricate landscape of tables is a critical skill for editors in any professional setting. Whether you're dealing with financial spreadsheets, scientific data, or editorial schedules, tables serve as a structured method for displaying complex information in an easily digestible format. As simple as they may appear, tables require a keen eye for detail, clarity, and coherence to maximize their impact and usefulness. For USDA/NASS's purposes, we use tables to collect data by topic/agricultural commodity and for editing data by the same standard.

1. Blaise 4 Tables at a Glance

Ever since Blaise 4 was implemented for USDA/NASS, tables have served a very useful purpose for our editing. Tables in Blaise 4 allow for organizing data more efficiently, allow more fields on a page, and assist our statisticians in editing/reviewing data for various commodities in one page. In Blaise 4, we had the luxury of the Popup Record Error Listing and Involved Fields, along with our table, to quickly and effectively edit/review data. Figure 1 shows an example of the use of tables in Blaise 4. Here, data are collected on the pounds sold, average price per pound, and total dollars received for various types of tobacco sold, either through contract or through auction houses. Notice from the figure, developers did not have to contend with the table spacing to fit a check or signal within the rows or at the table level.

Figure 1. Table Editing Environment in Blaise 4



	QtySold	QtyUnit	cQtySold	SoldChoice	AvgPrice	cAvgPrice	TotalPrice
CONTRACT11	1,000		1,000		115.00	115.00	115,000
AUCTION11	10,001		10,001		0.49	0.49	4,900
CONTRACT21	4,999		4,999		1.99	1.99	9,948
AUCTION21	499		499		1.49	1.49	744
CONTRACT31	1,000		1,000		1.49	1.49	1,490
AUCTION31	500		500		0.89	0.89	445
CONTRACT32	1,000		1,000		2500.00	2500.00	2,500,000
AUCTION32	500		500		1.50	1.50	750
CONTRACT35 ✗	3,999		3,999		?	?	✗ ?
AUCTION35	500		500		1.24	1.24	620
CONTRACT41	1,000						
AUCTION41	1,001						
MARKETED	1						

Errors

Errors
- [AUCTION WARNING 9] REPC
- [CONTRACT WARNING 9] REF
- [AUCTION WARNING 9] REPC
- [CONTRACT WARNING 9] REF
- [AUCTION WARNING 9] REPC
- [AUCTION WARNING 9] REPC
- [CONTRACT WARNING 8] REF
- [WARNING 8] REPORTED PRI
- [WARNING 9] REPORTED PRI
- [WARNING 11] REPORTED QU
- [ERROR] Quantity Sold and T
- [WARNING 11] REPORTED QU
- [WARNING 9] REPORTED PRI
- [WARNING 11] REPORTED QU
- [WARNING 10] REPORTED QU

Error information

Hard Error

[ERROR] Quantity Sold and Total Dollars must BOTH have positive responses or nonresponse.

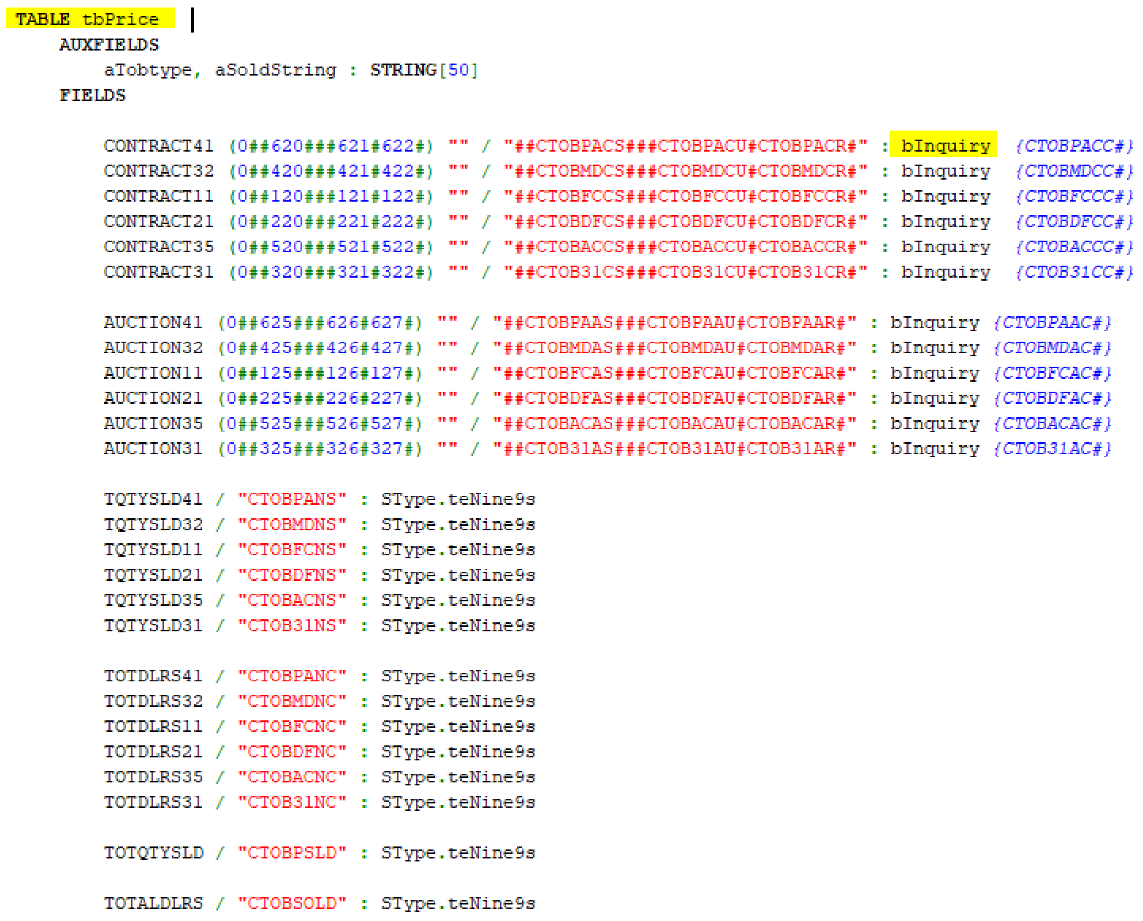
[Stat] You must code the entire line with positive responses or dont know[F4].

Questions involved	Value
Tobacco.CONTRACT35.TotalPrice	Don't know
Tobacco.CONTRACT35.QtySold	3999

> back to Table of Contents

The code to implement a table is straightforward. Tables in Blaise 4 are in full use, and one merely had to apply it in the code by declaring the table (TABLE tbPrice) and adding the appropriate blocks (bInquiry) within the table, and then efficient tables for Interviewing and Editing were ready. Figure 2 below shows an example of how the table storing the tobacco information is stored by type and by point of sale.

Figure 2. Table Code Blaise 4: Declaring a Table



```
TABLE tbPrice
    AUXFIELDS
        aTobtype, aSoldString : STRING[50]
    FIELDS

        CONTRACT41 (0##620###621#622#) "" / "##CTOBPACS###CTOBPACU#CTOBPACR#" : bInquiry  {CTOBPACC#}
        CONTRACT32 (0##420###421#422#) "" / "##CTOBMDCS###CTOBMDCU#CTOBMDCR#" : bInquiry  {CTOBMDCC#}
        CONTRACT11 (0##120###121#122#) "" / "##CTOBFCCS###CTOBFCCU#CTOBFCCR#" : bInquiry  {CTOBFCCC#}
        CONTRACT21 (0##220###221#222#) "" / "##CTOBDFCS###CTOBDFCU#CTOBDFCR#" : bInquiry  {CTOBDFCC#}
        CONTRACT35 (0##520###521#522#) "" / "##CTOBACCS###CTOBACCU#CTOBACCR#" : bInquiry  {CTOBACCC#}
        CONTRACT31 (0##320###321#322#) "" / "##CTOB31CS###CTOB31CU#CTOB31CR#" : bInquiry  {CTOB31CC#}

        AUCTION41 (0##625###626#627#) "" / "##CTOBPAAS###CTOBPAAU#CTOBPAAR#" : bInquiry {CTOBPAAC#}
        AUCTION32 (0##425###426#427#) "" / "##CTOBMDAS###CTOBMDAU#CTOBMDAR#" : bInquiry {CTOBMDAC#}
        AUCTION11 (0##125###126#127#) "" / "##CTOBFCAS###CTOBFCAU#CTOBFCAR#" : bInquiry {CTOBFCAC#}
        AUCTION21 (0##225###226#227#) "" / "##CTOBDFAS###CTOBDFAU#CTOBDFAR#" : bInquiry {CTOBDFAC#}
        AUCTION35 (0##525###526#527#) "" / "##CTOBACAS###CTOBACAU#CTOBACAR#" : bInquiry {CTOBACAC#}
        AUCTION31 (0##325###326#327#) "" / "##CTOB31AS###CTOB31AU#CTOB31AR#" : bInquiry {CTOB31AC#}

        TQTYSLD41 / "CTOBPANS" : SType.teNine9s
        TQTYSLD32 / "CTOBMDNS" : SType.teNine9s
        TQTYSLD11 / "CTOBFCNS" : SType.teNine9s
        TQTYSLD21 / "CTOBDFNS" : SType.teNine9s
        TQTYSLD35 / "CTOBACNS" : SType.teNine9s
        TQTYSLD31 / "CTOB31NS" : SType.teNine9s

        TOTDLRS41 / "CTOBPANC" : SType.teNine9s
        TOTDLRS32 / "CTOBMDNC" : SType.teNine9s
        TOTDLRS11 / "CTOBFCNC" : SType.teNine9s
        TOTDLRS21 / "CTOBDFNC" : SType.teNine9s
        TOTDLRS35 / "CTOBACNC" : SType.teNine9s
        TOTDLRS31 / "CTOB31NC" : SType.teNine9s

        TOTQTYSLD / "CTOBPSLD" : SType.teNine9s

        TOTALDLRS / "CTOBSOLD" : SType.teNine9s
```

For editing in Blaise 4, the Popup Record Error Listing and Involved Fields are great for editors to handle error review. Along with the table, space was not an issue for reviewing errors. Users could easily resolve warnings, and the error listing allowed quick access to the 'involving' field. With Blaise 5, the missing error listing box posed a challenge.

2. Enter Blaise 5

Developing Blaise 5 tables with the Layout Editor took considerable effort to format the tables in a more logical presentation to our editors. Equipped with the samples provided by Statistics Netherlands, we used/borrowed some work from the samples to create our table layouts and planned to create a documentation for our developers to use when implementing tables in their own projects. As with many things at NASS, much customization is required between different surveys.

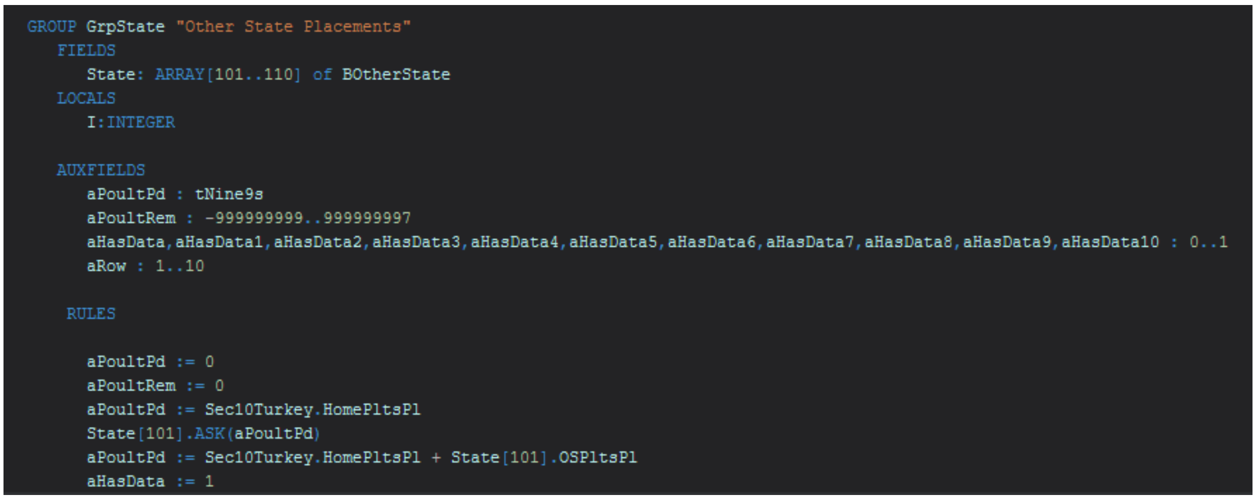
> back to Table of Contents

USDA/NASS conducts many regional and national surveys during the course of a year, and there are many opportunities to implement tables. We decided to implement tables in Blaise 5 for editing a small survey that occurs twice a year and has a small sample. The survey was able to implement a table, and editors had a small enough sample to meet their due date for finishing the editing and making corrections.

The table layout developed would only have 10 rows and a small number of columns. Edits would exist involving fields in these tables (groups) and display for calculation issues or violating our external edit limits, which are also used in this survey.

For Blaise 5 coding, we start with the Group declaration. See Figure 3. Once the Group is coded, the table template(s) are at our disposal. The example shown in the figure comes from a survey designed to forecast the number of turkeys raised in the United States based on poult placement at the state and national level. There are operations that are large enough that they report. In this table, each row can be used to report poult placement by state. Each row has the same number of columns, as the questions are the same for each state.

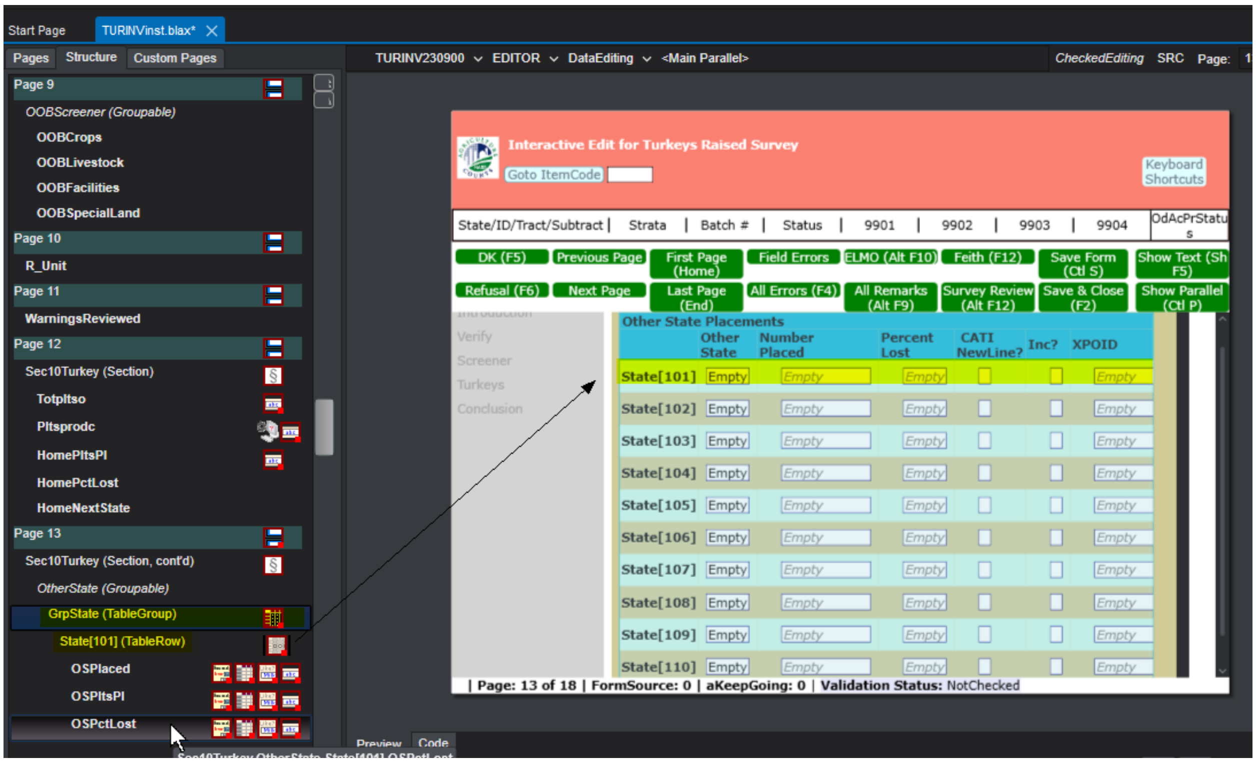
Figure 3. Blaise 5 Code for Setting Up Table Layout



```
GROUP GrpState "Other State Placements"
   FIELDS
      State: ARRAY[101..110] of BOtherState
   LOCALS
      I:INTEGER

   AUXFIELDS
      aPoultPd : tNine9s
      aPoultRem : -999999999..999999997
      aHasData,aHasData1,aHasData2,aHasData3,aHasData4,aHasData5,aHasData6,aHasData7,aHasData8,aHasData9,aHasData10 : 0..1
      aRow : 1..10

    RULES

      aPoultPd := 0
      aPoultRem := 0
      aPoultPd := Sec10Turkey.HomePltsPl
      State[101].ASK(aPoultPd)
      aPoultPd := Sec10Turkey.HomePltsPl + State[101].OSPltsPl
      aHasData := 1
```

> back to Table of Contents

Figure 4. Table in Layout Tab



State/ID/Tract/Subtract | Strata | Batch # | Status | 9901 | 9902 | 9903 | 9904 | OdAcPrStatus

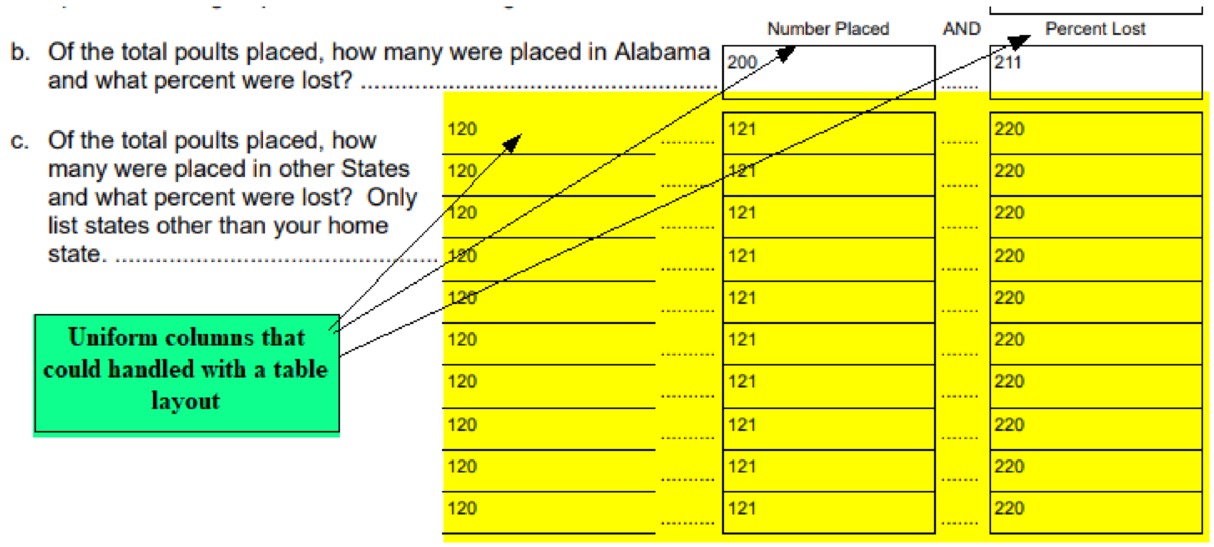
Other State Placements

	Other State	Number Placed	Percent Lost	CATI NewLine?	Inc?	XPOID
State[101]	Empty	Empty	Empty	☐	☐	Empty
State[102]	Empty	Empty	Empty	☐	☐	Empty
State[103]	Empty	Empty	Empty	☐	☐	Empty
State[104]	Empty	Empty	Empty	☐	☐	Empty
State[105]	Empty	Empty	Empty	☐	☐	Empty
State[106]	Empty	Empty	Empty	☐	☐	Empty
State[107]	Empty	Empty	Empty	☐	☐	Empty
State[108]	Empty	Empty	Empty	☐	☐	Empty
State[109]	Empty	Empty	Empty	☐	☐	Empty
State[110]	Empty	Empty	Empty	☐	☐	Empty

| Page: 13 of 18 | FormSource: 0 | aKeepGoing: 0 | Validation Status: NotChecked

The approach taken was to set the Layout instructions for the table using the Layout tab in the Control Centre and the Blaise Resource Editor, then start with the first row of that table. Depending on the uniformity of that row in relation to the following rows in the questionnaire we use (Figure 4), one could make use of the first row's layout instructions and parameters being set in the layout editor, and then cut and paste the other rows accordingly within the instrument's Source tab in the Control Centre.

Figure 5. Uniform Table and Small Number of Columns


Uniform columns that could handled with a table layout

b. Of the total poults placed, how many were placed in Alabama and what percent were lost? ....

c. Of the total poults placed, how many were placed in other States and what percent were lost? Only list states other than your home state. ....

	Number Placed	AND	Percent Lost
	200		211
120	121		220
120	121		220
120	121		220
120	121		220
120	121		220
120	121		220
120	121		220
120	121		220
120	121		220
120	121		220

> back to Table of Contents

3. Trials and Tribulations

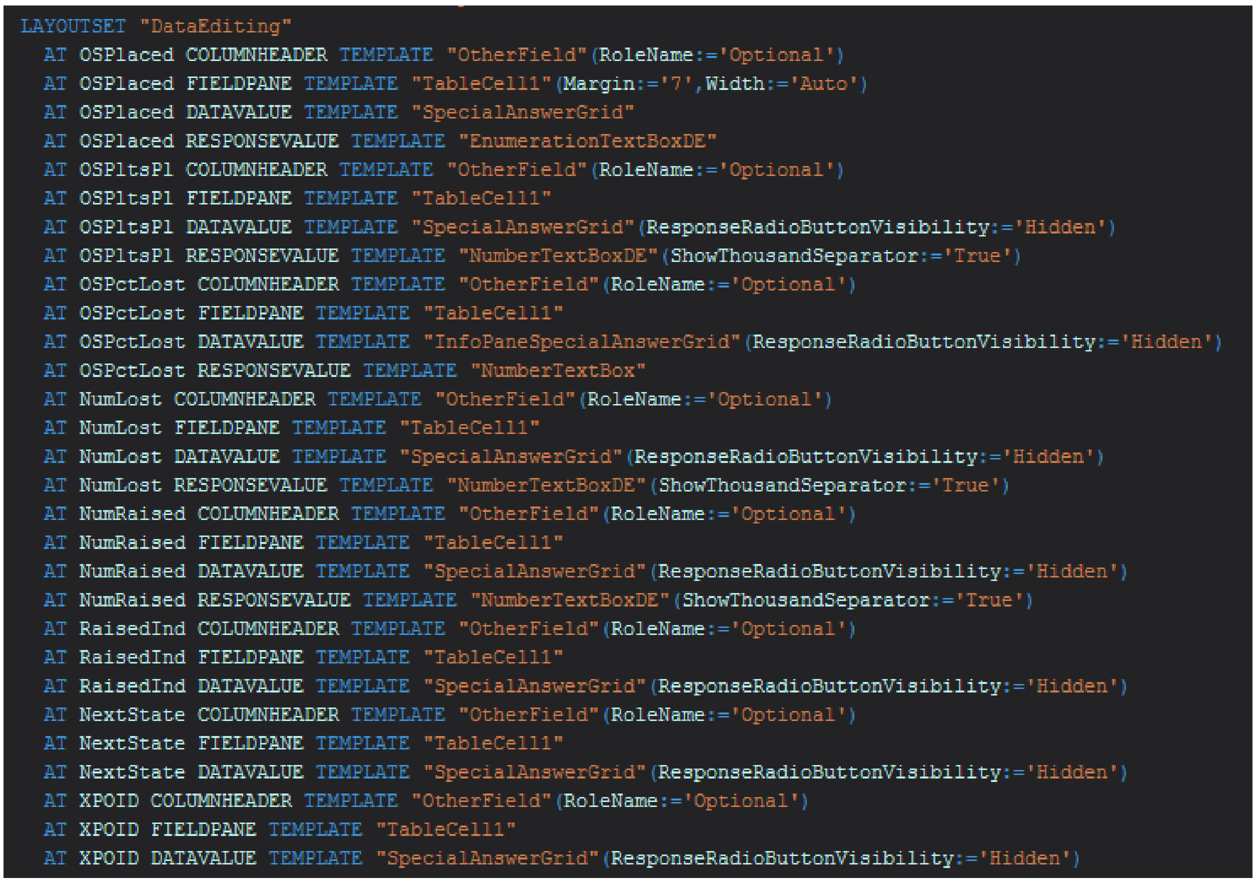
When first using the table layout, we started going through each row using the Layout tab and setting the properties. As Blaise 5 developers know, if you strictly use the Layout tab and the Blaise Resource Editor, the development could be a very long task to complete a layout. As shown in the Figure 5 table and columns, we could use the Layout tab and get away with setting properties. We only had to set the layout instructions and parameters for 10 rows. Even with 10 rows, this was tedious work within the Layout tab of the Control Centre when setting the properties in the Layout tab and having it sync to the code in the Source tab.

As we gained more knowledge of adjusting the rows to our liking, we could synchronize the layout for that first row to the source and then use that first row to cut and paste. Or in this case, since it was an array, one merely had to edit the first row to be a generic layout instruction for all rows for this survey.

In the code, we were able to use that first row to create instructions for that row (State [101]), and then adjust it with an open set of brackets for all rows in the Group. See Figures 6, 7, and 8.

At the Block level (that is called from the Group code):

Figure 6. Row and Cell Level Layout Instructions

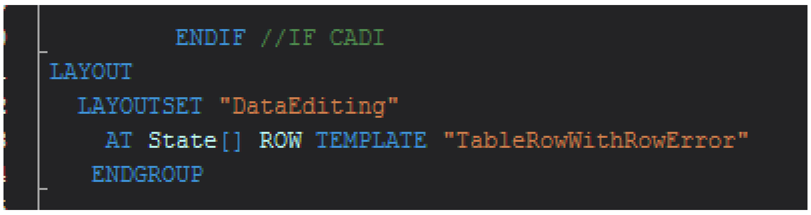


```
LAYOUTSET "DataEditing"
  AT OSPlaced COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT OSPlaced FIELDPANE TEMPLATE "TableCell1"(Margin:='7',Width:='Auto')
  AT OSPlaced DATAVALUE TEMPLATE "SpecialAnswerGrid"
  AT OSPlaced RESPONSEVALUE TEMPLATE "EnumerationTextBoxDE"
  AT OSPltsPl COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT OSPltsPl FIELDPANE TEMPLATE "TableCell1"
  AT OSPltsPl DATAVALUE TEMPLATE "SpecialAnswerGrid"(ResponseRadioButtonVisibility:='Hidden')
  AT OSPltsPl RESPONSEVALUE TEMPLATE "NumberTextBoxDE"(ShowThousandSeparator:='True')
  AT OSPctLost COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT OSPctLost FIELDPANE TEMPLATE "TableCell1"
  AT OSPctLost DATAVALUE TEMPLATE "InfoPaneSpecialAnswerGrid"(ResponseRadioButtonVisibility:='Hidden')
  AT OSPctLost RESPONSEVALUE TEMPLATE "NumberTextBox"
  AT NumLost COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT NumLost FIELDPANE TEMPLATE "TableCell1"
  AT NumLost DATAVALUE TEMPLATE "SpecialAnswerGrid"(ResponseRadioButtonVisibility:='Hidden')
  AT NumLost RESPONSEVALUE TEMPLATE "NumberTextBoxDE"(ShowThousandSeparator:='True')
  AT NumRaised COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT NumRaised FIELDPANE TEMPLATE "TableCell1"
  AT NumRaised DATAVALUE TEMPLATE "SpecialAnswerGrid"(ResponseRadioButtonVisibility:='Hidden')
  AT NumRaised RESPONSEVALUE TEMPLATE "NumberTextBoxDE"(ShowThousandSeparator:='True')
  AT RaisedInd COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT RaisedInd FIELDPANE TEMPLATE "TableCell1"
  AT RaisedInd DATAVALUE TEMPLATE "SpecialAnswerGrid"(ResponseRadioButtonVisibility:='Hidden')
  AT NextState COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT NextState FIELDPANE TEMPLATE "TableCell1"
  AT NextState DATAVALUE TEMPLATE "SpecialAnswerGrid"(ResponseRadioButtonVisibility:='Hidden')
  AT XPOID COLUMNHEADER TEMPLATE "OtherField"(RoleName:='Optional')
  AT XPOID FIELDPANE TEMPLATE "TableCell1"
  AT XPOID DATAVALUE TEMPLATE "SpecialAnswerGrid"(ResponseRadioButtonVisibility:='Hidden')
```

> back to Table of Contents

At the Group code level (displaying the open set of brackets to handle every row):

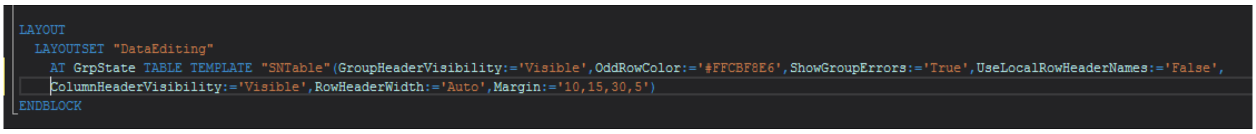
Figure 7. Table-Level Layout Instructions



```
        ENDIF //IF CADI
LAYOUT
  LAYOUTSET "DataEditing"
    AT State[] ROW TEMPLATE "TableRowWithRowError"
 ENDGROUP
```

At the Main Block level (where Group is located):

Figure 8. Main Bloc Layout Instructions



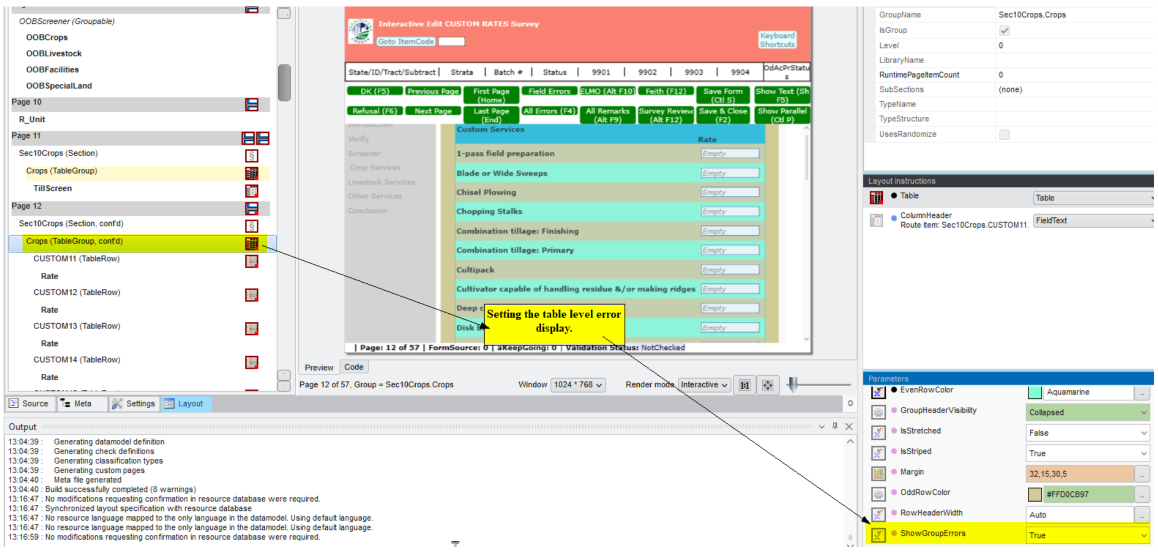
```
LAYOUT
  LAYOUTSET "DataEditing"
    AT GrpState TABLE TEMPLATE "SNTable"(GroupHeaderVisibility:='Visible',OddRowColor:='#FFCBF8E6',ShowGroupErrors:='True',UseLocalRowHeaderNames:='False',
    ColumnHeaderVisibility:='Visible',RowHeaderWidth:='Auto',Margin:='10,15,30,5')
ENDBLOCK
```

4. Displaying Error Text in Tables

As previously mentioned, USDA/NASS relied heavily on the Popup Record Error Listing and Involved Fields to handle error displays, as well as using the jump to fields to resolve CHECKS with Blaise 4.

As developers/layout designers, we had to make 'space' for error messages within the table environment. Blaise 5 allows developers to display checks and signals at the table level and at the row level. See Figures 9 and 10 We have used both for recent surveys and the results are mixed. Showing too many of the same errors caused editor fatigue. An editor can be overwhelmed with row and table both displayed, giving the appearance of having a lot of editing work to do. Also, because these tables can display many checks/errors, the table-level checks/errors could not fit within a user's screen. As these checks/errors were resolved, the listings at the bottom of a table were gradually reduced.

Figure 9. Table/Group Error Listing Display Settings in Layout Tab



State/ID/Tract/Subtract	Strata	Batch #	Status	9901	9902	9903	9904	OdAcPrStatus

Custom Services	Rate
1-pass field preparation	Empty
Blade or Wide Sweeps	Empty
Chisel Plowing	Empty
Chopping Stalks	Empty
Combination tillage: Finishing	Empty
Combination tillage: Primary	Empty
Cultipack	Empty
Cultivator capable of handling residue &/or making ridges	Empty
Deep c	Empty
Disk	Empty

| Page: 12 of 57 | FormSource: 0 | aKeepGoing: 0 | Validation Status: NotChecked

Page 12 of 57, Group = Sec10Crops.Crops

GroupName	Sec10Crops.Crops
IsGroup	✓
Level	0
LibraryName	
RuntimePageItemCount	0
SubSections	(none)
TypeName	
TypeStructure	
UsesRandomize	☐

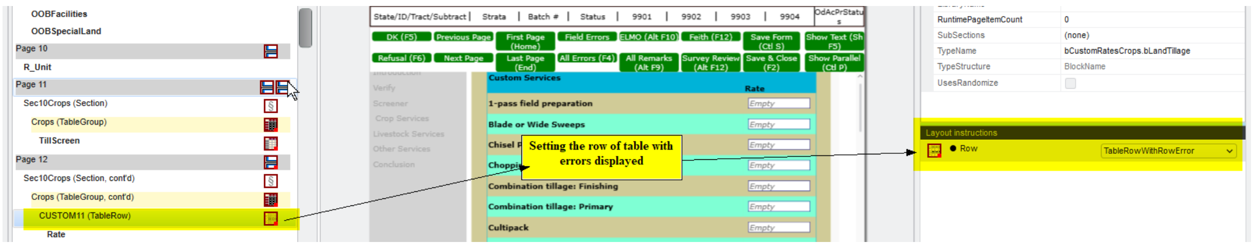
● Table	Table
● ColumnHeader Route Item: Sec10Crops.CUSTOM11.	FieldText

● EvenRowColor	Aquamarine
● GroupHeaderVisibility	Collapsed
● IsStretched	False
● IsStriped	True
● Margin	32,15,30,5
● OddRowColor	#FFD0CB97
● RowHeaderWidth	Auto
● ShowGroupErrors	True

```
Output
13:04:39 :    Generating datamodel definition
13:04:39 :    Generating check definitions
13:04:39 :    Generating classification types
13:04:39 :    Generating custom pages
13:04:40 :    Meta file generated
13:04:40 : Build successfully completed (8 warnings)
13:16:47 : No modifications requesting confirmation in resource database were required.
13:16:47 : Synchronized layout specification with resource database
13:16:47 : No resource language mapped to the only language in the datamodel. Using default language.
13:16:47 : No resource language mapped to the only language in the datamodel. Using default language.
13:16:59 : No modifications requesting confirmation in resource database were required.
```

> back to Table of Contents

Figure 10. Row-Level Error Listing Display Settings in Layout Tab



OOBFacilities
OOBSpecialLand
Page 10
R_Unit
Page 11
Sec10Crops (Section)
Crops (TableGroup)
TillScreen
Page 12
Sec10Crops (Section, cont'd)
Crops (TableGroup, cont'd)
CUSTOM11 (TableRow)
Rate

Custom Services	Rate
1-pass field preparation	Empty
Blade or Wide Sweeps	Empty
Chisel P	Empty
Choppi	Empty
Combination tillage: Finishing	Empty
Combination tillage: Primary	Empty
Cultipack	Empty

RuntimePageItemCount	0
SubSections	(none)
TypeName	bCustomRatesCrops.bLandTillage
TypeStructure	BlockName
UsesRandomize	

Layout instructions
● Row TableRowWithRowError

When the table-level errors settings were used, the error displayed at the bottom of the table. See Figure 11. When the row-level error settings were adjusted, the errors for each row would appear. See Figure 12.

Figure 11. Table-Level Error

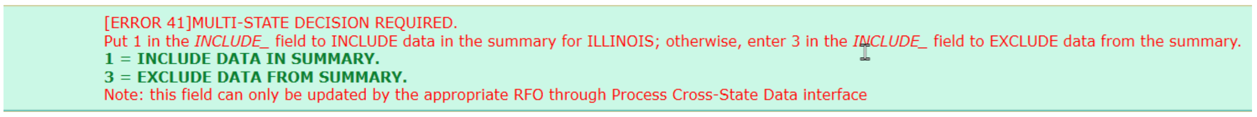
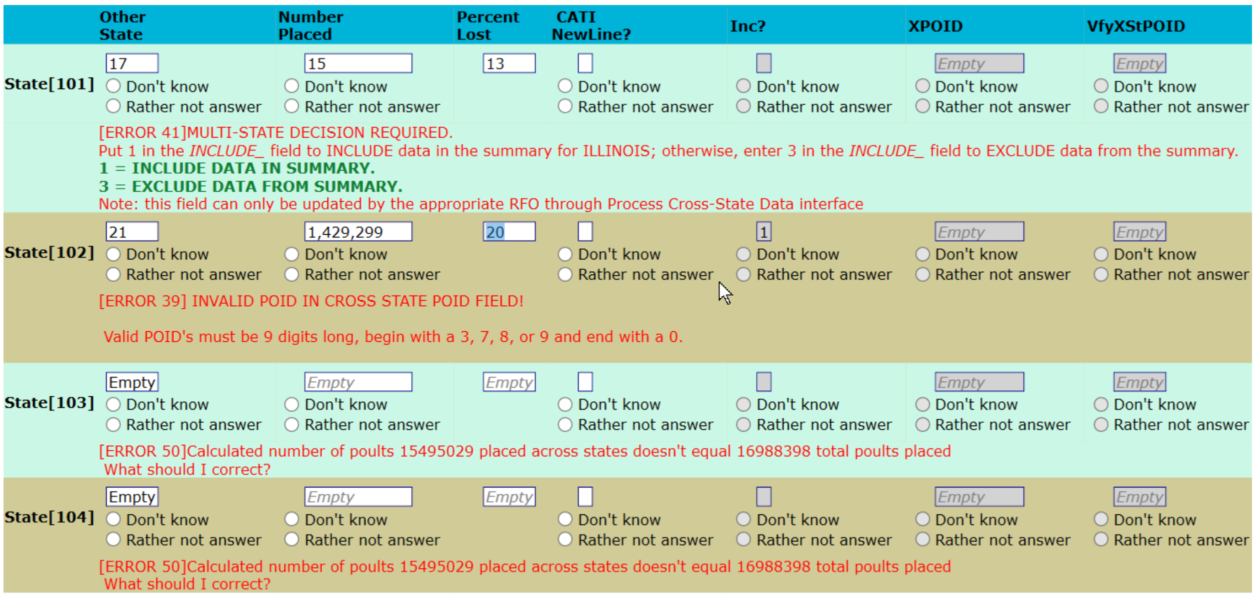

[ERROR 41]MULTI-STATE DECISION REQUIRED.
Put 1 in the *INCLUDE_* field to INCLUDE data in the summary for ILLINOIS; otherwise, enter 3 in the *INCLUDE_* field to EXCLUDE data from the summary.
1 = INCLUDE DATA IN SUMMARY.
3 = EXCLUDE DATA FROM SUMMARY.
Note: this field can only be updated by the appropriate RFO through Process Cross-State Data interface

Figure 12. Row-Level Errors



	Other State	Number Placed	Percent Lost	CATI NewLine?	Inc?	XPOID	VfyXStPOID
State[101]	17 ○ Don't know ○ Rather not answer	15 ○ Don't know ○ Rather not answer	13	☐ ○ Don't know ○ Rather not answer	☐ ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer
	[ERROR 41]MULTI-STATE DECISION REQUIRED. Put 1 in the *INCLUDE_* field to INCLUDE data in the summary for ILLINOIS; otherwise, enter 3 in the *INCLUDE_* field to EXCLUDE data from the summary. **1 = INCLUDE DATA IN SUMMARY.** **3 = EXCLUDE DATA FROM SUMMARY.** Note: this field can only be updated by the appropriate RFO through Process Cross-State Data interface						
State[102]	21 ○ Don't know ○ Rather not answer	1,429,299 ○ Don't know ○ Rather not answer	20	☐ ○ Don't know ○ Rather not answer	1 ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer
	[ERROR 39] INVALID POID IN CROSS STATE POID FIELD! Valid POID's must be 9 digits long, begin with a 3, 7, 8, or 9 and end with a 0.						
State[103]	Empty ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer	Empty	☐ ○ Don't know ○ Rather not answer	☐ ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer
	[ERROR 50]Calculated number of poults 15495029 placed across states doesn't equal 16988398 total poults placed What should I correct?						
State[104]	Empty ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer	Empty	☐ ○ Don't know ○ Rather not answer	☐ ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer	Empty ○ Don't know ○ Rather not answer
	[ERROR 50]Calculated number of poults 15495029 placed across states doesn't equal 16988398 total poults placed What should I correct?						

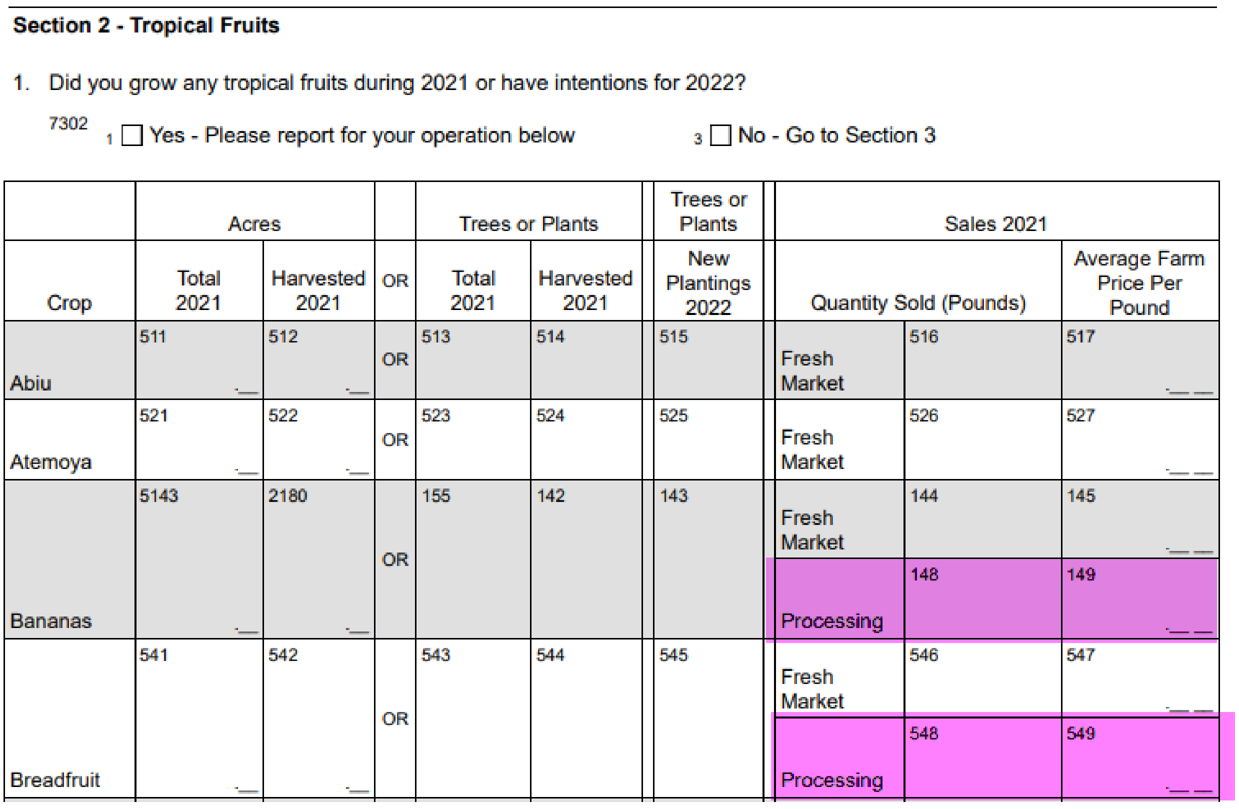
5. Problems with Complex Tables

With our limited success in setting up editing with a small survey, an attempt was made on a survey with large tables and a lot of columns. Hawaii has a Tropical Specialties Survey with a multitude of crops grown. See Figure 13. We wanted to use this specific survey to see how the edit would render. My attempt to adjust to large tables caused issues with our editors seeing the big picture. For our situation, the number of columns would not fit our layout appearance. One could resolve long tables with a multitude of crops with scroll bars, but it had a difficult time adapting to columns. See Figure 14. We implemented horizontal scroll bars, but the error listings involving the crop label and columns for processing were not helpful for the editor. It was recommended to insert a single column table, but time ran short to institute

> back to Table of Contents

that table, and even if that table was inserted, it doesn't guarantee that the crop name row would align with the data.

Figure 13. Hawaii Tropical Specialties



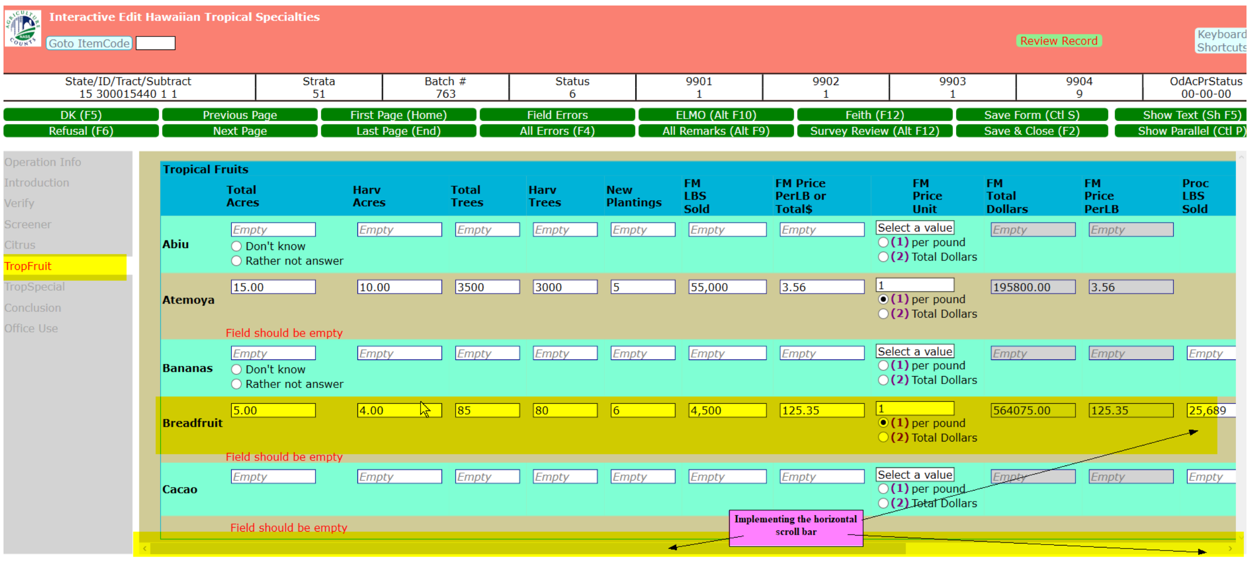
Section 2 - Tropical Fruits

1. Did you grow any tropical fruits during 2021 or have intentions for 2022?

7302 ₁ ☐ Yes - Please report for your operation below ₃ ☐ No - Go to Section 3

	Acres			Trees or Plants		Trees or Plants	Sales 2021		
Crop	Total 2021	Harvested 2021	OR	Total 2021	Harvested 2021	New Plantings 2022		Quantity Sold (Pounds)	Average Farm Price Per Pound
Abiu	511	512	OR	513	514	515	Fresh Market	516	517
Atemoya	521	522	OR	523	524	525	Fresh Market	526	527
Bananas	5143	2180	OR	155	142	143	Fresh Market	144	145
							Processing	148	149
Breadfruit	541	542	OR	543	544	545	Fresh Market	546	547
							Processing	548	549

Figure 14. Table Edit Rendering



Interactive Edit Hawaiian Tropical Specialties

Goto ItemCode

Review Record

Keyboard Shortcuts

State/ID/Tract/Subtract	Strata	Batch #	Status	9901	9902	9903	9904	OdAcPrStatus
15 300015440 1 1	51	763	6	1	1	1	9	00-00-00

DK (F5) | Previous Page | First Page (Home) | Field Errors | ELMO (Alt F10) | Feith (F12) | Save Form (Ctl S) | Show Text (Sh F5)

Refusal (F6) | Next Page | Last Page (End) | All Errors (F4) | All Remarks (Alt F9) | Survey Review (Alt F12) | Save & Close (F2) | Show Parallel (Ctl P)

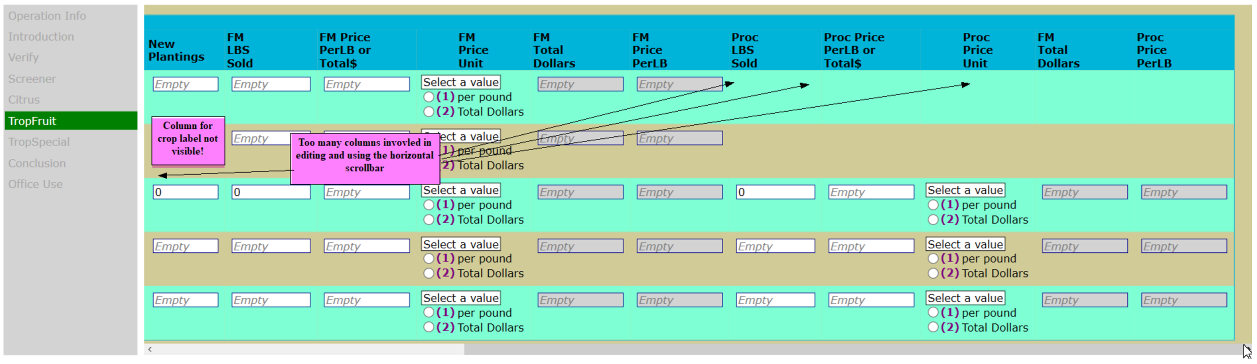
Operation Info
Introduction
Verify
Screener
Citrus
TropFruit
TropSpecial
Conclusion
Office Use

Tropical Fruits

	Total Acres	Harv Acres	Total Trees	Harv Trees	New Plantings	FM LBS Sold	FM Price PerLB or Total$	FM Price Unit	FM Total Dollars	FM Price PerLB	Proc LBS Sold
Abiu	Empty ○ Don't know ○ Rather not answer	Empty	Empty	Empty	Empty	Empty	Empty	Select a value ○ (1) per pound ○ (2) Total Dollars	Empty	Empty	
Atemoya	15.00 Field should be empty	10.00	3500	3000	5	55,000	3.56	1 ● (1) per pound ○ (2) Total Dollars	195800.00	3.56	
Bananas	Empty ○ Don't know ○ Rather not answer	Empty	Empty	Empty	Empty	Empty	Empty	Select a value ○ (1) per pound ○ (2) Total Dollars	Empty	Empty	Empty
Breadfruit	5.00 Field should be empty	4.00	85	80	6	4,500	125.35	1 ● (1) per pound ○ (2) Total Dollars	564075.00	125.35	25,689
Cacao	Empty Field should be empty	Empty	Empty	Empty	Empty	Empty	Empty	Select a value ○ (1) per pound ○ (2) Total Dollars	Empty	Empty	Empty

> back to Table of Contents

Figure 15. Continuation of Table Rendering


Operation Info
Introduction
Verify
Screener
Citrus
TropFruit
TropSpecial
Conclusion
Office Use
Column for crop label not visible!
Too many columns invovled in editing and using the horizontal scrollbar

New Plantings	FM LBS Sold	FM Price PerLB or Total$	FM Price Unit	FM Total Dollars	FM Price PerLB	Proc LBS Sold	Proc Price PerLB or Total$	Proc Price Unit	FM Total Dollars	Proc Price PerLB
Empty	Empty	Empty	Select a value ○(1) per pound ○(2) Total Dollars	Empty	Empty					
	Empty		ect a value 1) per pound 2) Total Dollars	Empty	Empty					
0	0	Empty	Select a value ○(1) per pound ○(2) Total Dollars	Empty	Empty	0	Empty	Select a value ○(1) per pound ○(2) Total Dollars	Empty	Empty
Empty	Empty	Empty	Select a value ○(1) per pound ○(2) Total Dollars	Empty	Empty	Empty	Empty	Select a value ○(1) per pound ○(2) Total Dollars	Empty	Empty
Empty	Empty	Empty	Select a value ○(1) per pound ○(2) Total Dollars	Empty	Empty	Empty	Empty	Select a value ○(1) per pound ○(2) Total Dollars	Empty	Empty

6. Shortcomings (The Troubles with Tables)

Table editing continues to be slow. Saving updates can take several seconds per field. Additionally, USDA/NASS utilizes a Generic InDepth data storage type pointing to MySQL, and we are left wondering if server contact can also be a concern. We also operate within a Citrix environment, and we wonder if that could also create issues. It was recommended that we change our settings to have changes saved at the page level to handle this lag in saving updates; however, there are calculations that need to take place to make sure some columns total correctly at that page level. We initially instituted a page save through the Data Entry Settings Tab, but got into trouble when editors made updates, thought they were clean, and saved the form, resulting in the record's ValidationStatus appearing as dirty.

Because we no longer have the Popup Record Error Listing and Involved Fields, we had to resort to a layout page that displays the 'error list box.' Jumping from the error list box to an 'involving field' that is a table field is also a slow process and can take several seconds to land on said field.

The lag in table saving can pose challenges to our editors, as much of our data comes in toward the end of a survey. We give our enumerators in the field and our statisticians a window to account for data, and sometimes that window is stretched due to weather conditions or farm operator availability. Editing a large amount of data stored in tables like these at the end of that short survey window can pressure our statisticians.

Space is also an issue. One of the surveys that we implemented tables on was a survey that had many sections and columns. Due to lack of space, we had wanted to use a freeze column, but the layout does not have that to implement. We did have conversations with Statistics Netherlands, and since the table layout is not a not a true table, it does not have that column freeze attribute. To compensate for the lack of a hold/freeze column, we had to resort to the use of a horizontal scroll bar to allow editing for some columns that needed to be edited, and this caused some consternation as the reviewers/editors lost track of what crop they were reviewing.

7. Conclusion

In this paper, we have discussed the use of table layouts for editing data. We have seen that Blaise 5 offers much flexibility to display tables in very useful and appealing ways. We can use tables effectively to display reported data, as well as metadata, and we are able to effectively use small tables for editing. We do continue to struggle with larger tables where there are large numbers of rows and columns that

> back to Table of Contents

require a lot of editing for cell values for a row. Our agency hopes that the software will evolve to be more efficient at allowing field updates. If not, our developers will need to find better ways of handling updates. Alternatively, our agency can rethink how Blaise 5 fits into our data analysis. Finally, we can also seek input from other organizations on how they effectively use table layouts with their editing strategies.

> back to Table of Contents

How Blaise 5 Improves Table Presentation

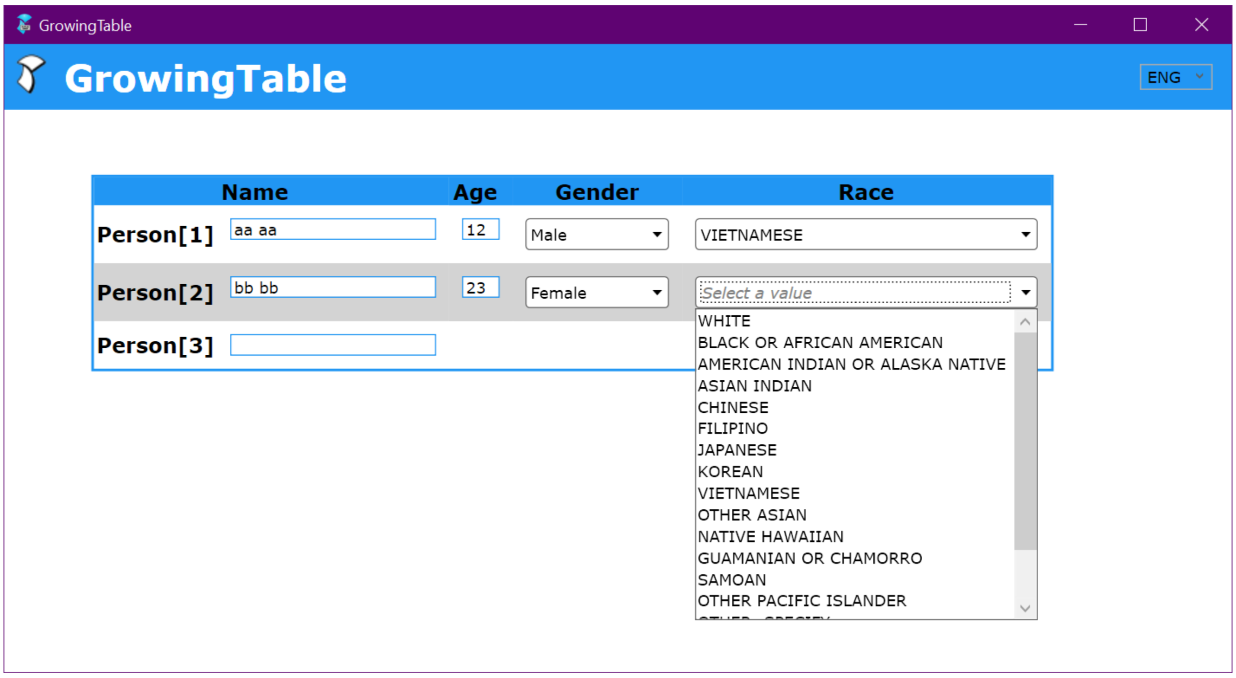
G J Boris Allan and Stéphane Ridoré, Westat

Presenter: Siu Chong Wan

As we look across versions of Blaise, the presentation of tables has continued to evolve. When we first encountered Blaise 3, one of our first questions concerned rosters and their implementation. The answer was Blaise tables, and we used tables successfully in both Blaise 3 and its successor, Blaise 4.
When Blaise 5 was introduced, we could use additional forms of tables, and one of the most important for our work is known as Option Tables. This paper discusses ways to work with tables in Blaise 5. We found that not only could we massage standard tables but also that we could extend our applications to various forms of option table by changing the nature of tables using the resource database (.blrd file).

1. The Basic Blaise Cross-Tabulation Table

The Blaise 5 samples have a variety of instruments that use tables, for example the (growing) roster-style table:



	Name	Age	Gender	Race
Person[1]	aa aa	12	Male	VIETNAMESE
Person[2]	bb bb	23	Female	Select a value
Person[3]				

WHITE
BLACK OR AFRICAN AMERICAN
AMERICAN INDIAN OR ALASKA NATIVE
ASIAN INDIAN
CHINESE
FILIPINO
JAPANESE
KOREAN
VIETNAMESE
OTHER ASIAN
NATIVE HAWAIIAN
GUAMANIAN OR CHAMORRO
SAMOAN
OTHER PACIFIC ISLANDER

This is a growing table, but that is incidental. We have embellished this table by adding on an extra column, the one labeled "Race." The race question is based on categories used in an earlier CAPI instrument in Blaise 4.8 and is used only as an example. Race questions can become complex (Blaise 4 was actually a SET OF), and so in the original interview, the CAPI interviewer had a good deal of extra information that was not read aloud to the respondent. Here is a version of the Blaise 4.8 CAPI translated to Blaise 5.13:

```
Race
"For this survey, Hispanic origins are not races. <newline> <newline>What is your
race? Please select one of the categories on this card. <newline><newline>ENTER ALL
```

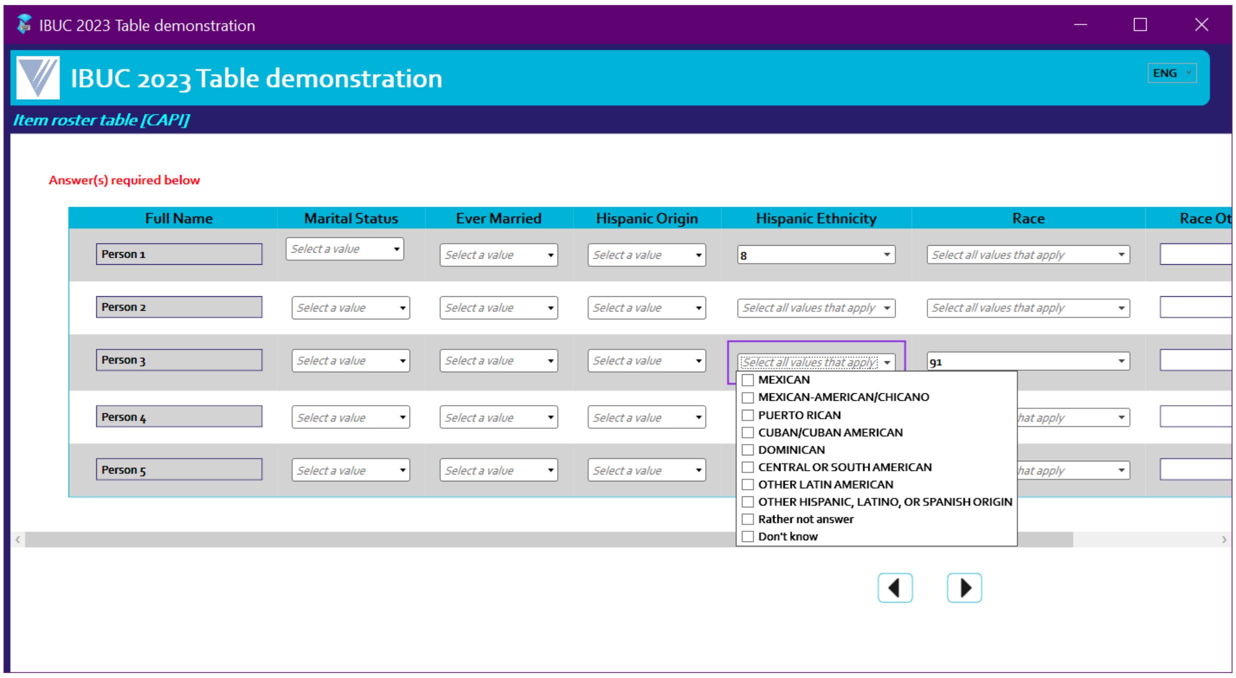
> back to Table of Contents

```
THAT APPLY."
/ "Race"
: (TRWhite    (1)  "WHITE" "BLANCA",
   TRAfrAmer  (2)  "BLACK OR AFRICAN AMERICAN" "NEGRA O AFROAMERICANA",
   TRAmInAlk  (3)  "AMERICAN INDIAN OR ALASKA NATIVE" "INDIA AMERICANA
```

One drawback in the Blaise 5 CAPI version above, therefore, is the lack of interviewer instructions we could have with Blaise 4.

1.1 A Simple Roster Version

A more extensive example of a roster in Blaise 5.13, based on the style of rosters in Blaise 4.8, is:



IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

ENG

Item roster table [CAPI]

Answer(s) required below

Full Name	Marital Status	Ever Married	Hispanic Origin	Hispanic Ethnicity	Race	Race Ot
Person 1	*Select a value*	*Select a value*	*Select a value*	8	*Select all values that apply*	
Person 2	*Select a value*	*Select a value*	*Select a value*	*Select all values that apply*	*Select all values that apply*	
Person 3	*Select a value*	*Select a value*	*Select a value*	*Select all values that apply*	91	
Person 4	*Select a value*	*Select a value*	*Select a value*		*hat apply*	
Person 5	*Select a value*	*Select a value*	*Select a value*		*hat apply*	

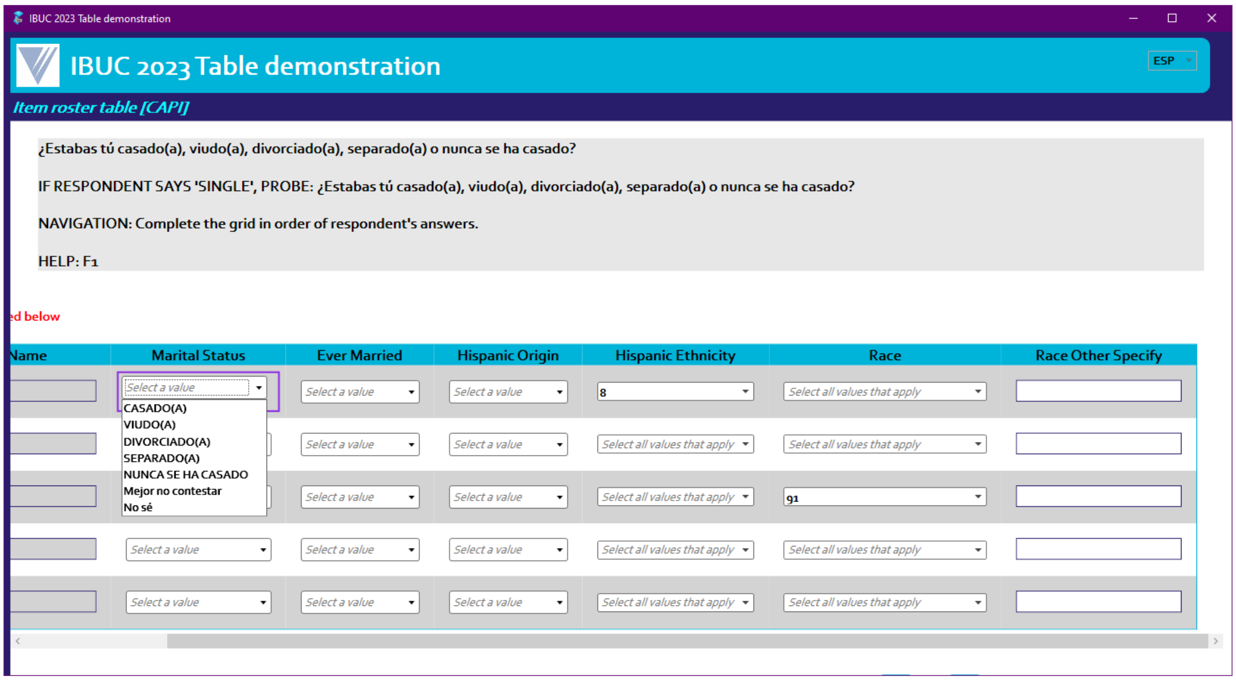
- ☐ MEXICAN
- ☐ MEXICAN-AMERICAN/CHICANO
- ☐ PUERTO RICAN
- ☐ CUBAN/CUBAN AMERICAN
- ☐ DOMINICAN
- ☐ CENTRAL OR SOUTH AMERICAN
- ☐ OTHER LATIN AMERICAN
- ☐ OTHER HISPANIC, LATINO, OR SPANISH ORIGIN
- ☐ Rather not answer
- ☐ Don't know

Note the horizontal scroll bar—very useful for wide tables—but there are no instructions for the list of types of Hispanic ethnicities.

1.2 The Simple Roster Version with an Extra Display

We can extend the roster with information for the interviewer and, in this case, a mixture of English and Spanish. The English parts of the question text (mainly in CAPS) are directions to the interviewer, and the Spanish text is to be read to the respondent. This is a fairly simple extension of the appropriate master page template:

> back to Table of Contents


IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

ESP

Item roster table [CAPI]

¿Estabas tú casado(a), viudo(a), divorciado(a), separado(a) o nunca se ha casado?

IF RESPONDENT SAYS 'SINGLE', PROBE: ¿Estabas tú casado(a), viudo(a), divorciado(a), separado(a) o nunca se ha casado?

NAVIGATION: Complete the grid in order of respondent's answers.

HELP: F1

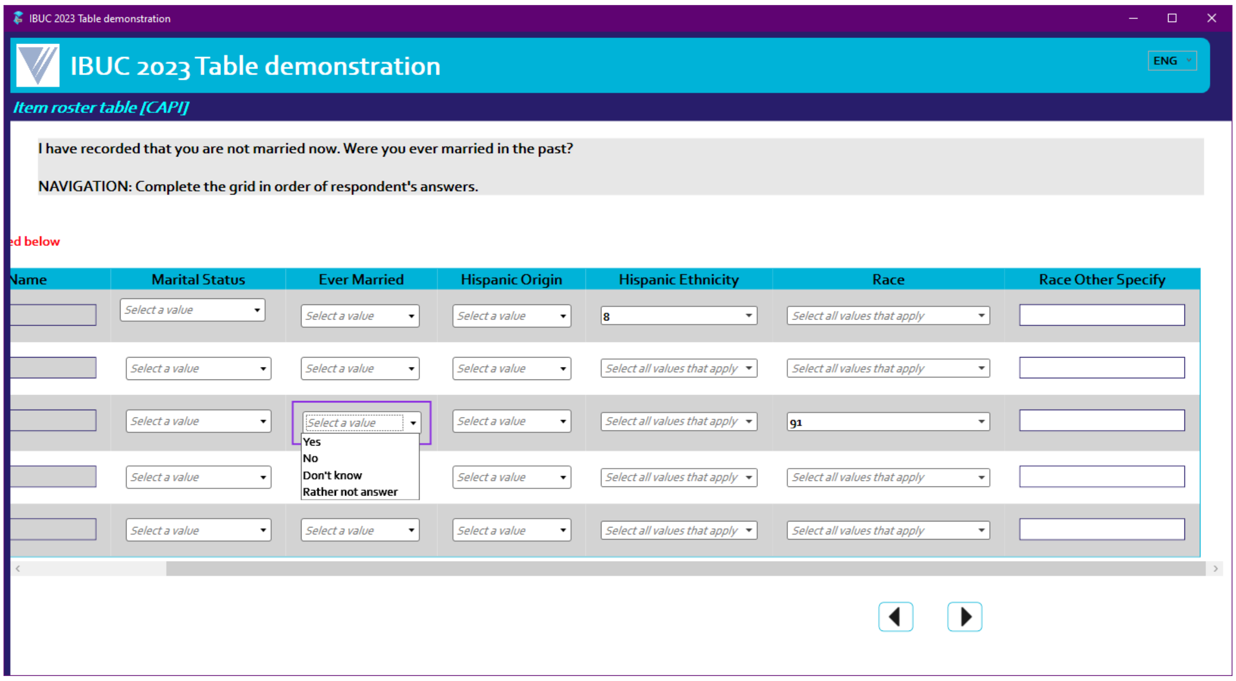
ed below

Name	Marital Status	Ever Married	Hispanic Origin	Hispanic Ethnicity	Race	Race Other Specify
	Select a value	*Select a value*	*Select a value*	8	*Select all values that apply*	

	CASADO(A)
	VIUDO(A)
	DIVORCIADO(A)
	SEPARADO(A)
	NUNCA SE HA CASADO
	Mejor no contestar
	No sé
		Select a value	*Select a value*	*Select all values that apply*	*Select all values that apply*	

		Select a value	*Select a value*	*Select all values that apply*	91	
	Select a value	*Select a value*	*Select a value*	*Select all values that apply*	*Select all values that apply*	
	Select a value	*Select a value*	*Select a value*	*Select all values that apply*	*Select all values that apply*	

A version totally in English is:


IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

ENG

Item roster table [CAPI]

I have recorded that you are not married now. Were you ever married in the past?

NAVIGATION: Complete the grid in order of respondent's answers.

ed below

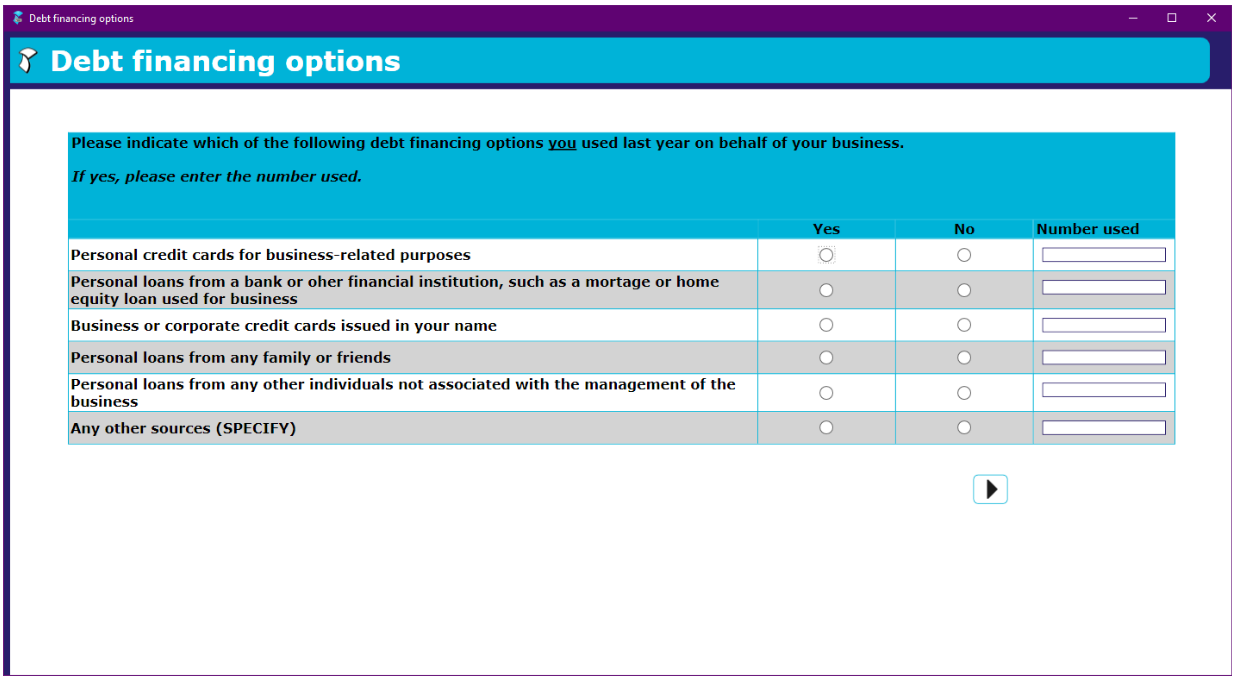
Name	Marital Status	Ever Married	Hispanic Origin	Hispanic Ethnicity	Race	Race Other Specify
	Select a value	*Select a value*	*Select a value*	8	*Select all values that apply*	
	Select a value	*Select a value*	*Select a value*	*Select all values that apply*	*Select all values that apply*	
	Select a value	*Select a value* Yes No Don't know Rather not answer	*Select a value*	*Select all values that apply*	91	
	Select a value		*Select a value*	*Select all values that apply*	*Select all values that apply*	
	Select a value	*Select a value*	*Select a value*	*Select all values that apply*	*Select all values that apply*	

Note the purple highlight to show which cell is being answered.

> back to Table of Contents

2. The Blaise Options Table

We have—for many studies—a particular need for a type of table that is not a roster-style table, that is, we need an options table. The Blaise 5 samples have an example that is a combination of the two table types (roster and options):



Debt financing options

Debt financing options

Please indicate which of the following debt financing options you used last year on behalf of your business.

If yes, please enter the number used.

	Yes	No	Number used
Personal credit cards for business-related purposes	○	○	
Personal loans from a bank or oher financial institution, such as a mortage or home equity loan used for business	○	○	
Business or corporate credit cards issued in your name	○	○	
Personal loans from any family or friends	○	○	
Personal loans from any other individuals not associated with the management of the business	○	○	
Any other sources (SPECIFY)	○	○	

Each subject topic (say, credit cards for business) has two associated questions: the first is a Yes/No question (two options) and the second is a simple numerical question. Whereas with, say, the question about Hispanic Origin in the roster (Yes/No with perhaps DK/RF) we used a drop-down list, in this options table example, we showed the two options in two columns. Options tables are used most frequently when we have many stimuli/topics and we want to have the options in a more convenient presentation.

2.1 Initial Views

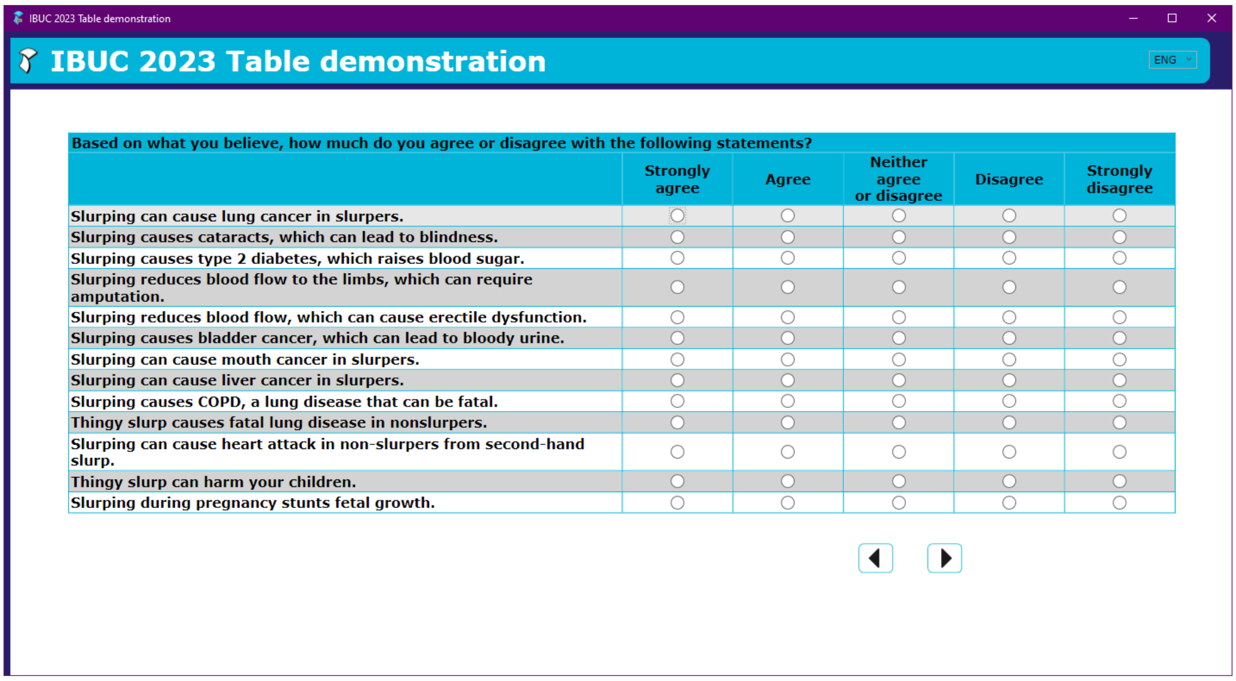
The rest of this paper investigates two types of options table (many versions are possible) and how we have extended functionality to improve understanding of what is required. Based on some of our methodologists' findings and some of our clients' requirements, we have extended functionality: all these changes are elaborations of basic templates extended to incorporate new features in the .blrd. The elaborations are dependent on these requirements.

Whereas rosters are principally a CAPI approach, options tables have applicability for CASI, such as a questionnaire on a browser.

2.1.1 The Standard Options Table

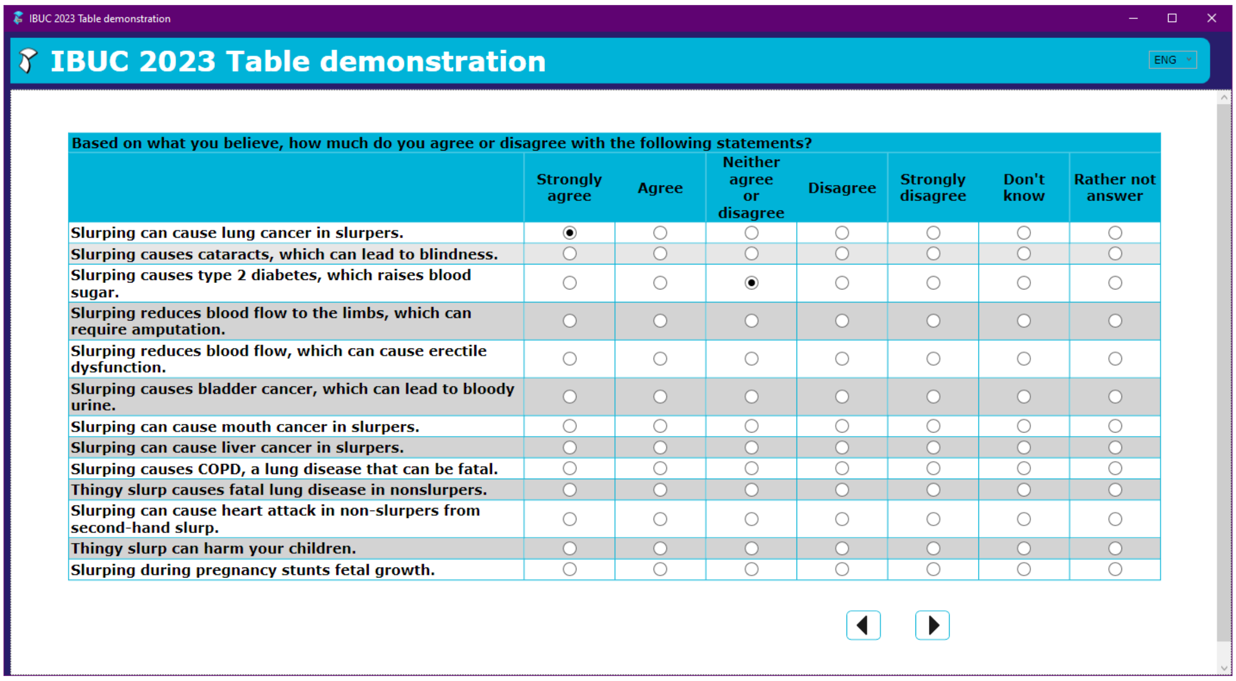
This is a standard display.

> back to Table of Contents



Based on what you believe, how much do you agree or disagree with the following statements?	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree
Slurping can cause lung cancer in slurpers.	○	○	○	○	○
Slurping causes cataracts, which can lead to blindness.	○	○	○	○	○
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	○	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	○	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	○	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	○	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○
Thingy slurp can harm your children.	○	○	○	○	○
Slurping during pregnancy stunts fetal growth.	○	○	○	○	○

If you click/touch/select some buttons on the above table, you can add data:

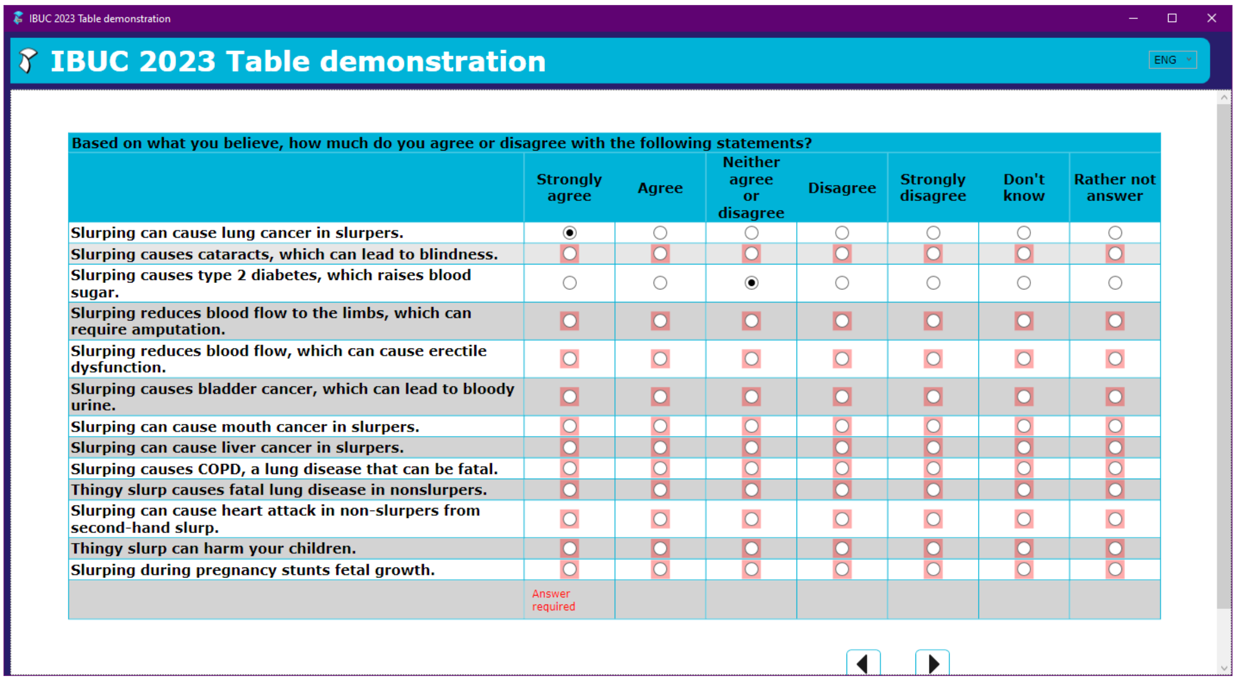


Based on what you believe, how much do you agree or disagree with the following statements?	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree	Don't know	Rather not answer
Slurping can cause lung cancer in slurpers.	●	○	○	○	○	○	○
Slurping causes cataracts, which can lead to blindness.	○	○	○	○	○	○	○
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	●	○	○	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	○	○	○	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	○	○	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	○	○	○	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	○	○	○	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○	○	○
Thingy slurp can harm your children.	○	○	○	○	○	○	○
Slurping during pregnancy stunts fetal growth.	○	○	○	○	○	○	○

You selected “Strongly agree” for lung cancer, skipped a row, and then selected “Disagree” for blood sugar. As soon as you leave an empty row and select the next row, the Blaise 5 normal nonresponse behavior (defined in Settings) kicks in. “Normal behavior” is to not show a nonresponse button until the next field on route receives focus.

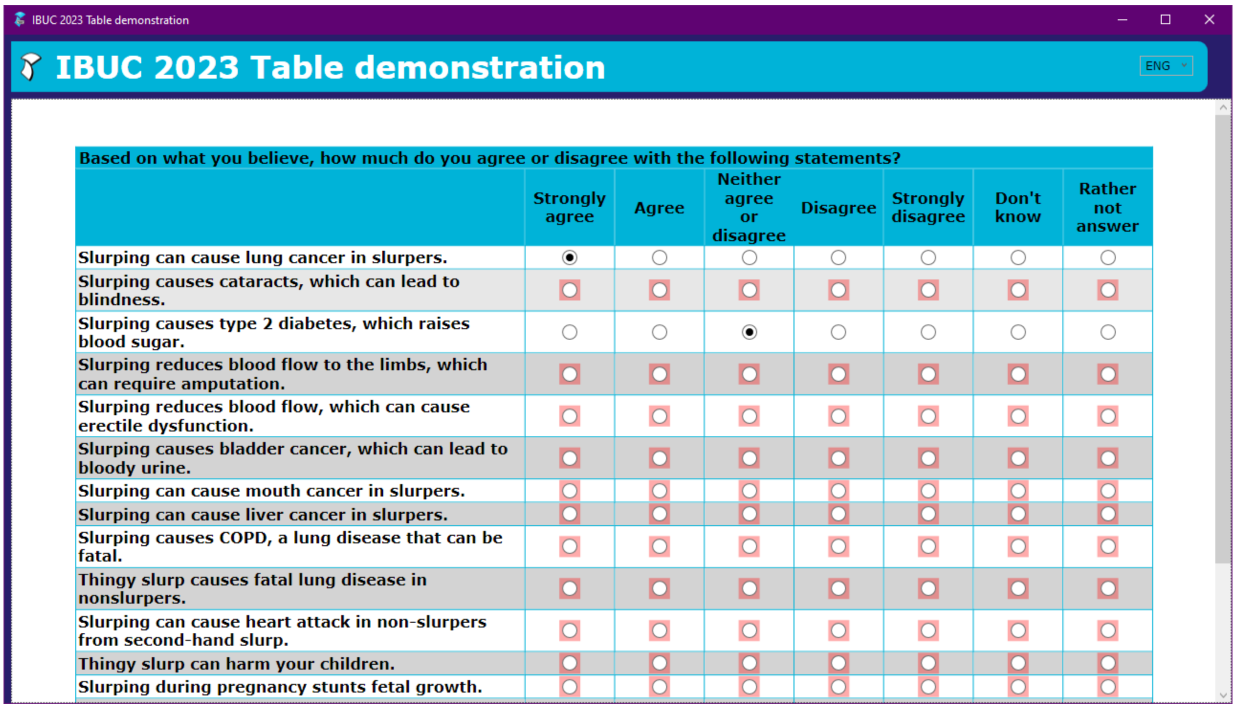
> back to Table of Contents

If you select the next-page button, then you find:



Based on what you believe, how much do you agree or disagree with the following statements?	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree	Don't know	Rather not answer
Slurping can cause lung cancer in slurpers.	●	○	○	○	○	○	○
Slurping causes cataracts, which can lead to blindness.	○	○	○	○	○	○	○
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	●	○	○	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	○	○	○	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	○	○	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	○	○	○	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	○	○	○	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○	○	○
Thingy slurp can harm your children.	○	○	○	○	○	○	○
Slurping during pregnancy stunts fetal growth.	○	○	○	○	○	○	○
	Answer required						

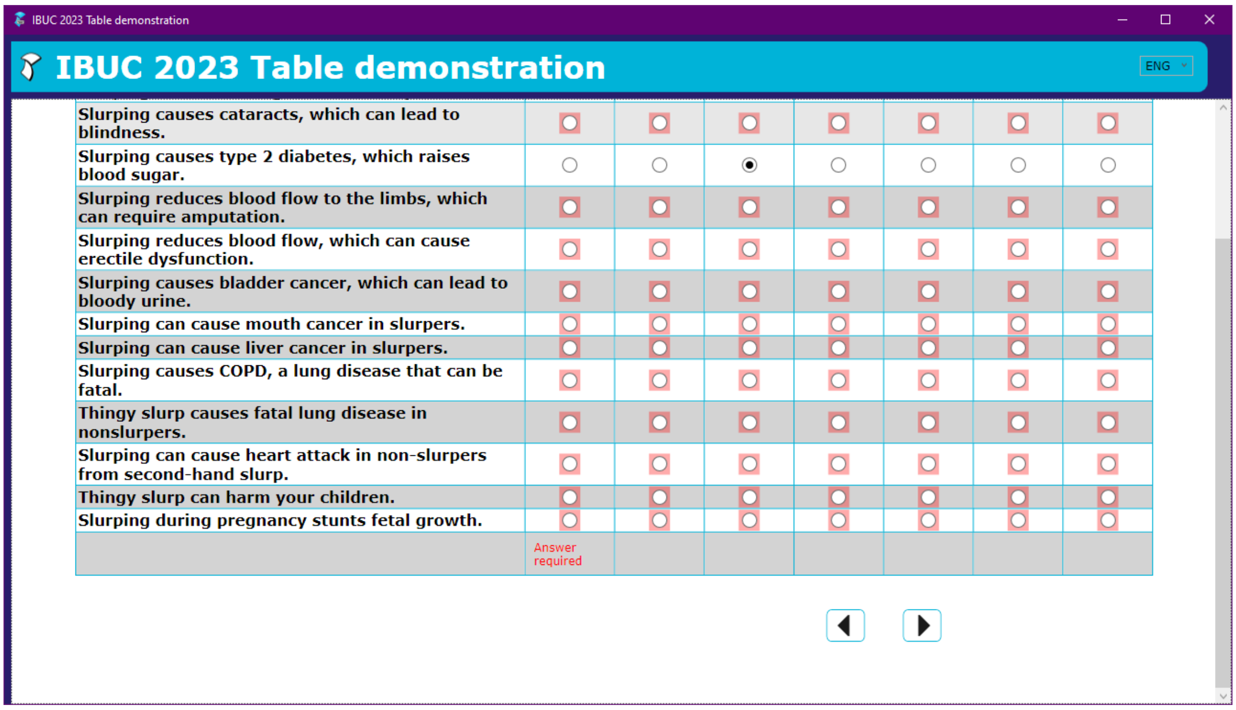
If we had any more rows, we might find that the back and next buttons would drop off the bottom of the view and disappear. There are studies for which the number of rows far exceeds the number we have above and, to show disappearing rows and buttons, we make the above window smaller:



Based on what you believe, how much do you agree or disagree with the following statements?	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree	Don't know	Rather not answer
Slurping can cause lung cancer in slurpers.	●	○	○	○	○	○	○
Slurping causes cataracts, which can lead to blindness.	○	○	○	○	○	○	○
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	●	○	○	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	○	○	○	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	○	○	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	○	○	○	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	○	○	○	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○	○	○
Thingy slurp can harm your children.	○	○	○	○	○	○	○
Slurping during pregnancy stunts fetal growth.	○	○	○	○	○	○	○

> back to Table of Contents

There is a scroll bar so that we can see lower rows, but if the rows and navigation buttons are shown, then we cannot see the question or the column headings:

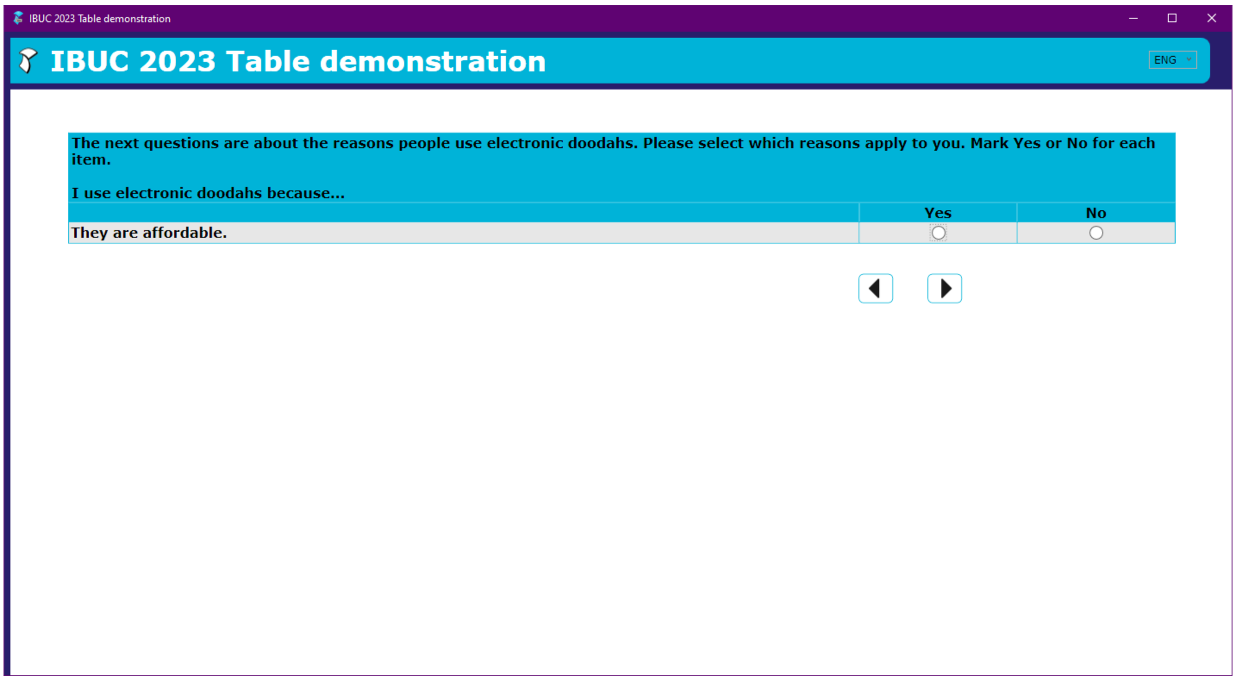

IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

Slurping causes cataracts, which can lead to blindness.	○	○	○	○	○	○	○
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	●	○	○	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	○	○	○	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	○	○	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	○	○	○	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	○	○	○	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○	○	○
Thingy slurp can harm your children.	○	○	○	○	○	○	○
Slurping during pregnancy stunts fetal growth.	○	○	○	○	○	○	○
	Answer required						

2.1.2 The Growing Options Table

We have already come across an expanding (or growing) table, and here is an example based on a study's requirements:


IBUC 2023 Table demonstration

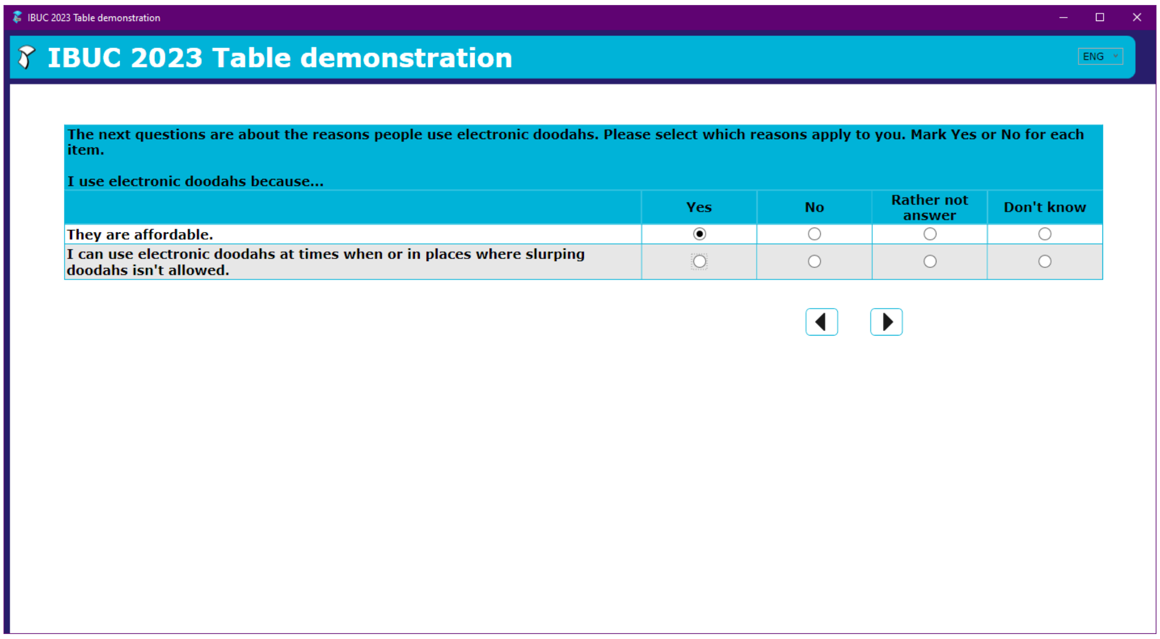
IBUC 2023 Table demonstration

The next questions are about the reasons people use electronic doodahs. Please select which reasons apply to you. Mark Yes or No for each item.

I use electronic doodahs because...

	Yes	No
They are affordable.	○	○

Select buttons for the first rows, that is, select a button then select next, and so on.



IBUC 2023 Table demonstration

The next questions are about the reasons people use electronic doodahs. Please select which reasons apply to you. Mark Yes or No for each item.

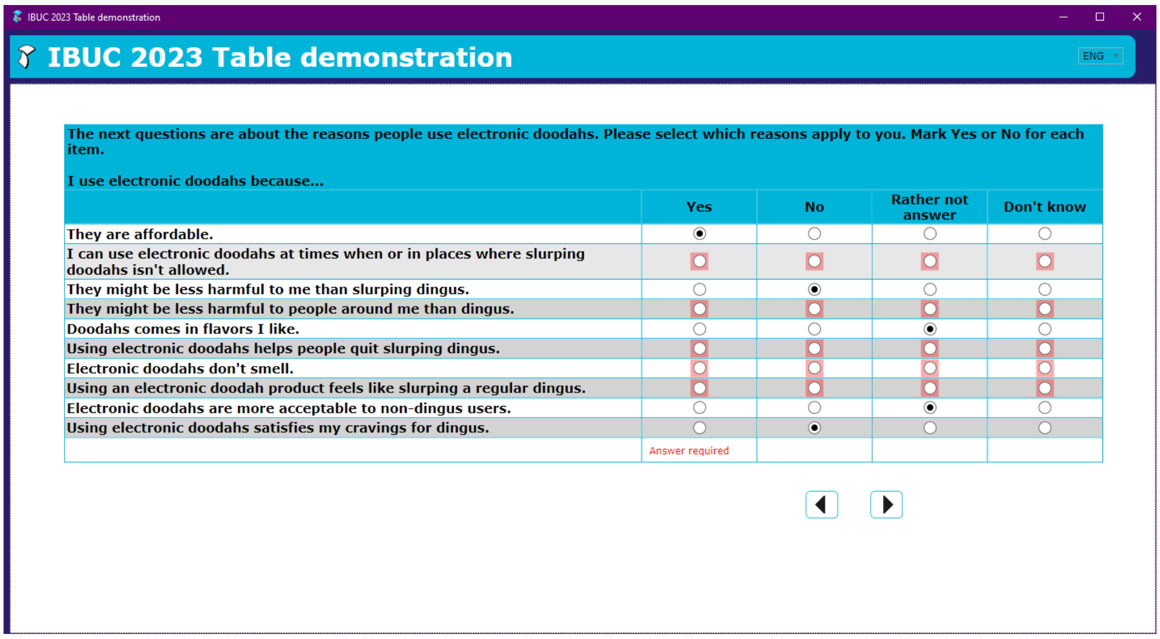
I use electronic doodahs because...

	Yes	No	Rather not answer	Don't know
They are affordable.	●	○	○	○
I can use electronic doodahs at times when or in places where slurping doodahs isn't allowed.	○	○	○	○

Because of the way that the appearance of rows is controlled in the Blaise code:

```
ROW1
IF ROW1 <> EMPTY OR ROW1.IsVisited = 1 THEN
  ROW2
ENDIF
```

Our client wanted us to allow new rows to appear, even though an answer was required. We were not to show any indication that answers were required until the last row was reached and the next page was selected.



IBUC 2023 Table demonstration

The next questions are about the reasons people use electronic doodahs. Please select which reasons apply to you. Mark Yes or No for each item.

I use electronic doodahs because...

	Yes	No	Rather not answer	Don't know
They are affordable.	●	○	○	○
I can use electronic doodahs at times when or in places where slurping doodahs isn't allowed.	○	○	○	○
They might be less harmful to me than slurping dingus.	○	●	○	○
They might be less harmful to people around me than dingus.	○	○	○	○
Doodahs comes in flavors I like.	○	○	●	○
Using electronic doodahs helps people quit slurping dingus.	○	○	○	○
Electronic doodahs don't smell.	○	○	○	○
Using an electronic doodah product feels like slurping a regular dingus.	○	○	○	○
Electronic doodahs are more acceptable to non-dingus users.	○	○	●	○
Using electronic doodahs satisfies my cravings for dingus.	○	●	○	○
	Answer required			

> back to Table of Contents

This goes beyond the normal behavior of DK/RF.

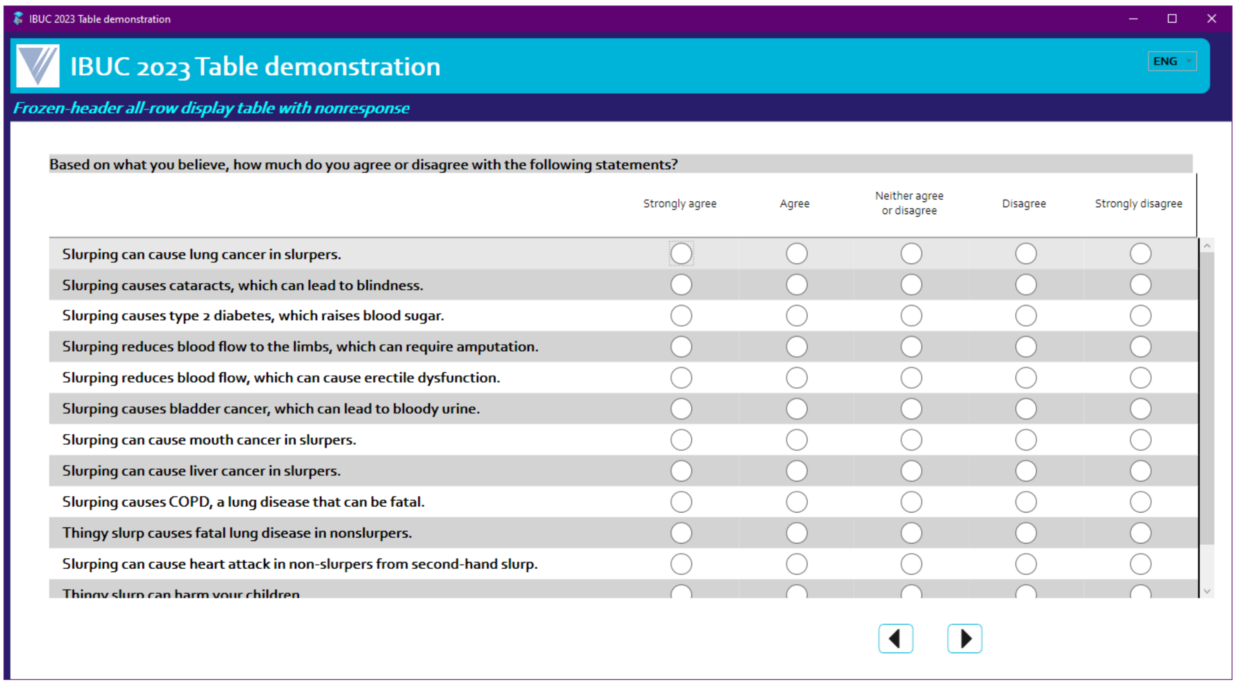
2.2 Elaborated Views

Our client had two major concerns:

- The client did not want either the navigation buttons or the table headings (question and option headings) to disappear when scrolling. Their preference was to have a frozen header such as existed in Excel.
- The client did not want the answer-required columns to appear until the respondent had gone through the complete table.

2.2.1 The Standard Options Table Version

After some design work that involved paying a good deal of attention to the resource database, we have an approach that satisfies our original client and is being evaluated by others.



IBUC 2023 Table demonstration

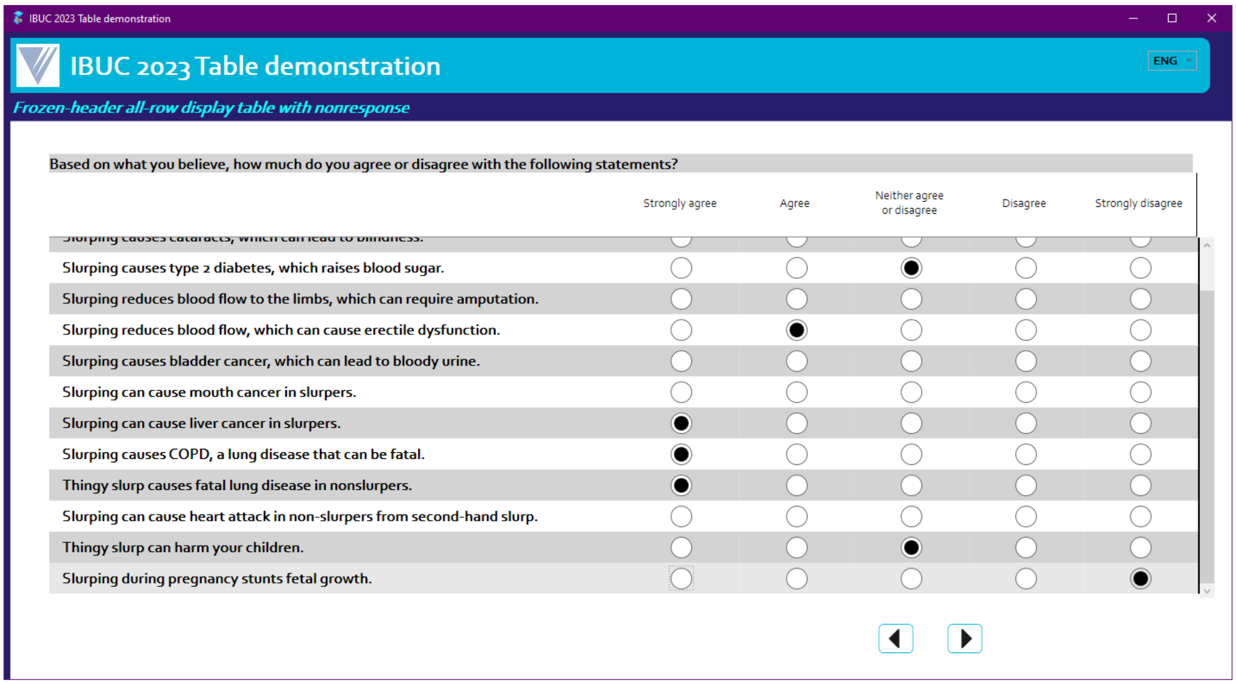
Frozen-header all-row display table with nonresponse

Based on what you believe, how much do you agree or disagree with the following statements?

	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree
Slurping can cause lung cancer in slurpers.	○	○	○	○	○
Slurping causes cataracts, which can lead to blindness.	○	○	○	○	○
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	○	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	○	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	○	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	○	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○
Thingy slurp can harm your children	○	○	○	○	○

We answered some, but not all, questions because the specification said that the respondent should not be troubled by missing an answer:

> back to Table of Contents


IBUC 2023 Table demonstration

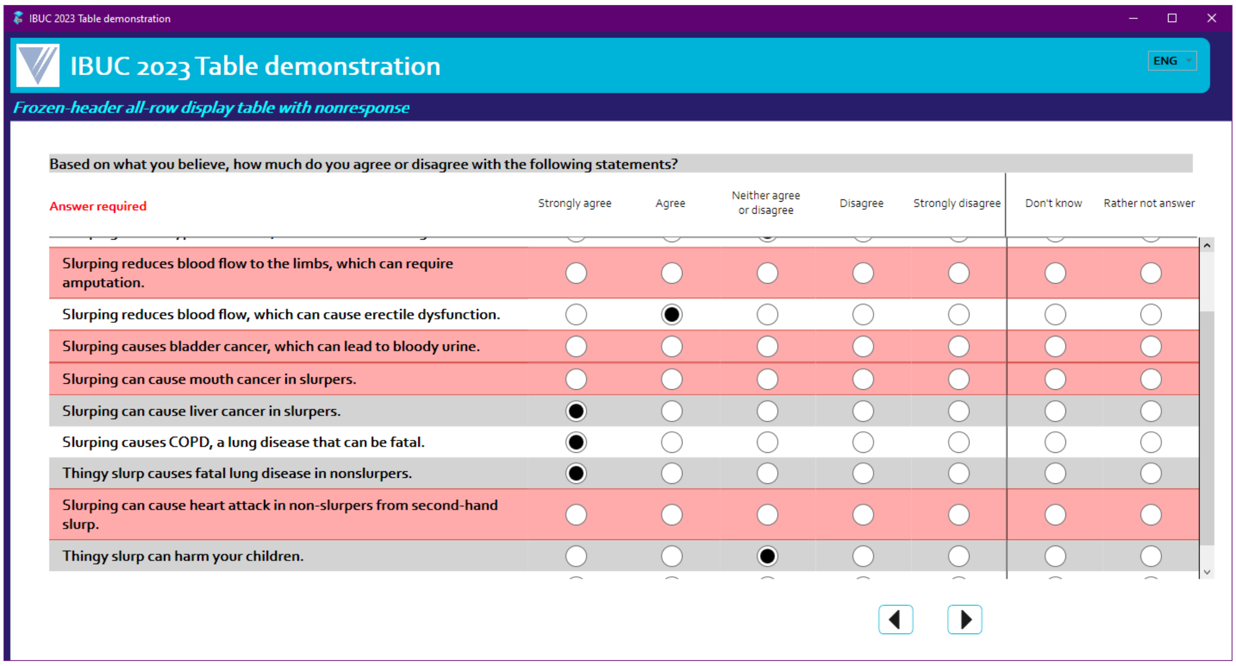
IBUC 2023 Table demonstration

Frozen-header all-row display table with nonresponse

Based on what you believe, how much do you agree or disagree with the following statements?

	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree
Slurping causes cataracts, which can lead to blindness.	○	○	○	○	○
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	●	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	●	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	●	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	●	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	●	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○
Thingy slurp can harm your children.	○	○	●	○	○
Slurping during pregnancy stunts fetal growth.	○	○	○	○	●

However, respondents needed to answer all the questions, and so when they select the next-page button:


IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

Frozen-header all-row display table with nonresponse

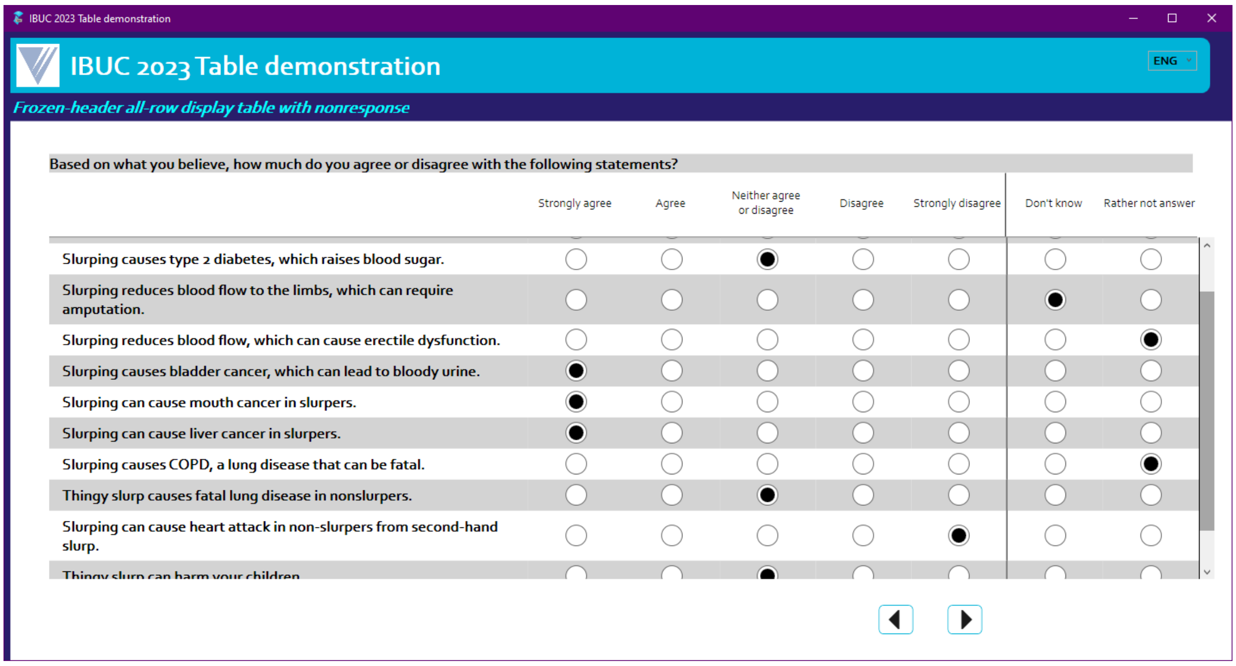
Based on what you believe, how much do you agree or disagree with the following statements?

Answer required

	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree	Don't know	Rather not answer
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○	○	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	●	○	○	○	○	○
Slurping causes bladder cancer, which can lead to bloody urine.	○	○	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	○	○	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	●	○	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	●	○	○	○	○	○	○
Thingy slurp causes fatal lung disease in nonslurpers.	●	○	○	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	○	○	○
Thingy slurp can harm your children.	○	○	●	○	○	○	○

> back to Table of Contents

The rows without answers are highlighted, and the respondent has to answer all the missing rows:



IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

ENG

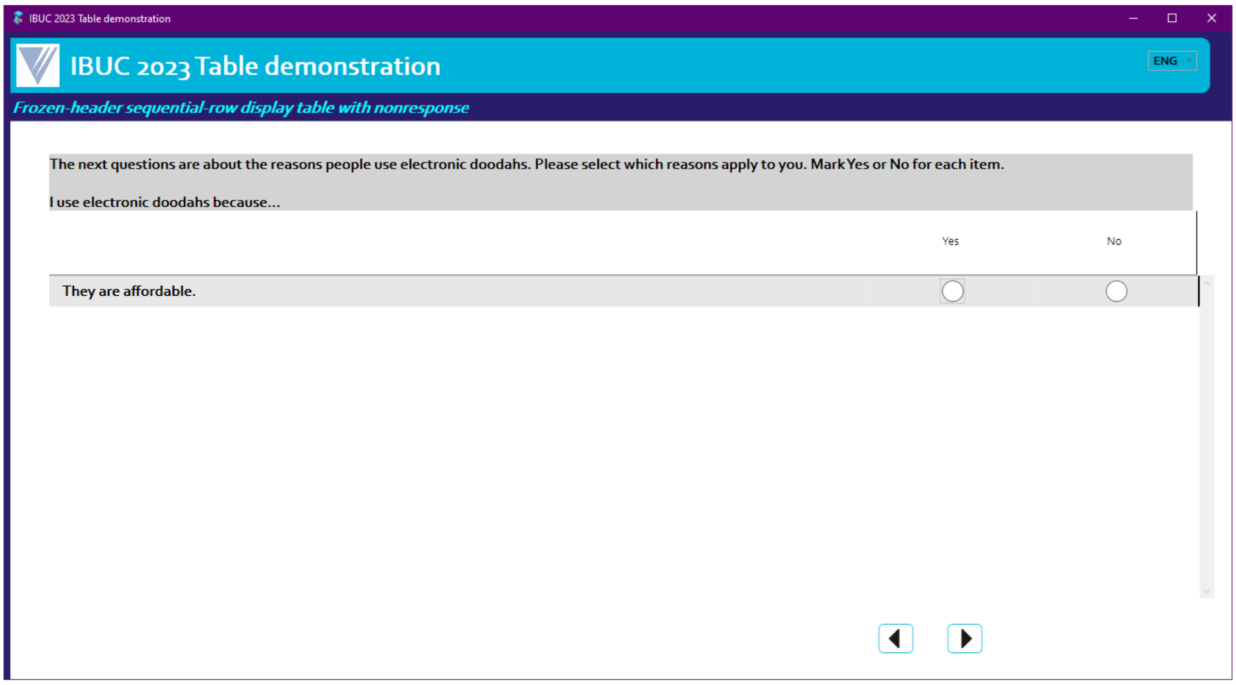
Frozen-header all-row display table with nonresponse

Based on what you believe, how much do you agree or disagree with the following statements?

	Strongly agree	Agree	Neither agree or disagree	Disagree	Strongly disagree	Don't know	Rather not answer
Slurping causes type 2 diabetes, which raises blood sugar.	○	○	●	○	○	○	○
Slurping reduces blood flow to the limbs, which can require amputation.	○	○	○	○	○	●	○
Slurping reduces blood flow, which can cause erectile dysfunction.	○	○	○	○	○	○	●
Slurping causes bladder cancer, which can lead to bloody urine.	●	○	○	○	○	○	○
Slurping can cause mouth cancer in slurpers.	●	○	○	○	○	○	○
Slurping can cause liver cancer in slurpers.	●	○	○	○	○	○	○
Slurping causes COPD, a lung disease that can be fatal.	○	○	○	○	○	○	●
Thingy slurp causes fatal lung disease in nonslurpers.	○	○	●	○	○	○	○
Slurping can cause heart attack in non-slurpers from second-hand slurp.	○	○	○	○	●	○	○
Thingy slurp can harm your children	○	○	●	○	○	○	○

2.2.2 The Growing Options Table Version

We start here:



IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

ENG

Frozen-header sequential-row display table with nonresponse

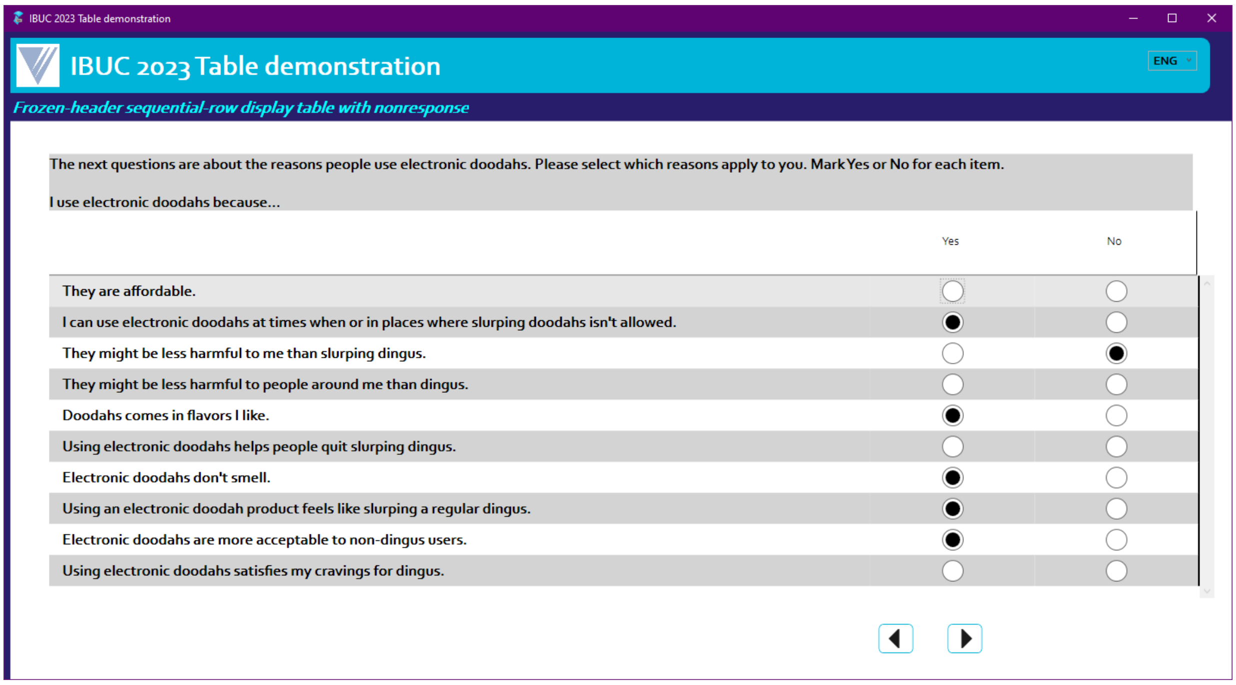
The next questions are about the reasons people use electronic doodahs. Please select which reasons apply to you. Mark Yes or No for each item.

I use electronic doodahs because...

	Yes	No
They are affordable.	○	○

The respondent is not forced to answer a question:

> back to Table of Contents


IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

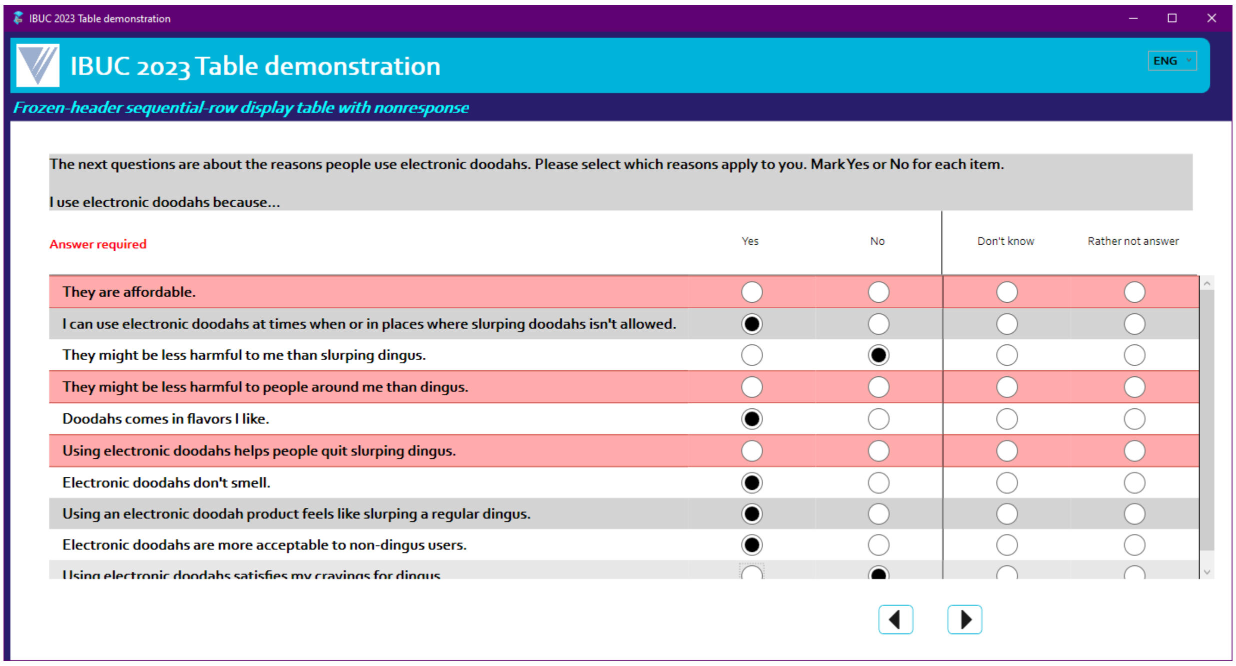
Frozen-header sequential-row display table with nonresponse

The next questions are about the reasons people use electronic doodahs. Please select which reasons apply to you. Mark Yes or No for each item.

I use electronic doodahs because...

	Yes	No
They are affordable.	○	○
I can use electronic doodahs at times when or in places where slurping doodahs isn't allowed.	●	○
They might be less harmful to me than slurping dingus.	○	●
They might be less harmful to people around me than dingus.	○	○
Doodahs comes in flavors I like.	●	○
Using electronic doodahs helps people quit slurping dingus.	○	○
Electronic doodahs don't smell.	●	○
Using an electronic doodah product feels like slurping a regular dingus.	●	○
Electronic doodahs are more acceptable to non-dingus users.	●	○
Using electronic doodahs satisfies my cravings for dingus.	○	○

However, at the end of the questions and when the next-page button is selected:


IBUC 2023 Table demonstration

IBUC 2023 Table demonstration

Frozen-header sequential-row display table with nonresponse

The next questions are about the reasons people use electronic doodahs. Please select which reasons apply to you. Mark Yes or No for each item.

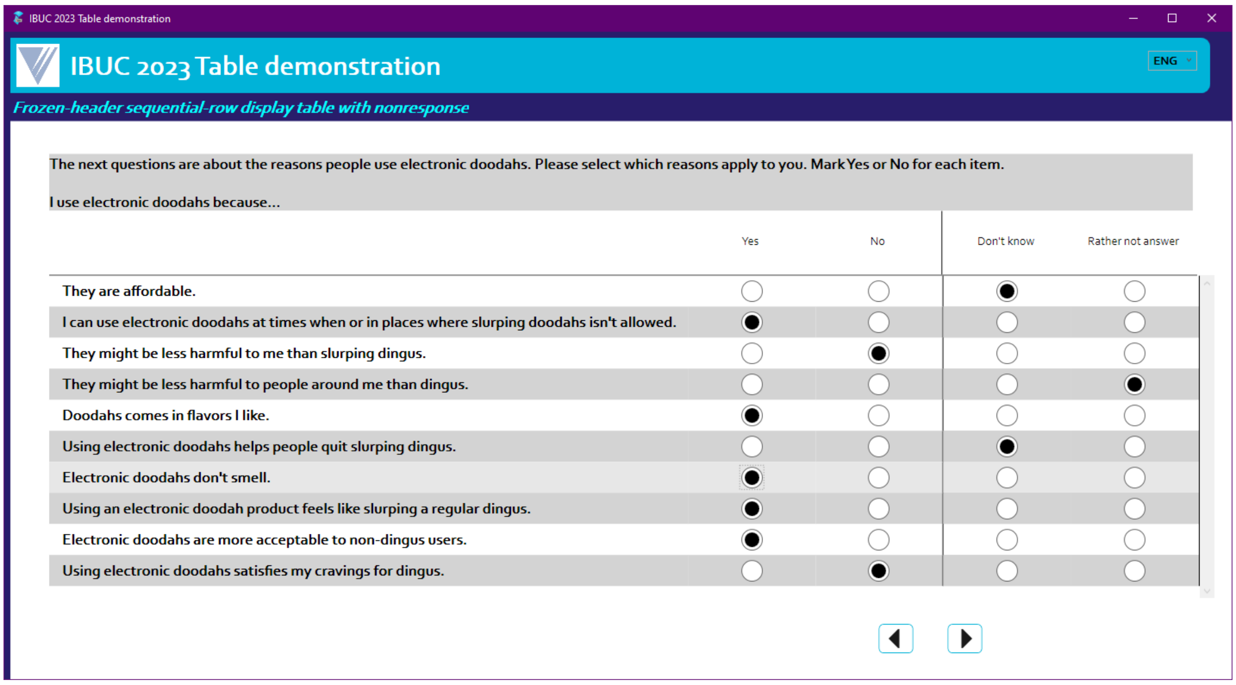
I use electronic doodahs because...

Answer required

	Yes	No	Don't know	Rather not answer
They are affordable.	○	○	○	○
I can use electronic doodahs at times when or in places where slurping doodahs isn't allowed.	●	○	○	○
They might be less harmful to me than slurping dingus.	○	●	○	○
They might be less harmful to people around me than dingus.	○	○	○	○
Doodahs comes in flavors I like.	●	○	○	○
Using electronic doodahs helps people quit slurping dingus.	○	○	○	○
Electronic doodahs don't smell.	●	○	○	○
Using an electronic doodah product feels like slurping a regular dingus.	●	○	○	○
Electronic doodahs are more acceptable to non-dingus users.	●	○	○	○
Using electronic doodahs satisfies my cravings for dingus.	○	●	○	○

The respondent has to answer all the questions before continuing to the next page.

> back to Table of Contents



Frozen-header sequential-row display table with nonresponse

The next questions are about the reasons people use electronic doodahs. Please select which reasons apply to you. Mark Yes or No for each item.

I use electronic doodahs because...

	Yes	No	Don't know	Rather not answer
They are affordable.	○	○	●	○
I can use electronic doodahs at times when or in places where slurping doodahs isn't allowed.	●	○	○	○
They might be less harmful to me than slurping dingus.	○	●	○	○
They might be less harmful to people around me than dingus.	○	○	○	●
Doodahs comes in flavors I like.	●	○	○	○
Using electronic doodahs helps people quit slurping dingus.	○	○	●	○
Electronic doodahs don't smell.	●	○	○	○
Using an electronic doodah product feels like slurping a regular dingus.	●	○	○	○
Electronic doodahs are more acceptable to non-dingus users.	●	○	○	○
Using electronic doodahs satisfies my cravings for dingus.	○	●	○	○

3. Conclusions

There are many powerful tools in Blaise 5, and many of them reside in the resource database. There are so many tools that we are only now discovering some of them—and many are not well documented. It is only by pushing the envelope that we discover some of these tools.

For example, we use a Boolean server variable named AllVisited that is true when all rows of a table have been visited. We started with:

```
ServerVariables.SetBoolean('AllVisited', Page.Fields.All(item.isvisited))
```

However, sometimes we had cases where we could see that all fields on a page were visited, but AllVisited was never true. On further investigation, we found that in the "items" were Field References that were never visited. After consulting with StatNeth, we added another condition to the All() directive:

```
ServerVariables.SetBoolean('AllVisited', Page.Fields.All(item.IsVisited OR
item.Origin = Reference))
```

That is, there was a way to distinguish between types of fields by their origin, and this feature had existed for some time (but was unpublicized). At the time of writing, the origin feature is still undocumented.

The main takeaway from our adventures with tables is that many things are possible and, in Blaise 5, those many things are often tucked away in the resource database. We always try to find a generic solution to a specific problem because we want to apply the solution to a range of instances. In the case of tables, by using tools in the resource database, we found generic solutions.

> back to Table of Contents

Video Interviewing: An Overview

Andrew Hupp, University of Michigan

1. Background

1.1 Why Video-Mediated Interviews?

When face-to-face data collection is required, video-mediated interviewing appears to be an effective alternative to in-person data collection, since it's also face to face. It allows interviewers to help with difficult response tasks, like cognitive assessments. It enables data to be collected from members of remote populations, like those deployed in the military or those with security or privacy concerns. It reduces or eliminates interviewer travel costs. It promotes completion (Hupp et al., 2021) and reduces straightlining when compared with self-administration (Conrad et al., 2023), and it promotes the same levels of rapport between the respondent and interviewer that are observed in in-person interviews (Sun et al., 2021).

1.2 Respondent Considerations

But not all respondents have access to video communication, potentially leading to coverage errors (Schober et al., 2020). To do a video interview, one needs a stable internet connection and a device with a working camera and microphone, and they must be willing (Schober et al., 2023) and comfortable enough with using video. In 2021, Pew reported that 81% of U.S. adults have used video to talk with others and that those with more education are likely to make frequent video calls. These data are from early in the COVID-19 pandemic and presumably have increased since then.

On the other hand, having video as a communication mode might improve access in some cases, like for those who might be too shy to ask an interviewer to speak up but could easily turn up the volume in a video interview to better hear the question.

1.3 Early Visions of Video Communication

Video communication was first conceptualized in the 1870s by Bell Labs. The first video call (by Bell Labs), was a one-way audio and video call that President Hoover made to New York in the late 1920s.

The 1930s saw early prototypes of two-way calls in Germany and New York. Bell Labs debuted the picture phone at the 1964 World's Fair.

Video communication began appearing in popular culture around this time. In the 1968 movie *2001: A Space Odyssey*, Dr. Heywood Floyd (in a space station) goes into the picturephone booth and inserts his credit card to pay for the video call to his daughter, who is on the Earth. In the early 1960s, prior to the World's Fair, *The Jetsons* debuted, with video calls envisioned as one-to-one communication, much like a telephone call. In season 1, episode 10, they anticipated telemedicine, with Jane calling the doctor to evaluate Elroy, who says he is too sick to go to school.

1.4 Current Use

There is interest in the use of video in data collection operations. Several projects in the United Kingdom, Europe, Australia, and the United States have incorporated video in some capacity in their recent data collections. Research Strand 3 of the Survey Futures initiative in the United Kingdom is dedicated to investigating video interviewing further. There is an international video interviewing special interest group through the National Centre for Research Methods Survey Data Collection Network in the United Kingdom. There is an upcoming special issue of the journal *Methods, Data, Analyses* on the topic of

> back to Table of Contents

video interviewing for collecting survey data that is scheduled for publication in early 2025, and there was an American Association for Public Opinion Research webinar on video survey interviews in 2022.

1.5 Vocabulary

What was originally called videoconferencing, is now more commonly referred to as video communication, video calls, or video meetings.

I do *not* advocate the use of four-letter acronyms with a "C" for "computer assisted," like CAVI for computer-assisted video interview. All video communication involves a computer, which *mediates* the communication more than *assists* an interviewer. These acronyms made sense when the shift from paper to computerization occurred. The assumption now should be that a computer is being used.

I distinguish live video interviews from a mode in which video recordings of interviewers reading questions are embedded in online questionnaires.

I use live video interviews to mean live, two-way communication and use in person for what has been historically referred to as face to face. I do this, since both modes are face to face and don't provide enough detail in describing the interaction.

2. Design and Implementation

2.1 Sample and Recruitment

One potential option for recruiting respondents is cold calling, although there are likely challenges with assembling a sampling frame. There are also questions as to how effective this recruitment method might be. Unsolicited contact, like inviting a household via an address-based sample, is at the moment unlikely to be productive (Hupp et al., 2021). The invitation to video interviews needs to be in another mode, like email, in person, or telephone.

A second option is having the respondent self-schedule in advance. It's a good idea to obtain contact information, such as a phone number or email address, so it can be used to remind the respondent of when their appointment is, and it provides the interviewer with other methods in which to contact the respondent if they are having technical issues when trying to join or during the interview.

The third option is an on-demand approach, where there are interviewers on standby waiting to do a video interview. The American National Election Studies tried this during its 2020 data collection and found that it's feasible but inefficient.

Video interviews seem well suited for longitudinal panel studies in which there is already trust with the survey organization, and the possibility to instruct a respondent on the use of video and to check or test connections during earlier in-person visits.

2.2 Scheduling

You'll want to develop a strategy for reminding the respondent of their appointment. Conrad and colleagues (2023) implemented a strategy where the respondent was first reminded the day prior to the appointment, then 2 hours prior to the appointment time on the day of the interview, and 5 minutes after the appointment time if the respondent had not joined the meeting.

The respondent received either an email, text message, or both depending on the information they provided when scheduling the appointment. Each message contained the meeting link and a link to reschedule the appointment. The 5-minute late message was triggered by the interviewer from within the

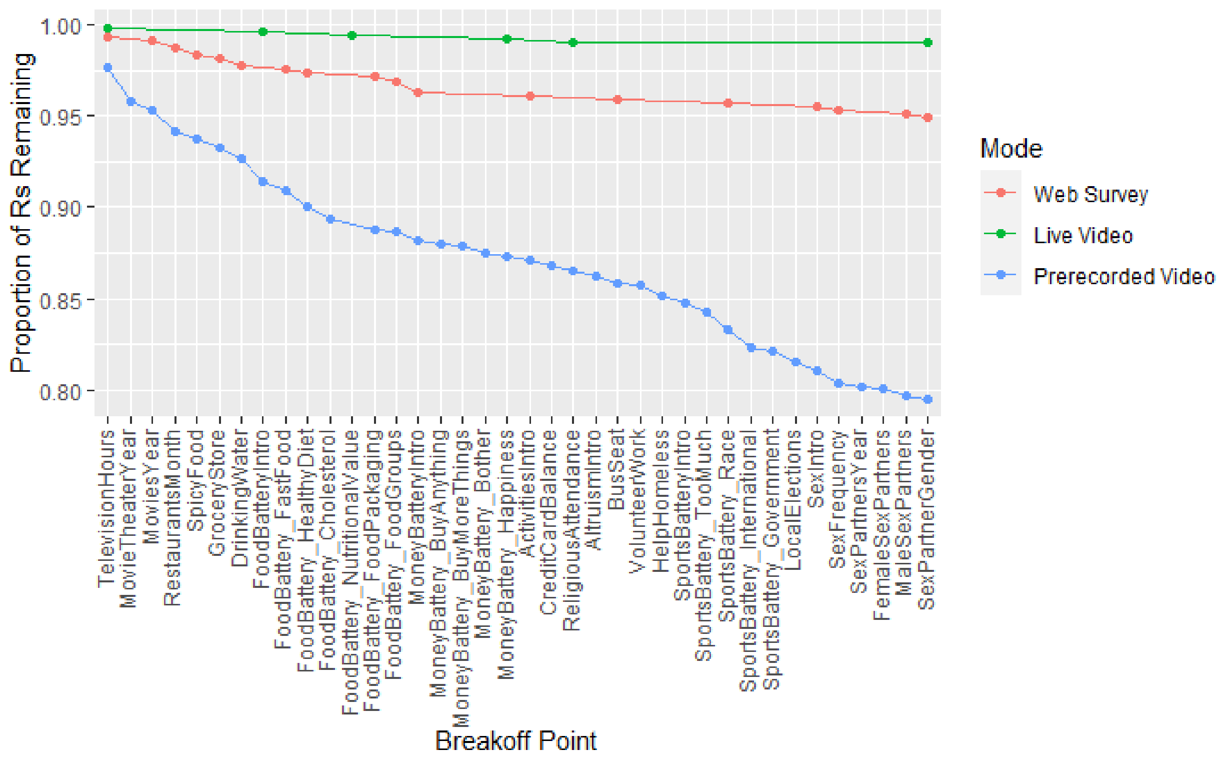
> back to Table of Contents

management system. The message mentioned that the interviewer would remain in the meeting for 10 minutes. Based on evidence from studies in other modes and our own experience with video interviews, we suspect that the scheduling approach may work better for participants who have already agreed to participate in an ongoing study than for newly invited sample members to a cross-sectional study.

2.3 Breakoffs

We have evidence that those who start a video interview are likely to finish (Hupp et al., 2021). Figure 1 depicts breakoffs using data from the Conrad et al. (2023) study. The x-axis is the question where the breakoff occurred, and the y-axis is the proportion of cases remaining. The green line represents live video interviews, the red line represents web surveys, and the blue line represents prerecorded video—a web survey with a video of an interviewer asking the questions.

Figure 1. Breakoffs



We see that once participants were recruited into the live video mode, there were very few breakoffs, especially compared with the two types of web surveys, perhaps due to the presence of a live interviewer. This is encouraging, although those live video breakoffs that did occur were due to technical issues (not present in other modes).

3. Data Quality

Looking at data quality, there are two published studies that have examined this. The first is a lab study conducted by Endres and colleagues (2022) that compared data quality in live video, web, and in-person interviews. The second is field study by Conrad and colleagues (2023) that compared data quality between live video, web, and prerecorded video.

> back to Table of Contents

Both studies found that most satisficing behaviors are less common in live video than in a web survey, with rounding being the exception, much like in in-person interviewing where there is greater time pressure than during a self-administered, relatively asynchronous web survey.

Table 1. Data Quality

Data Quality Measure	Endres et al. (2022)	Conrad et al. (2023)
Length of open responses	Live video > Web	
Straightlining	Live video (marginally) < Web	Live video < Web
Missing data	Live video < Web	Live video < Web
Rounding		Live video > Web
Disclosure	Live video < Web	Live video < Web

Endres and colleagues (2022) found no data quality differences between in-person and live video interviews. Conrad and colleagues' (2023) findings are analogous to published comparisons of in-person and web.

- *Straightlining*: is less prevalent in in-person interviews than in web surveys (Heerwegh & Loosveldt, 2008).
- *Disclosing sensitive information*: there is more socially desirable responding in in-person interviews than in web surveys (Heerwegh, 2007).
- *Rounding*: is greater in in-person interviews than in web surveys (Liu & Wang, 2015); this is attributed to there being greater time pressure in in-person interviews than in web surveys.

4. Interviewer Effects

It's possible that as much as interviewers in in-person interviews are known to introduce error variance, that is, to create interviewer effects, live video interviewers may introduce interviewer error. West and colleagues (2022) examined this and report that interviewer variance is low overall, with IICs less than 0.02. They didn't have an in-person group to compare to, but this suggests that live video interviewers introduce no more variance than is typical in an in-person interview.

5. Discussion

Scheduling a meeting seems to be the norm, compared with the cold calling model. There are options depending on the project design. The respondent can be offered a self-schedule option where they are sent a link and they select a time that works for them, or have an interviewer schedule a video interview at the conclusion of a prior in-person interview.

Video interviewing needs to be easy for the respondent. Implementing a "one-click" solution where it utilizes the browser rather than having the respondent download or install specific software apps they may be unfamiliar with will be key. Having one platform that is browser based rather than having the respondent choose the platform they are the most comfortable with will also limit the burden on the survey organization from having to support multiple platforms and purchasing operating system–specific equipment.

Video is more likely to succeed when it is offered as a choice in a single interview, rather than the lone choice, or as a follow-up to an in-person interview, like in the American National Election Studies.

Some studies have been screen sharing content for things like showcards, but there are products where other content can be shared, such as sharing a self-administered questionnaire with the respondent to allow them some privacy when answering sensitive questions. More methodological work is needed to

> back to Table of Contents

understand how various video interviewing features—things like turning the camera off or turning survey control over to the respondent when responding to sensitive questions—impact implementation and data quality in video interviews.

6. References

Conrad, F.G., Schober M.F., Hupp A.L., West B.T., Larsen K.M., Ong A.R., & Wang T. (2023). Video in survey interviews: Effects on data quality and respondent experience. *Methods, Data, Analyses, 17*(2) 135–170. https://doi.org/10.12758/mda.2022.13

Endres, K., Hillygus, D. S., DeBell, M., & Iyengar, S. (2022). A randomized experiment evaluating survey mode effects for video interviewing. *Political Science Research and Methods*, 1–16. https://doi.org/10.1017/psrm.2022.30

Heerwegh, D. (2007). Mode differences between face-to-face and web surveys: An experimental investigation of data quality and social desirability. *International Journal of Public Opinion Research*, *21*(1), 111–121. https://doi.org/10.1093/ijpor/edn054

Heerwegh, D., & Loosveldt, G. (2008). Face-to-face versus web surveying in a high-internet-coverage population: Differences in response quality. *Public Opinion Quarterly*, *72*(5), 836–846. https://doi.org/10.1093/poq/nfn045

Hupp, A.L., Larsen K.M., Conrad F.G., Ong, A.R., Schober M.F., West B.T. & Wang, T. (2021). *Recruitment and participation in video interviews* [Paper presentation]. European Survey Research Association 9th Conference, Virtual.

Liu, M., & Wang, Y. (2015). Data collection mode effect on feeling thermometer questions: A comparison of face-to-face and Web surveys. *Computers in Human Behavior*, *48*, 212–218. https://doi.org/https://doi.org/10.1016/j.chb.2015.01.057

McGonagle, K., & Sastry, N., (2021). An experimental evaluation of an online interview scheduler: Effects on fieldwork outcomes. *Journal of Survey Statistics and Methodology*, *9*(3), 412–428. https://doi.org/10.1093/jssam/smaa031

Pew Research Center. (2021, September). *The internet and the pandemic*.

Schober, M.F., Okon, S., Conrad, F. G., Hupp, A.L., Ong, A.R., & Larsen, K.M. (2023). Predictors of willingness to participate in survey interviews conducted by live video. *Technology, Mind, and Behavior, 4*(2). https://doi.org/10.1037/tmb0000100

Schober, M.F., Conrad, F.G., Hupp, A. L., Larsen, K.M., Ong, A.R., & West, B.T. (2020). Design considerations for live video survey interviews. *Survey Practice, 13*(1). https://doi.org/10.29115/SP-2020-0014

Sun, H., Conrad, F. G., & Kreuter, F. (2021). The relationship between interviewer-respondent rapport and data quality. *Journal of Survey Statistics and Methodology, 9*(3), 429–448. https://doi.org/10.1093/jssam/smz043

West, B.T., Ong, A.R., Conrad, F. G., Schober, M.F., Larsen, K.M., & Hupp, A.L. (2022). Interviewer dffects in live video and prerecorded video interviewing. *Journal of Survey Statistics and Methodology, 10*(2), 317–336. https://doi.org/10.1093/jssam/smab040

> back to Table of Contents

Video Interviewing: An Optimal Solution for a National Behavioral Health Survey

Preethi Jayaram, Lilia Filippenko, Curry Spain, Matthew Check, Wendy Reed, Christine Carr, Heidi Guyer, and R. Suresh

1. Introduction/Abstract

The Mental and Substance Use Disorders Prevalence Study (MDPS) is a pioneering national study to estimate the prevalence of serious mental and substance use disorders among adults in the United States, including those residing in households, prisons, homeless shelters, and state psychiatric hospitals. The MDPS used a three-stage design for the household survey that consisted of a roster to establish eligibility and select adults for participation, a mental health screening survey that was used to disproportionately select those with a higher likelihood of disorders, a clinical interview that included the Structured Clinical Interview for DSM-5 (SCID-5®) (SCID-5®; First et al., 2015), and questions about treatment receipt. In the non-household component, a roster of age-eligible residents was obtained from participating prisons, homeless shelters, and state psychiatric hospitals. Then, the roster was sorted by key characteristics of the individuals, such as age and time since admission, and a random probability sample was then selected from the sorted roster via a systematic sampling scheme. MDPS utilized clinicians with clinical training in mental health, including experience conducting the SCID-5®, to conduct clinical interviews. The clinical interview was programmed in Blaise 5 and included a link to the NetSCID-5, a web-based version of the SCID-5. Video interviewing was planned for a large subset of the household sample, but the COVID-19 pandemic forced us to switch to this new paradigm for all of the household clinical interviews and to offer video interviewing as an option for clinical interviews conducted in the non-household settings. This paper describes the systems developed to support video interviewing and integration of a Blaise instrument on a large national data collection effort conducted in multiple settings. It also describes the development of an interview scheduling tool, automated reminders, logistical considerations in the various settings (i.e., interviewer and respondent setup), video recordings, quality reviews, interview editing, and feedback provided by both interviewers and respondents. Over 3,700 video interviews and 1,600 phone interviews were conducted using Zoom, in addition to approximately 200 in-person interviews within the facility settings. Video interviewing offers a novel mode of data collection with many of the same benefits of face-to-face interviewing and the added benefit of audio and video recording of the interviews. Lessons learned and future recommendations for national surveys are also provided.

2. Background

The initial plan was to conduct most household clinical interviews in person or by video, a subset by phone, and the non-household clinical interviews in person. Data collection was scheduled to start in July 2020. Because of the COVID-19 pandemic, we abandoned the in-person clinical interview data collection mode for the household sample. Video interviews were prioritized, and phone interviews were offered to respondents who couldn't or did not want to participate via video. The non-household data collection plan was also revised to incorporate video and phone modes.

3. Developing Systems for MDPS Video Interviewing

3.1 Web Scheduler

Because the household rostering and screening was web based, it became imperative that we have a reliable and convenient mechanism to schedule the video clinical interview. We took advantage of the fact that the respondent was already on the web completing the screener to offer them the opportunity to schedule their interview through the web. Later in the data collection process, when we did send out field

> back to Table of Contents

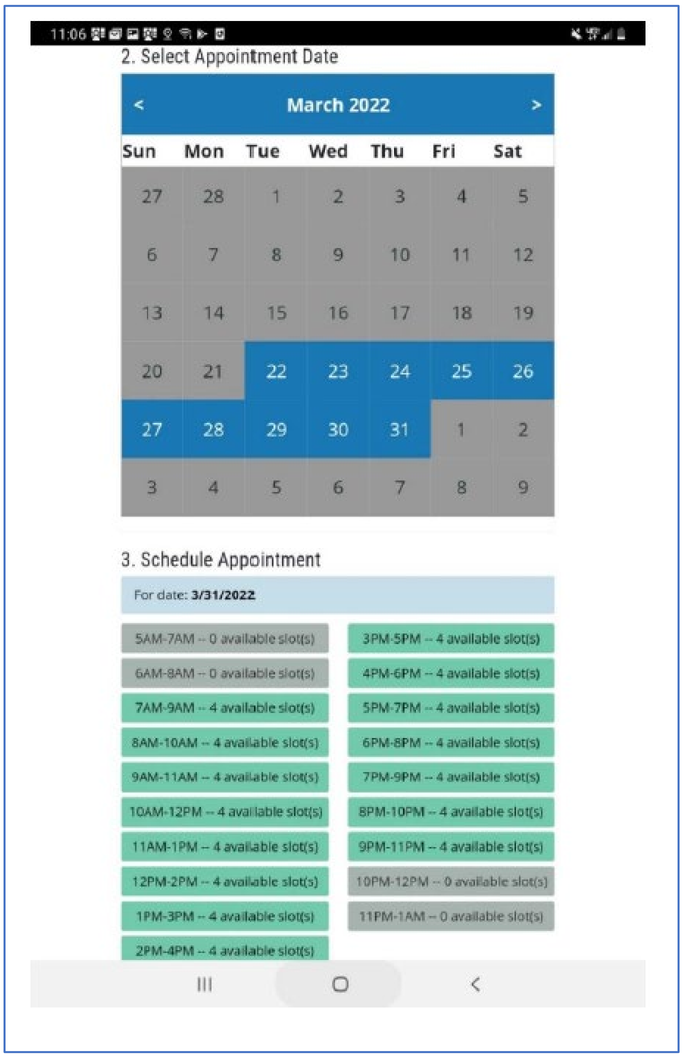
interviewers to administer the roster and screener, we were able to leverage the same web-based scheduler to schedule the household clinical interviews.

We designed the web scheduler to be as flexible as possible so that the same page could be used by both the respondents themselves and a variety of interviewing staff. At the end of the screener, respondents were automatically directed to the scheduler; respondents could also access the scheduler from the MDPS website using an access code. RTI International's call center staff could access the scheduler if a respondent called in to schedule or reschedule their appointment. RTI's computer-assisted telephone interviewing (CATI) staff and RTI field staff were routed to the scheduler at the end of the screener. RTI field staff also accessed the scheduler from their tablets as part of the field interviewer prompting effort.

The scheduler allowed the user to schedule, reschedule, or cancel the appointment. The scheduler automatically adjusted for the time zone and adapted for the availability of bilingual interviewers. It provided a confirmation screen for each of the actions that made the outcome of the user's actions clear to them.

Automated emails were sent to the respondents to confirm their appointments and to alert them 3 weeks, 1 week, and 1 day prior to their appointments. In addition, CIs received automated emails that listed the appointments for a given day. Figure 1 shows how data collection staff and respondents selected the date and time for the clinical interview appointment in the scheduler.

Figure 1. Scheduler Screen for Appointment Date/Time



2. Select Appointment Date

< March 2022 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

3. Schedule Appointment

For date: **3/31/2022**

5AM-7AM -- 0 available slot(s)	3PM-5PM -- 4 available slot(s)
6AM-8AM -- 0 available slot(s)	4PM-6PM -- 4 available slot(s)
7AM-9AM -- 4 available slot(s)	5PM-7PM -- 4 available slot(s)
8AM-10AM -- 4 available slot(s)	6PM-8PM -- 4 available slot(s)
9AM-11AM -- 4 available slot(s)	7PM-9PM -- 4 available slot(s)
10AM-12PM -- 4 available slot(s)	8PM-10PM -- 4 available slot(s)
11AM-1PM -- 4 available slot(s)	9PM-11PM -- 4 available slot(s)
12PM-2PM -- 4 available slot(s)	10PM-12PM -- 0 available slot(s)
1PM-3PM -- 4 available slot(s)	11PM-1AM -- 0 available slot(s)
2PM-4PM -- 4 available slot(s)	

3.2 Clinical Interview Instrument

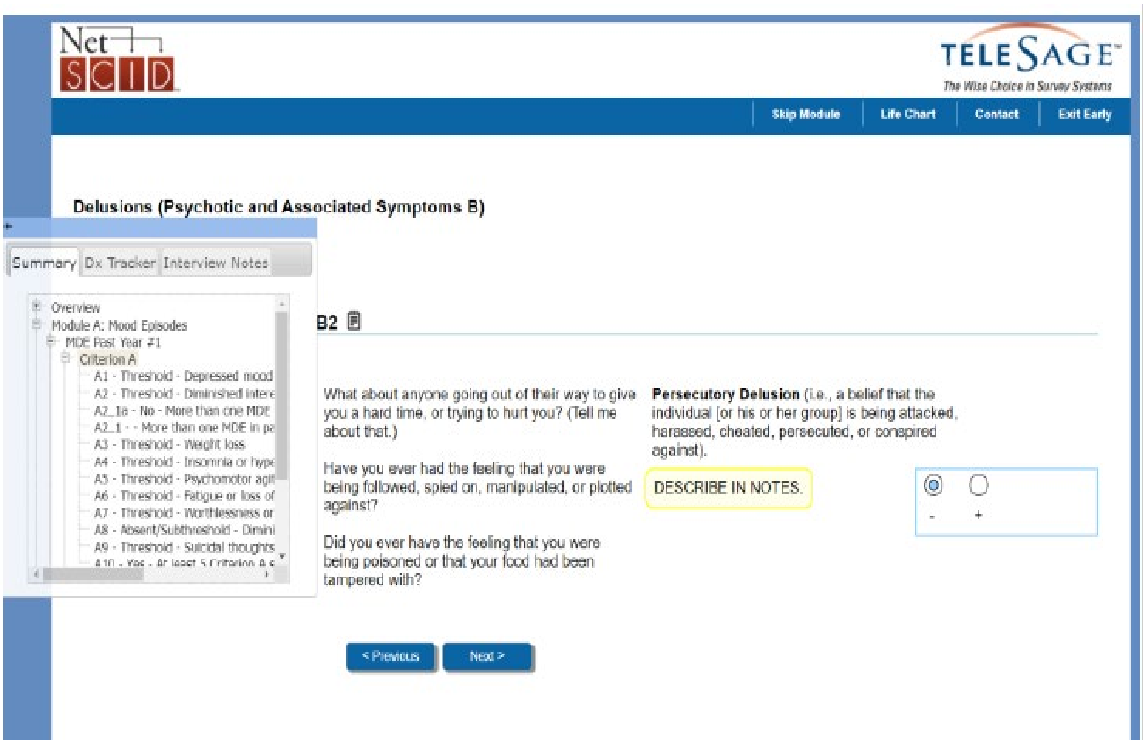
The purpose of the clinical interview instrument was to collect data to assess symptoms of mental and substance use disorders among adults and the proportion of adults who received treatment. The instrument was programmed using Blaise 5, and a SCID-5® instrument was launched within Blaise. All interviews were conducted by trained clinicians such as psychologists, psychiatrists, or social workers who had received training on MDPS instrumentation and procedures. Interviewers met with respondents via video, phone, or in person (non-household sample only) to conduct the clinical interview.

At the beginning of the clinical interview, respondents were asked for permission to record the interview. After completion, demographic information; use of tobacco, alcohol, and drugs; and the names of used medications was

> back to Table of Contents

collected. To begin the SCID-5, an action "GotoUri" was used from the Blaise instrument to start the RTI-developed NetSCID app.

The NetSCID allowed interviewers to connect to a third-party website to continue the interview. The NetSCID is a computerized version of the SCID-5. Once the NetSCID was completed, control went back to the Blaise survey for interview completion. In addition, a stand-alone Windows application (using C#) was built to allow administrators to connect to a specific case and download reports that contain data saved for that case.


Net SCID
TELESAGE
The Wise Choice in Survey Systems
Skip Module
Life Chart
Contact
Exit Early
Summary
Dx Tracker
Interview Notes
Overview
Module A: Mood Episodes
MDE Past Year #1
Criterion A
A1 - Threshold - Depressed mood
A2 - Threshold - Diminished intere
A2_1a - No - More than one MDE
A2_1 - - More than one MDE in pa
A3 - Threshold - Weight loss
A4 - Threshold - Insomnia or hype
A5 - Threshold - Psychomotor agit
A6 - Threshold - Fatigue or loss of
A7 - Threshold - Worthlessness or
A8 - Absent/Subthreshold - Dimini
A9 - Threshold - Suicidal thoughts
A10 - Yes - At least 5 Criterion A s
B2
DESCRIBE IN NOTES.
-
+
< Previous
Next >

Delusions (Psychotic and Associated Symptoms B)

What about anyone going out of their way to give you a hard time, or trying to hurt you? (Tell me about that.)

Have you ever had the feeling that you were being followed, spied on, manipulated, or plotted against?

Did you ever have the feeling that you were being poisoned or that your food had been tampered with?

Persecutory Delusion (i.e., a belief that the individual [or his or her group] is being attacked, harassed, cheated, persecuted, or conspired against).

The clinical interview was installed on a laptop with a Case Management System was set up to help the clinical interviewers (CIs) navigate through the list of respondents and to launch the Blaise interview.

When the SCID-5 was completed, control was returned to Blaise and the interviewer collected treatment information and feedback from the respondent about the interview and their own opinion in the Blaise instrument.

3.3 Video

Zoom was used by CIs to conduct clinical interviews with respondents in households and non-household facilities. Zoom meeting invitations were sent by email through CIs' Outlook accounts so that each Zoom session would have its own unique password. An Android tablet was used by the clinical interviewer to schedule and record the Zoom interviews and a laptop was used to administer the clinical interview questions. The video was recorded with the respondent's consent, and recordings were uploaded to the MDPS private site for a quality check.

Approximately 67% of the household clinical interviews and 31% of non-household clinical interviews were conducted via video.

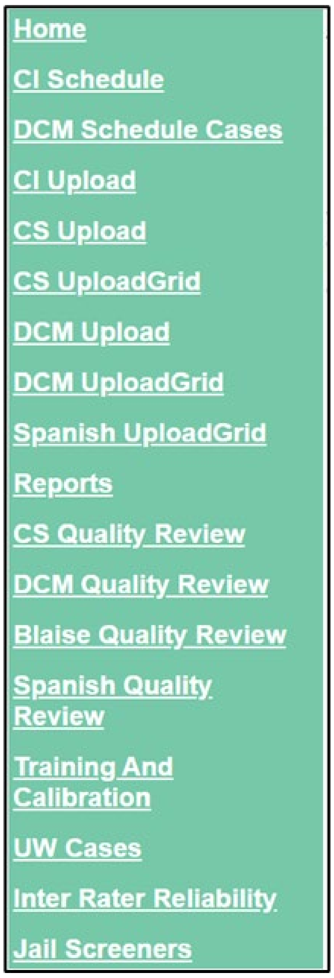
> back to Table of Contents

3.4 Private Site

A private site was developed to allow managers and supervisors to track interview scheduling and to modify schedules if needed. The site also allowed interviewers to upload Zoom recordings so that supervisors were able to review data quality. In addition, the site was designed to allow authorized users to conduct training and interrater reliability exercises.

The site was designed using .Net C#. It used a combination of webforms, APIs, and web services to allow users to not only navigate the site but also incorporate web services to allow access for recordings and external clients. Once the interview was completed, the interviewer received the URL to the Zoom recording file and associated password to access the recording. The interviewer then uploaded the file to the private site. Once uploaded, the file was linked to a case, and managers and supervisors were able to access, download, and review the file for quality control. Figure 2 shows the private site main menu.

Figure 2. Private Site Main Menu



3.5 Quality Control

Another advantage of video interviewing was that with the respondent's permission, the interview was recorded; these recordings were available for data quality reviews. Recordings were stored and reviewed on our private study website. The clinical section, which consisted of the structured clinical interview for the DSM-5, was reviewed by clinical supervisors (CSs). The nonclinical (Blaise) section, which included consents and questions on gender identification, cigarette and e-cigarette use, treatment, and COVID-19, was reviewed by data quality managers (DQMs).

CSs accessed the CS Data Quality section of the private study website to review and score the clinical module administration. This involved the CS accessing both the completed SCID and the interview recording, reviewing the data collected item by item, and comparing the notes provided by CIs with diagnostic ratings. CSs reviewed 10% of all completed interviews, which included (1) interviews selected at random and (2) interviews manually selected for review via the CI or the CS requesting that the interview be placed in the 10% review pile because of some uncertainty about any part of scoring. In addition, some video- and audio-recorded interviews that were not selected for full review received partial reviews upon CI request. For example, if a CI needed a second opinion on the scoring of a specific module, such as PTSD, this part of the interview was reviewed by their CS. For full reviews, either the audio or video recordings were reviewed in their entirety. If an interview was partially reviewed, audio and video file(s) were used, as necessary, to clarify responses, supplement notes, or support the CI's SCID ratings.

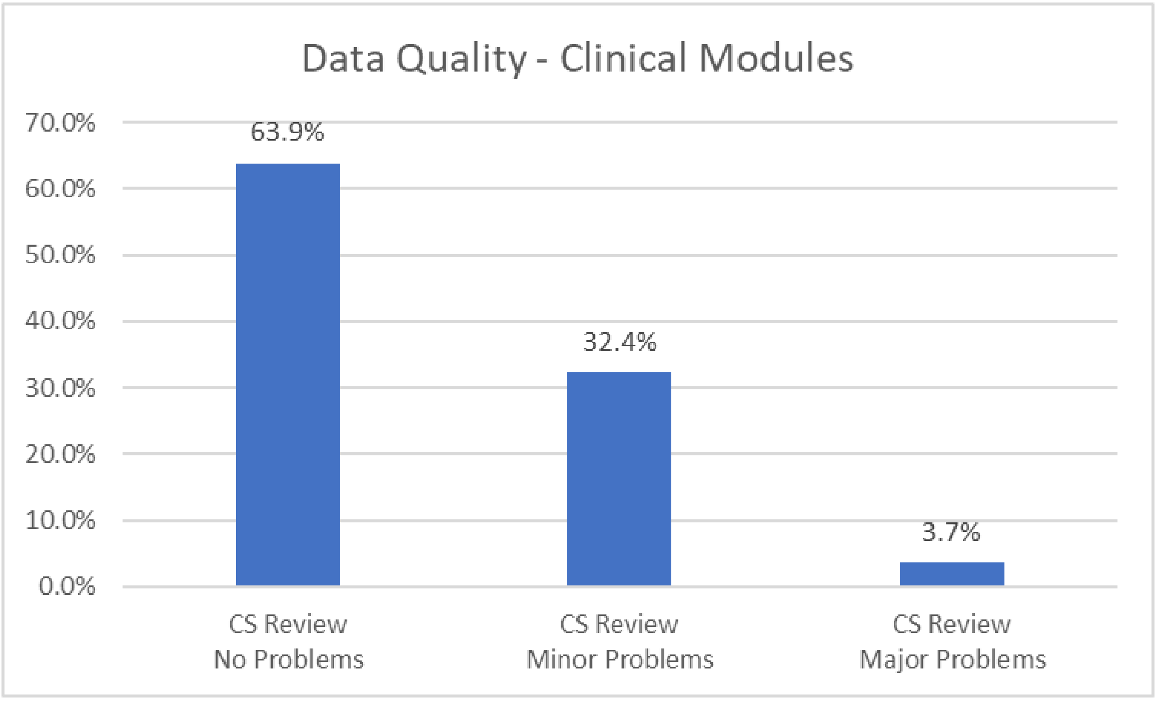
All clinical interview feedback was documented on a dedicated, interview-specific clinical review summary form, which included records of clinical editing, individualized feedback, and systems for monitoring CI performance. CSs provided CI performance ratings on their SCID module administration, and documented strengths and areas of improvement related to clinical interviewing. This summary form was used to provide feedback directly to CIs. Reviews included ratings on the following:

> back to Table of Contents

- providing accurate and sufficient notes,
- assigning proper or missed codes,
- the amount of probing,
- the types of probing,
- obtaining an adequate amount of relevant information,
- resolving inconsistencies,
- maintaining a professional demeanor,
- establishing rapport with the respondent,
- dealing with the respondent's emotions and body language properly,
- obtaining a description of the experiences a respondent reports in the respondent's own words,
- maintaining the correct interview pace,
- differentiating between symptoms that are easily confused, and
- handling distressed respondents appropriately.

After each review, the CS coded the case on the private site to indicate whether there were major issues, minor issues, or no issues with the interview. Minor issues involved symptom-level scoring changes that did not impact disorder-level scoring changes. Major issues involved disorder-level scoring changes. Data from interviews deemed to have issues that required a change in scoring were corrected by the CS before case finalization. Figure 3 shows the data quality outcomes of the clinical modules reviewed.

Figure 3. Clinical Data Quality



The DQMs accessed the DQM Data Quality section of the private study website to review the same set of interviews elected for CS review. DQMs focused the review on how well the CI administered the Blaise portion (i.e., non-SCID portion) of the clinical interview. This review included ensuring that CIs followed instrument scripts and collected accurate, high-quality data. All CIs' first two completed clinical interviews and 10% of all completed clinical interviews were selected to have the Blaise instrument administration reviewed. The DQM shared feedback from these Blaise reviews with the CI's Data

> back to Table of Contents

Collection Manager, who shared results with their CI via email or a face-to-face meeting, depending on the number and severity of issues identified. Results of these reviews were independently tracked by the data quality team.

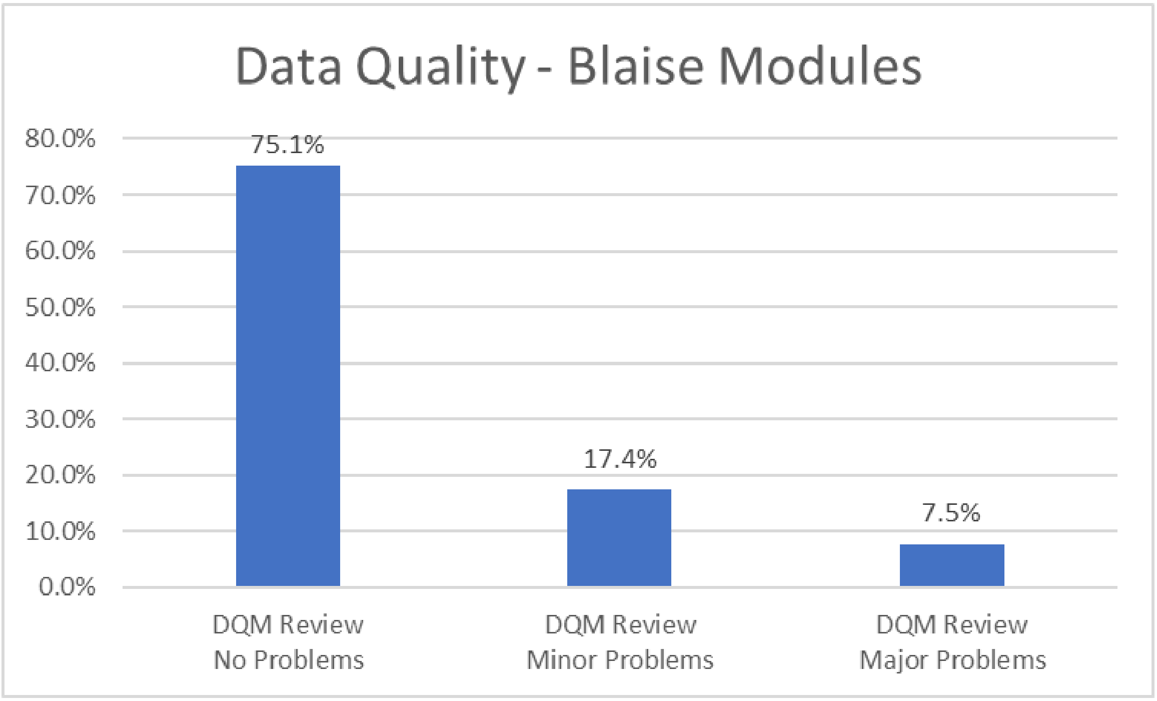
Blaise reviews were conducted by reviewing clinical interview recordings alongside the Blaise instrument specifications. Reviewers input individual interview results within a project-specific Blaise review tracking system that allowed for section- and interviewer-specific grades. The tracking system was broken down into three review sections:

- Front-end Blaise—included ratings and error notes (i.e., no errors, few errors, many errors) on properly administering consent and all pre-SCID questions
- Back-end Blaise—included ratings and error notes (i.e., no errors, few errors, many errors) on all post-SCID modules
- Interviewer Feedback—included ratings (i.e., unsatisfactory, satisfactory, excellent) on reading questions verbatim, effective probing techniques, not introducing bias, answering respondent questions and concerns, and interviewing pace and presence

After each review, the DQM coded the case on the private site to indicate whether there were major problems; minor problems, which meant there were errors that did not change the meaning of the question; and major problems, which meant that there were errors that did change the meaning of a question.

Figure 4 shows the data quality outcomes of the Blaise modules reviewed.

Figure 4. Blaise Data Quality



> back to Table of Contents

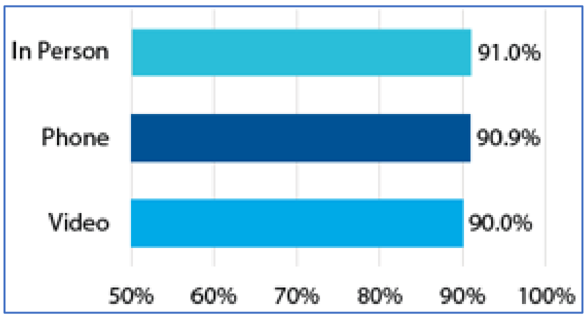
3.6 Feedback

Obtaining CI and respondent feedback on the clinical interview process was critical to inform study protocols and future data collection efforts. At the end of each clinical interview, we collected feedback from both the respondent and the CI about the clinical interview. We also sent each CI a survey near the end of the data collection period to collect additional information. A summary of the feedback received is included in the sections below.

3.7 Respondent Feedback

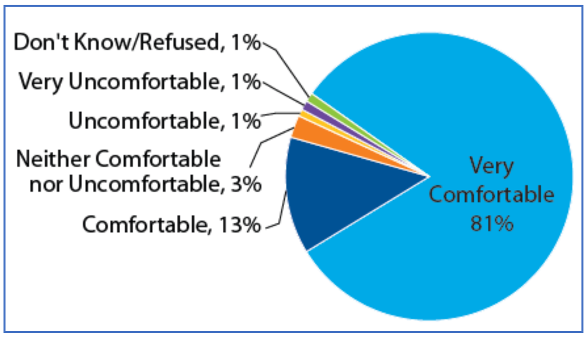
Because respondents could participate in the clinical interview via video, phone, or in person (non-household cases only), we asked about their comfort level with their selected interview mode. We found that 90% of respondents who completed the interview via video indicated they were comfortable; this was comparable to phone and in-person modes (Figure 5).

Figure 5. Respondent Comfort Level with Selected Interview Mode



We also asked video respondents how comfortable they were using Zoom; 94% indicated they were very comfortable or comfortable (Figure 6).

Figure 6. Respondent Comfort Level Using Zoom

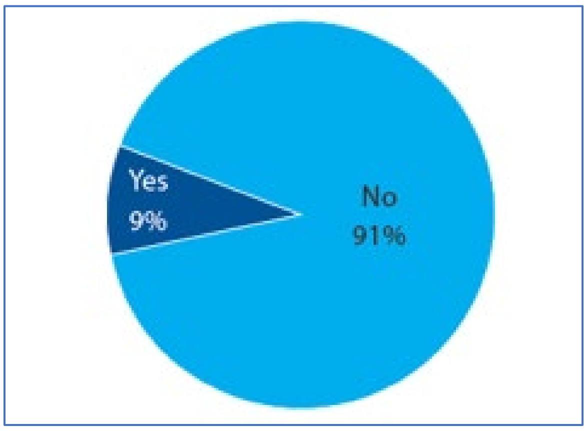


3.8 Clinical Interviewer Feedback

At the end of the interview, CIs were asked about the respondent's experience with Zoom. Figure 7 shows that most respondents did not have any technical difficulties with Zoom.

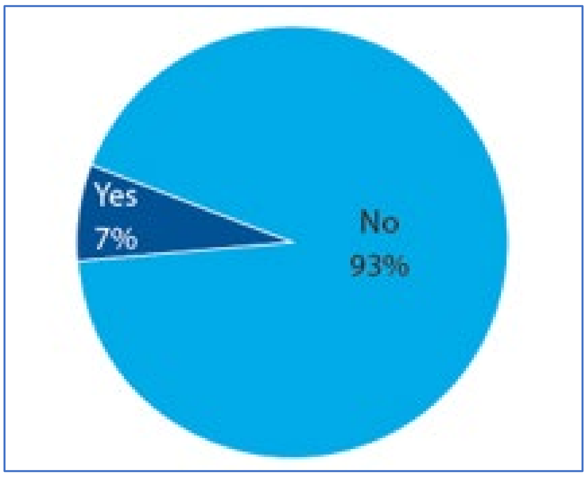
> back to Table of Contents

Figure 7. Respondent Technical Difficulties with Zoom



CIs also reported that most respondents did not get disconnected from the video interview (Figure 8).

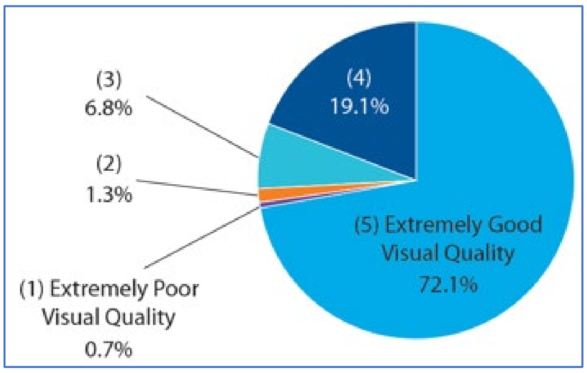
Figure 8. Respondent Disconnection from Video Interview



Because visual observations were important for this study (clinical observations were critical to accurately assess the negative symptoms of schizophrenia spectrum disorders), we also asked about the quality of the video. CIs indicated that for 91% of the interviews, the quality of the video was extremely good or good (Figure 9).

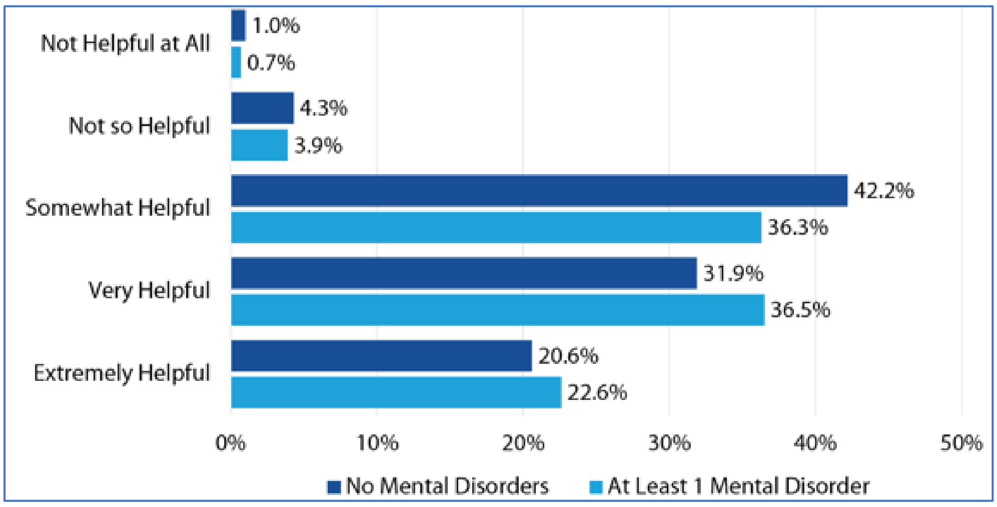
> back to Table of Contents

Figure 9. Overall Visual Quality of Interview



We also asked CIs about the use of visual observations in making diagnoses. In about 71% of cases where the respondent had at least one mental disorder, CIs indicated that they used visual observations. Figure 10 shows CIs' ratings of how helpful visual observations were in making a diagnosis—comparing respondents with no mental disorders to respondents with at least one mental disorder.

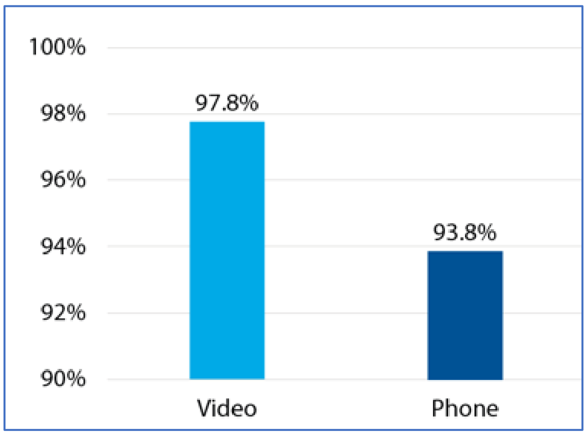
Figure 10. Helpfulness of Visual Observations in Making Diagnosis



CIs also indicated that the CI and the respondent were able to hear each other clearly during most of the interviews; the percentage was slightly higher for video interviews (Figure 11).

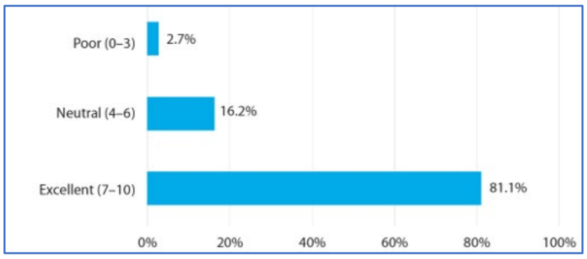
> back to Table of Contents

Figure 11. Interviewer and Respondent Able to Hear Each Other Clearly During Most of Interview



In a debriefing survey, CIs were asked to rate the functionality of the video interviewing process using a scale of 0 to 10**.** Figure 12 shows that most CIs rated the process as excellent.

Figure 12. Functionality of Video Interviewing Process



CIs also reported that Zoom was easy to use for all parties (CIs and respondents) and that most respondents were very familiar with Zoom.

4. Conclusions (Including Benefits/Drawbacks)

We found video interviewing to be an effective method and a feasible alternative to in-person interviewing. Ninety percent of our respondents reported being comfortable with video interviewing; 94% reported being comfortable with the interviewing software. Having the audio and video recordings made it easier to authenticate the interviews.

However, there were some drawbacks to using this method. Initial setup of Zoom required significant involvement of a technical support team. Uploading large video files took a lot of bandwidth, which resulted in increased interviewer and technical support labor. While the scheduler did send an appointment confirmation and automated reminders, the CI was still tasked with sending the interview Zoom link. Cis shared that using two devices (the tablet for Zoom and the laptop to administer the clinical interview questions) was challenging.

5. Future Recommendations

To avoid the issue with uploading large video files, reviewing them directly in the cloud would be a more robust solution. In addition, using a single device with two monitors—one to display the interview and

> back to Table of Contents

the other for the Zoom window—would be more convenient from a usability point of view. Automating the process of sending the Zoom link to the respondents would reduce the burden on CIs.

6. References

First, M. B., Williams, J. B. W., Karg, R. S., & Spitzer, R. L. (2015). *Structured clinical interview for DSM-5—Research version* (SCID-5 for DSM-5, research version; SCID-5-RV) (pp. 1–94). American Psychiatric Association.

NetSCID is a computerized version of the Structured Clinicial Interview for the DSM-5 (SCID-5) and is fully licensed by the American Psychiatric Association through American Psychiatric Association Publishing. https://www.telesage.com/netscid-5

> back to Table of Contents

Video Interviewing at the University of Michigan

Andrew Hupp, University of Michigan

1. Background

Video communication is increasingly common, from live two-way video (e.g., Zoom, FaceTime, etc.), to live one-way streaming (e.g. Ring doorbells, baby monitors, etc.), to playing recorded videos (e.g., TikTok, YouTube, etc.). It's become standard on computers and smartphones.

The pandemic prompted survey researchers to turn to video technology as an alternative to in-person data collection. Despite a widespread increase in familiarity with video technology, the novelty of video as a survey mode leaves many questions unanswered.

This paper discusses three video related projects that the Survey Research Center (SRC) at the University of Michigan (UM) has been engaged with to try and understand some of these issues, and test different ways in which some of the various components can be implemented.

2. Video Communication Technologies

The first study we carried out was a mode experiment study where three modes, live video, prerecorded video (a web survey with a video of an interviewer reading the questions), and a textual web survey (Conrad et al., 2023), were compared in terms of data quality. The project was carried out from August 2019 to March 2020 using both an address-based sample, and two online non-probability panels as recruitment sources. All three modes were implemented in a single Blaise 5.6.5 instrument that allowed for mode specific displays. To promote comparability between modes, question batteries were always presented as a series of individual questions. In the two web modes, the display was optimized for screen size, (e.g., using response buttons that included the text of the response within the button for devices with smaller screens, and radio buttons for devices with larger screens).

2.1 Live Video

See Conrad et al., 2023, and Schober et al., 2020) for more detail on the implementation and design considerations. See https://www.mivideo.it.umich.edu/media/t/1_1zoid4cu for an example of a live video interview.

2.1.1 Platform Considerations

The platform decision came down to Zoom and BlueJeans. Zoom was the institutional platform at the New School for Social Research (a research partner), and BlueJeans was the institutional platform at the UM. Either came at no cost to the project. The project ultimately decided to move forward with one platform, BlueJeans. BlueJeans had an option to force the launch of the platform in a browser, making it an easy solution for the respondent since they didn't need to download and set-up anything, and we were able to get additional paradata from the platform related to the video call (see Ong et al., 2019). Using one platform also simplified the (limited) support interviewers provided when technical issues were encountered. We were also able to generate unique meeting links en masse prior to the project and assigned a link to each sample line in the management system.

2.1.2 Recruitment

> back to Table of Contents

There were questions about how respondents should be recruited and able to join video interviews. Unsolicited calls using a platform like FaceTime, were not considered because it seemed unlikely that respondents would answer, and researchers would have trouble assembling a sampling frame. Two possible models remained, scheduling a time in advance, or having interviewers available to conduct video interviews on-demand (see Guggenheim et al., 2021), both of which require an invitation in another mode (e.g., mail, email, text message, in-person, or telephone). The on-demand model seemed potentially cost inefficient, with interviewers waiting around for people to call in, so the project decided to recruit via another mode and have potential respondents schedule a time via Calendly (an online scheduling software). We were concurrently conducting a methodological experiment to understand interviewer effects in video interviewing (see West et al., 2022) which added another complexity to the scheduling. An interviewer was randomly assigned to a potential respondent during the scheduling phase. When a respondent arrived at the site to schedule an appointment they were shown times that particular interviewer was available.

2.1.3 Interviewer Considerations

2.1.3.1 Set-up

Live video interviews were conducted by nine experienced call center interviewers from the Survey Research Center (SRC) at the UM. Since interviewers informed respondents they were calling from the UM it was decided to leave the (live) call center as their background, rather than using a virtual background. The interviewers used one monitor during the interview. The screen was split with the video window in the upper portion of the screen and the Blaise instrument in the lower portion of the screen. We did this because we wanted the respondent to have the sense that the interviewer was looking at them while they were answering the question, like in an in-person interview. Each sample line had a unique BlueJeans link preloaded in the management system. That unique BlueJeans link was pushed to the Blaise instrument, where the interviewer clicked a BlueJeans icon to launch the interview meeting.

2.1.3.2 Training

While the interviewers were experienced administering telephone interviews, they did not have any prior experience conducting live video interviews. Video interaction is different from voice only interaction, in that the interviewer's facial expressions and visual reactions can now be seen, and it's different than an in-person interaction in that the interview is being mediated (Schober et al., 2020). As part of training the interviewers had practice sessions on remaining neutral, professional behavior (e.g., what to wear, drinking and eating, etc.), practicing setting up the multiple windows on their screens, practicing the interview, and sending messages to the respondent via the management system if the respondent was late to their appointment. As part of classroom training they received background about the project, and documentation on providing limited technical support to the respondent (e.g., turning the camera/microphone on/off, rejoining a meeting, etc.) if problems were to occur.

2.2 Prerecorded Video

See Conrad et al., 2023 for more detail, and https://www.mivideo.it.umich.edu/media/t/1_vjhtigaf for an example of a prerecorded video interview.

As noted earlier recorded videos are common. This type of video communication is one-way, and engages the viewer in ways that text alone does not. Given this, it seemed that it would be worth exploring prerecorded videos of survey interviewers reading questions in a self-administered web survey.

> back to Table of Contents

Prior research on this topic is limited. Fuchs and Funke (2007) report more socially desirable answers to one of four questions; Haan et al., (2017) report no differences across 19 questions. Fuchs (2009) reports gender of interviewer effects for four of four questions. So, even though recoded videos of interviewers are inanimate and insensate, respondents sometimes react socially, which is consistent with the "Computers Are Social Actors" framework (Nass and Moon, 2000).

The same nine interviewers that administered the live video interviews were recorded speaking the questions using Camtasia. Each question was recorded as a separate file. Each page in the web survey, initially contained only the video (no question text was ever displayed). In the desktop/laptop version, the videos auto-played to reduce the respondent's effort, and to give the delivery of the questions an interviewer-administered character. On mobile devices the video would not auto-play, so respondents were instructed to tap the play button to view each video. The respondent could replay the video to hear the question again.

Response options did not appear until the question was asked, again mimicking the way an interviewer would read the question as worded to make sure the respondent heard the entire question in an in-person interview. The video controls available in the software did not allow us to do this, so we created a look-up table with the speed at which each interviewer read each question so the Blaise instrument knew when to display the responses on that page, for that question, for that interviewer.

3. Standard Achievement Assessments

The second study was a small (n=32) pilot of conducting standard achievement assessments of math, reading, and vocabulary skills with children between the ages of 5-17, via video. Traditionally these assessments have only been done in-person, but the publisher endorsed data collection via video for both clinical and research purposes during the COVID-19 pandemic.

3.1 Set-up

3.1.1 Interviewer Set-up

Two experienced interviewers were provided a laptop with a camera, an additional monitor, and a headset. The laptop had Blaise and the Zoom client installed. Each interviewer received 22 hours of training on how to administer the interview.

3.1.2 Respondent Set-up

Given how (and to whom) the standard achievement assessments are administered it was determined it would be best to have the survey organization provide a device, with internet connectivity. This allowed for standardization of the device used by the respondent, and provided a family a way in which to participate if they didn't own the equipment needed or have an internet connection.

The survey organization purchased, and set up five Samsung Galaxy 10 tablets to provide to respondents. Each tablet was in a protective case, had cellular connectivity, and was limited in functionality. The tablet was set-up in kiosk mode with two applications: Zoom (for the video interview), and Pushover (for notifications).

3.2 Implementation

3.2.1 Recruitment, Scheduling, and Notification

> back to Table of Contents

Interviewers recruited families via a Qualtrics screening instrument. As part of the recruitment, interviewers reviewed study information with the parent, set an appointment for the interview, and reviewed the tablet shipping logistics.

Appointment information, including the unique Zoom meeting link, was entered into a central project calendar. Notifications of the appointments were pushed to the tablet via Pushover. Interviewers would call parents via the phone to walk them through the steps to start the Zoom session for the child. Once the Zoom session was started, the interviewer ended the call with the parent and continued the Zoom session with the child.

3.2.2 Instrument

Three subtests were adapted for administration via video. Pages from those tests were digitized and loaded in the Blaise instrument. Blaise would launch a window with the digitized image to the second monitor, which was in portrait orientation. The interviewer shared the display in the second monitor to the tablet the respondent was using. The video feed from the respondent's tablet was viewable on the interviewer's second monitor so they could observe the test-taker and the testing environment. Annotation was enabled on the tablet, which allowed the interviewer to observe when a child pointed to a word or an image on the screen. Any annotations could be cleared by the interviewer prior to displaying the next assessment image to the respondent.

3.2.3 Logistics

A tablet was shipped to the respondent's address prior to the interview. When the interview was completed the interviewer set-up the UPS return, and reviewed the packing instructions with the parent. Prior to the date of pick-up, the interviewer contacted the parent to remind them.

Once the tablet was received the respondent was sent $50 for their participation. The tablet was then sanitized, had the Zoom link removed, checked to see if the Zoom application had accidently been logged out of, or had any settings updated, charged, and prepared for shipment to the next family for their interview.

4. Child Development Supplement

The third study, the Child Development Supplement explored augmenting their in-person interviews with live video interviews for children ages eight to eleven as part of a small (n=28) 2023 pretest. Traditionally these interviews have been done in-person, but due to time limitations on in-person visits, and the remote locations of some families, interviewing in-person is prohibitive.

4.1 Set-up

Zoom was installed on the interviewer's organization issued smartphone. The smartphone was positioned on a stand, "peaking" over the interviewer's laptop screen. The Blaise instrument was displayed on the laptop screen, with the interviewer reading each question and entering the response spoken by the respondent.

A parent helped the child respondent set-up a family owned device with Zoom. The parent was allowed to stay in the room and could help if there were technical issues. If the family did not have a device on which to do the interview via video, they could do the interview over the phone, or could choose not to do the interview.

> back to Table of Contents

Show cards were utilized on a few questions in the instrument. Due to the current configurations on the organization issued smartphones, interviewers were not able to share the content via Zoom, so the interviewer held the physical show card up to the camera for the respondent to view. In practice, this didn't work well, especially with a virtual background.

4.2 Zoom Configuration

The study required several Zoom settings to be updated. The interviewers were instructed how to configure their global Zoom settings. *Outbound Caller ID* was adjusted from the default generic number, to the interviewer's smartphone number.

Under *Security Settings, Requiring a passcode when scheduled new meetings*, *Require a passcode for instant meetings*, *Require a passcode for participants joining by phone*, and *Embed passcode in invite link for one-click join* were all enabled, and *Require a passcode for Personal Meeting ID (PMI)* was disabled.

Two items under *Schedule Meeting* were adjusted. *Audio Type* was set to *Telephone and Computer Audio,* and *Enable Personal Meeting ID* was disabled.

Under *In Meeting (Basic)* four settings were adjusted. *Require encryption for 3rd party endpoints (SIP/H.323)*, and *Whiteboard (Classic)* were enabled, and *Meeting chat – Auto-save* and *Send files via meeting chat,* and *Automatically create local export when sharing is stopped*, under *Whiteboard (Classic)*, were disabled.

Under *In Meeting (Advanced)* two settings were adjusted. *Save Captions* was disabled, and *Virtual background* was enabled. A background provided by the study was selected as the virtual background.

Under *Recording*, *Local recording*, *cloud recording*, and *Allow cloud recording sharing* were all disabled.

4.3 Scheduling

The interviewer contacted a parent via the phone to schedule the interview meeting. The interviewer scheduled the meeting with the respondent using their Google calendar with the Zoom for Google Workspace add-on. The interviewer would add the email address of the parent, and a study email account so both would receive the meeting invite details. The Zoom meeting ID generated, was entered into the sample management system. The interviewer called the parent 24 hours prior reminding them of the appointment scheduled for the next day.

4.4 Quality Control

Interviews were recorded for quality control purposes. The interviewer's organization issued smartphone was connected to the laptop so the audio stream from Zoom and the data entry of responses into Blaise were recorded together. There was a concern about the potential of recording something that might need to be purged, so the decision was made to not record the video stream from Zoom.

5. Ongoing and Future Work

We have an upcoming project that interviews people at two different points in time. The first interview is in-person, with 50% of the second interviews being conducted in-person, and the other 50% via live video. We are conducting a methodological pilot to understand what influences participating in video interviews in this model. We are testing, 1) prepaying for the video interview at the conclusion of the first (in-person) interview, 2) helping the respondent with any questions they might have about second (video),

> back to Table of Contents

including helping them conduct a video call on the spot, at the conclusion of the first (in-person) interview, and 3) the timing of scheduling and reminding about the second interview. There is a delay of at least a month between the two interviews in the pilot (longer in production). We are scheduling the second interview at the conclusion of the first and trying a reminder strategy where the interviewer sends the respondent an email and/or text message with the Zoom link the day prior to the appointment, a reminder (~2 hours) the day of the appointment, and another message if the respondent is late (~5 minutes) to the appointment asking them to contact the interviewer if they are having problems joining.

This paper has highlighted different areas of three different projects that have conducted interviews using video. There are some similarities, and some differences in how things were implemented. This could be due to the design of a particular project, or could be due to the fact that there are multiple ways in which some of these things can be implemented. More research is needed to understand what works well, what doesn't work, and how these things effect things like data quality when using video for survey interviews.

6. References

Conrad F.G., Schober M.F., Hupp A.L., West B.T., Larsen K.M., Ong A.R., & Wang T. (2023). Video in Survey Interviews: Effects on data quality and respondent experience. *methods, data, analyses, 17*(2) 135-170. https://doi.org/10.12758/mda.2022.13

Fuchs, M. (2009). Gender-of-interviewer effects in a video-enhanced web survey: Results from a randomized field experiment. Social Psychology, 40(1), 37–42. https://doi.org/10.1027/1864-9335.40.1.37

Fuchs, M., & Funke, F. (2007). Video web survey - Results of an experimental comparison with a text-based web survey. In M. Trotman, T. Burrell, L. Gerrard, K. Anderton, G. Basi, M. Couper, K. Morris, K. Birks, A. Johnson, R. B. (Market Strategies), M. R. (PSI), S. T. (Inputech), & A. W. (Survey & S. Computing) (Eds.), *Proceedings of the Fifth International Conference of the Association for Survey Computing: The Challenges of a Changing World* (pp. 63–80).

Guggenheim, L., Howell, D., Amsbary, M., & DeBell, M. (2021). Live Video Interviewing in the ANES 2020 Time Series Study? Presented at the 2021 Conference of the European Survey Research Association, Virtual.

Haan, M., Ongena, Y. P., Vannieuwenhuyze, J. T. A., & De Glopper, K. (2017). Response behavior in a video-web survey: A mode comparison study. *Journal of Survey Statistics and Methodology*, 5(1), 48–69. https://doi.org/10.1093/jssam/smw023

Nass, C., & Moon, Y. (2000). Machines and Mindlessness: Social Responses to Computers. *Journal of Social Issues, 56*(1), 81-103. https://doi.org/10.1111/0022-4537.00153

Ong, A.R., Conrad, F.G., Larsen, K.M., Schober, M.F., Hupp, A.L., & West, B.T. (2019). What Can We Learn About Data Quality from Video Communication Paradata? Presented at the Midwest Association for Public Opinion Research, Chicago, IL.

Schober, M.F., Conrad, F.G., Hupp, A. L., Larsen, K.M., Ong, A.R., & West, B.T. (2020). Design Considerations for Live Video Survey Interviews. *Survey Practice, 13*(1). https://doi.org/10.29115/SP-2020-0014

West, B.T., Ong, A.R., Conrad, F. G., Schober, M.F., Larsen, K.M., & Hupp, A.L. (2022). Interviewer Effects in Live Video and Prerecorded Video Interviewing. *Journal of Survey Statistics and Methodology, 10*(2), 317-336. https://doi.org/10.1093/jssam/smab040

> back to Table of Contents

Converting Social Survey Blaise 4 Questionnaires to Blaise 5: Creating a Blaise 5 Application and Modernizing the Labour Force Survey

Andy Watson and Steve Maurice, Office for National Statistics, United Kingdom

1. Introduction

The Office for National Statistics (ONS) Social Surveys Delivery (SSD) runs numerous longitudinal, annual, and ad hoc surveys. Most of these surveys are conducted as face-to-face interviews on Blaise 4. The decision has been made to upgrade these surveys to Blaise 5 so that future surveys can more easily incorporate mixed mode elements such as computer-assisted web interviewing (CAWI) and computer-assisted telephone interviewing (CATI).

The Blaise 5 uplift project was initiated to transform all the SSD surveys from Blaise 4 to Blaise 5 and all the associated legacy support systems.

1.1 The Survey for Living Cost and Food (LCF)

The Living Costs and Food Survey (LCF) collects information on spending patterns and the cost of living that reflects household budgets across the country. The study provides information about household spending patterns, which is used to update the contents of the consumer inflation basket of goods and services. It is also used to provide information about food consumption and nutrition. It is an important source of economic and social data for government and other research agencies.

The study is conducted throughout the year across the whole of the United Kingdom and is the most significant consumer study undertaken in the country. The results are essential for understanding society and planning to meet its needs.

Government departments use the results of the study to identify how and where they should be using public resources.

They use the information to check how different groups in the community are affected by existing policies and to inform future policy changes.

1.2 The Labour Force Survey (LFS)

The Labour Force Survey (LFS) is the largest regular social survey in the United Kingdom. Its purpose is to provide information on the UK labor market, which can then be used to develop, manage, evaluate ,and report on labor market policies. It has been running since 1973.

The main estimates of employment and unemployment in the United Kingdom are taken from the LFS, and around 600 field interviewers and 200 telephone interviewers regularly work on the LFS. They survey a random sample of almost 45,000 UK households every 3 months.

2. Behind the Surveys

2.1 LCF

The LCF collects information on spending patterns and the cost of living that reflects household budgets across the country.

The study provides information that is used to update the contents of the consumer inflation basket of goods and services. It is also used to provide information about food consumption. It is an important source of economic and social data for government and other research agencies.

> back to Table of Contents

Currently, detailed expenditure data is collected by interviewers using Excel spreadsheets and embedding images of receipts within them. The process was developed at pace when the COVID-19 pandemic hit in March 2020. No user testing was carried out prior to delivery due to the pace of implementation needed to minimize the pause in data collection for the LCF.

2.1.1 LCF Methodology

The overall response rate for the LCF in Great Britain was 40% in the financial year ending (FYE) 2020, affected by fieldwork being paused because of the pandemic. This is a 3% decline when compared with FYE 2018 and FYE 2019. A total of 13,996 addresses were sampled for the LCF in Great Britain. Of these, 10% did not contain a private household and were therefore classified as ineligible.

Of the eligible sample, it was not possible to contact 9% of addresses; a further 43% refused to take part and 6% had another reason for nonresponse. Of the 5,072 responding households in Great Britain, 4,964 cooperated fully, meaning they completed both the interview and diary sections of the survey.

In FYE 2020, partial responses accounted for 2% of all cooperating households. Of these 108 partial responses, 107 occurred because one or more adults in the household refused to keep the diary but were happy to take part in the interview.

Interviewers recorded the main reason why people refused before or during an interview from a list of precoded answers. In FYE 2020, the two most cited reasons for refusing to take part in the survey were:

- Can’t be bothered—24%, which remains the top reason cited as was true in the previous financial year.
- Temporarily too busy—16%, which is in line with the previous year.

Falling response rates are an acknowledged problem, and we have various initiatives to help tackle these.

2.1.2 LCF Collection

Data collection for the LCF is done face to face by field force interviewers in the homes of the respondents. Advanced letters are sent to all sampled respondents and a follow-up contact is made by the interviewers to arrange a convenient time for the interview. An In-house case management system, Casebook, is used to manage all the interviewer’s cases every month.

The respondent is asked to keep track of all their spending over a two-week period, by either making note of what they purchase or preferably by keeping the receipts for all their purchases. When the two-week period ends, the interviewer returns to the respondent’s address to collect this data.

The interviewer enters this detailed expenditure data using Excel spreadsheets and embedding images of receipts within them. The process was developed at pace when the pandemic hit in March 2020. No user testing was carried out prior to delivery due to the pace of implementation needed to minimize the pause in data collection for the LCF.

2.2 LFS

The LFS is a survey of households living at private addresses in the United Kingdom. Its purpose is to provide information on the UK labor market that can then be used to develop, manage, evaluate, and report on labor market policies.

Having originally been conducted every two years from 1973 the methodology, the frequency of collection and sample size changed over time, moving to its current state of collection by calendar

> back to Table of Contents

quarters plus an annual boost sample added in May 2006.

2.2.1 LFS Methodology

The LFS uses a rotational sampling design, whereby a household, once initially selected for interview, is retained in the sample for a total of five consecutive quarters.

The interviews are scheduled to take place exactly 13 weeks apart, so that the fifth interview takes place one year after the first.

We define Wave 1 to be the first quarter an address is selected, Wave 2 to be the second quarter in the selection, and so on. Therefore, Wave 5 is the last time that household will be interviewed for the main LFS.

2.2.2 LMS Collection

LFS fieldwork is carried out by the LFS interviewing force, which is composed of both face-to-face interviewers, who work from their homes, and telephone interviewers, who work in a centralized Telephone Operations Unit. Attempts were made to develop a web version of the questionnaire in Blaise 4, but the results weren’t great.

3. LCF Questionnaire

The questionnaire is written in in Blaise 4.8 in a Computer-assisted personal interviewing (CAPI) format, with the interviewer leading the respondent through the questionnaire.

3.1 The Diary

The diary is a separate collection to the Main LCF questionnaire, which is collected by the interview and imputed into a Blaise 4 questionnaire by coders. The coders are asked to apply a Classification of Individual Consumption According to Purpose (COICOP) code for each item so that types of purchases can be analyzed at a granular level.

A small household section is completed first that includes First Name, Surname and Diary Type. Diary Type establishes if the household member is and adult or a youth. All household members are asked to fill in a diary but can refuse. This needs to be captured to establish household size.

The Usual purchases section records what type of items the household usually purchases, so that if any information is incomplete, an informed decision can be made to complete the coding.

For example, if the answer to “What type of sausages do you normally buy?” is Pork, then an entry of “Sausages - £2.50” is entered. They will be coded as Pork Sausages.

The diary collects 10 main areas of interest on expenditure within a household:

1. Food and drink
2. Takeaway food
3. Eating out
4. Clothing and footwear
5. Winnings
6. Home grown food
7. Trips abroad
8. Business refunds
9. Pocket money
10. All other purchases and payments

> back to Table of Contents

The diary is completed at the person level, meaning each person within the household (unless they have refused) has a record of their spending over the two-week period.

There is a Main Food Spender; this person is identified within the household as the person who does the "weekly" shopping for the household and usually has the most recorded items.

The main diary keeper keeps all large receipted "family" shopping as well as populating the "Usual Purchases."

Each person enters their individual Spending Pattern information, which references whether the household member has any reasons for any unusual spending patterns. It may be that they were on holiday for the two weeks and did not do a weekly shop, etc.

Non-Receipted items are listed separately and usually cover small amounts of items that are bought on an ad hoc basis, like going to the newsagents and buying a paper and a can of coke, whereas receipted items are entered as a total amount on the receipt. For these items, only the shop name and the total amount are recorded at this point. The coders then enter each item separately applying COICOP codes as they go.

3.1.1 Existing Process—and Paper

Previously, each person kept a record of their individual spending by recording it all on a paper diary and giving this to the interviewer along with the receipts they had collected for the period.

Now, the interviewer sets up an Excel spreadsheet for each member of the household, which contains a tab for each day of the collection period and a separate "area" on that tab for each expenditure block. The interviewer then must scan the receipts to create a digital version.

The Excel spreadsheets and digital receipts are then submitted to HQ for the coders to pick up for manual entry into the Blaise 4 questionnaire.

3.1.2 The Original RB5 Redesign

The new requirement is to be able to enter the information into a Blaise 5 questionnaire so that some of the data can be transferred automatically.

It needs to be a single questionnaire but with the maneuverability of an Excel spreadsheet. It should still gather the same data as the multiple files.

To make the questionnaire easy to navigate between the numerous areas of interest, use of the "Go to Parallel" within the Resource Database was going to be employed. But this would mean a lot of parallels. A lot! These parallels would also have to be nested, creating another complication to the design process.

- 10 People − 14 Days − 10 areas of Expenditure
- 10 × 14 × 10 = 1,400 Parallels!

The expenditure question blocks were all programmed and tested independently, so that multiple workstreams could be run. The main questionnaire with routing and navigation was developed separately and the question blocks would be entered later. During initial testing, the questionnaire contained arrays for 2 people, on 2 days, for 10 blocks, creating only 40 parallels.

The expenditure blocks were then added in successfully and a test questionnaire produced was uploaded to our internal hosting environment.

> back to Table of Contents

3.2 Problems with OG Design

Although the questionnaire was built successfully and worked as expected, as soon as the arrays were fully expanded to all 10 people for 14 days and all 1,400 parallels, the performance became a serious issue.

Even though the pages were built for the package, which took approximately 8 hours, it was observed that the loading time between pages was extremely slow—about 1.5 minutes per page. It was quickly decided that this was untenable and that a new solution had to be explored.

3.3 The New Proposed Solution

Previously, we had explored a "two-part" questionnaire but had discarded this solution as our in-house hosting system would not be able to support the movement between two questionnaires. When the issues arose, we quickly created a proof of concept for a two-part design to see if the issues with the in-house system could be resolved. With the support of our digital services team, a change to the core code in the servers' token system allowed the movement from one URL to another using the same token, allowing us to jump from one questionnaire to another.

The "A" section of the new questionnaire would collect the household information—people in the house, usual purchases, etc.—and then from a menu page containing a list of valid members of the household, and a button that then would launch the "B" section.

The new "B" section would then hold the household members information at a "person-level" database. Each person would enter all their expenditure for all the days.

This reduces the number of parallels in each section to:

- A—10 Parallels (1 for each person)
- B—140 Parallels (14 days, each with 10 Expenditure blocks)

Although this required two questionnaires, doubling monthly resource requirements, this development greatly reduced both the compilation time and page load times at runtime.

3.4 Further Developments

During development of the expenditure block, it was decided that functioning summary pages would aid the interviewer in reviewing the potentially large amount of items at the end of each block. It was further discussed that it would also be helpful to enable them to edit those blocks from that review page.

If the rollout of the finalized tool is successful, Blaise 5 would be considered for the further development of the respondent-facing tool.

4. Labour Market Survey (LMS) Questionnaire

As with the LFS, the LMS questionnaire has already seen quite a few changes since its inception, and more are planned for the very near future.

4.1 Early LMS

Originally set up as a longitudinal test ahead of starting work on an LMS proper questionnaire, the longitudinal test questionnaire was web-only based and the large database included samples for all five waves for 13 cohorts. Cohorts were to be invited in one-week intervals, and the whole test was to run over five quarters mirroring LFS. As the ONS was planning to repurpose the census collection tool we had developed for business and social survey use, we were unable to host the questionnaire ourselves and were helped by colleagues at NISRA, who hosted the questionnaire for us.

> back to Table of Contents

The LMS had a login questionnaire with a “start survey” instruction at parallel end to redirect to the single test questionnaire GUID.

Because no training had been provided to Telephone Operations staff at this point, management of refusals, appointments, and other processing such as invite/reminder letters were conducted from a separate Blaise 4 Survey Enquiry Line (SEL) questionnaire, which every night just after midnight was populated with outcome codes from the Blaise 5 instrument downloaded and inserted via automated Manipula runs. All post-collection processing was run off this Blaise 4 instrument.

As news of a global pandemic broke, it was decided to increase the sample and ramp up the development of the longitudinal test, adding new cohorts each quarter, so that if the worst-case scenario happened and LFS collection had to stop, we might still be able to gather meaningful data. Over many weeks and months, and with many late nights working from home, the questionnaire was further developed, adding new question sections and increasing the sample. This proved challenging as not only were we making changes at a rapid pace, which was only possible thanks to Blaise 5, but we were also releasing new versions, which meant downloading a very large database from NISRA, moving collected data to the new structure (being careful to map old data to the new structure so that partials could still be completed), and packaging and reuploading to NISRA for them to install for us. The whole process with the increased sample could take 6 hours and was usually occurring every week or two.

4.2 Current LMS

At the start of 2021, as an office, it became clear that moving away from Blaise 5 was not going to happen anytime soon, and that the LMS collection in Blaise 5 had been a great success.

A demand was growing for a telephone mode LMS, and having recently been buoyed by the successful rollout of Blaise 5 training to Opinions and Lifestyle Survey staff, it was decided that Blaise 5 would be used for all questionnaires.

With plans to increase the Wave 1 sample to 12,000 cases when introducing CATI collection, it was clear that we couldn’t continue with the “One large database for the entire sample” approach we’d been struggling to maintain until this point. It was agreed that we would instead make a single questionnaire for each of the five new cohorts and start collection in any given week, which was later changed to be one Wave 1 questionnaire and one merged Wave 2–5 questionnaire each week. This required a new approach to the login questionnaire, which was developed to redirect to whichever GUID was listed alongside the Unique Access Code entered by the user. Security measures were taken to ensure that nobody could reverse engineer access to any of the cohort questionnaires directly by setting server-side variables and passing through checksum values.

A lot of work by David Kinnear was conducted to merge the SEL questionnaire into the main questionnaire and add a telephone layout that could easily be navigated by keyboard when run in CATI mode. The call treatment options of the SEL questionnaire were all added as parallel tabs of the new LMS questionnaire. This enabled us to have one questionnaire for both CAWI and CATI modes while also meeting the needs of the SEL.

A Data Collection Platform was created by our IT colleagues, which enabled us to install the Blaise questionnaires to their cloud servers for CATI collection and a nice front end was created for managing data and questionnaire removals. The stock Blaise 5 CATI Dashboard would be used for interviewers, interviewer managers, and SEL colleagues to log in and carry out their work after Roles and Skills had been set up. Unfortunately, we were not in a position to enable CAWI collection at this point, but it was planned that this Data Collection Platform would one day host our Blaise 5 web questionnaires.

> back to Table of Contents

With no CAWI hosting capacity, a further agreement was made with our colleagues at NISRA, who very kindly still host our LMS web questionnaire and have installed and hosted over 700 LMS questionnaires for us at this point—and a significant number of other surveys too!

Instead of downloads occurring every night as with the longitudinal test, the download frequency was increased to take place every hour to prevent chasing people who had just completed their questionnaire, with the full set of CAWI data being downloaded by an automated process and web completions and partials inserted into the CATI database. Every night a copy of the CATI data is output in the format requested by users along with a BDBX for post-collection processing to be run on.

A C# application "cloner" was developed to make short work of creating five, and eventually two, new questionnaires each week, which we now use for other surveys and is continually being developed. A separate app was created for users to run any post-collection processing, which is essentially a "Manipula Script runner" that accepts specific cohorts as input, enabling us to retire the Blaise 4 SEL questionnaire and post-collection processes entirely.

New versions of this questionnaire content are being continually developed as user needs change and a parallel run is currently taking place with a view to one day retire LFS once it's confirmed that data sufficiently meet customer needs.

4.3 Future LMS

The future of the LMS has Blaise colleagues at ONS very excited currently, as it was recently proved that the Data Collection Platform is able to host web questionnaires successfully, with a recent run of the National Survey for Wales data being conducted online via the platform.

The Data Collection Platform manages the creation of Unique Access Codes and has its own user login page, meaning we'll no longer have to maintain the Blaise 5 login questionnaire.

This means that alongside the further refinement and development of the current question set, we've been given the green light to host our own LMS CAWI questionnaire for the first time.

Work is currently being carried out at a rapid pace by David Kinnear to add a CAPI element for face-to-face interviewer collection, so that all three modes—CAWI, CATI, and CAPI—can be run on our first true mixed mode questionnaire, with all data being collected and managed in a single database. This is something we've dreamed of doing since we first saw Blaise 5 and it's very exciting that we're heading in that direction. We're aiming to achieve this goal at the start of 2024, so if anyone has experience doing this themselves and has any advice or tips to share, we'd be very grateful to chat with you.

5. Summary/Reflections

Being in a current state of transitioning from Blaise 4 to Blaise 5 gives a distinctly unique viewpoint. It is a good opportunity to review what worked well in the old system and what we would like to see in the new system. What didn't work well and should be left behind. What we look forward to using in the new system and features that we feel we would benefit greatly from, as well as those new available features that we are not so sure about yet, but in keeping an open mind, we may be convinced are an improved method or system.

6. References

Office for National Statistics. Methodology of LFS, Volume 1. https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/methodologies/labourforcesurveyuserguidance

Office for National Statistics, LCF Methodology.

> back to Table of Contents

https://www.ons.gov.uk/peoplepopulationandcommunity/personalandhouseholdfinances/expenditure/methodologies/livingcostsandfoodsurveytechnicalreportfinancialyearendingmarch2022/pdf

Office for National Statistics. LCF Survey. https://www.ons.gov.uk/peoplepopulationandcommunity/personalandhouseholdfinances/incomeandwealth/methodologies/livingcostsandfoodsurvey

> back to Table of Contents

An Electronic Life History Calendar in a Web Survey

Joseph Nofziger, Lilia Filippenko, Emilia Peytcheva, RTI International

1. Abstract

In surveys that collect data of experiences over multiple years of respondents' lives, respondents are often offered cues to help with the recall process. A life history calendar is one such tool that many organizations have used over time, including the National Survey of Family Growth (NSFG). Using the existing paper Life History Calendar as a starting point, RTI developed an electronic life history calendar and integrated it with a Blaise 5 instrument so that web respondents have the option to view and report significant events in a calendar format. When subsequently asked to place other events in time, respondents could refer to the calendar for context and easier recall. The electronic calendar can be displayed on demand at any time—or in specific sections, if so configured—either in full-screen mode or along with the Blaise question. Its responsive design allows for a display on any screen infAppendix

portrait or landscape orientation.

In iterative collaboration with our client, the National Center for Health Statistics (NCHS), we heavily customized an off-the-shelf chart and automatically populated the calendar with responses in real time. Events were put into temporal context with year, month, respondent age, and previously entered responses to help orient the user. The Blaise instrument was designed to easily collect information for effortless processing by the calendar application. In this presentation, we describe the architecture of the calendar and its implementation with the Blaise 5 Web instrument.

2. Background

Life History Calendars (LHC) improve recall for events of interest by linking to landmark events in a respondent's life (Belli et al., 2001). Historically, they have been used in in-person, interviewer-administered surveys. With the move to web and mixed-mode data collections, surveys like the NSFG that utilize LHCs needed an alternative for the web administration.

NCHS began conducting the NSFG 50 years ago. The survey collects data on reproductive and general health. Since 1995, it has used in-person CAPI data collection with instruments programmed in various versions of Blaise. In January 2022, it moved to a mixed-mode design, including a self-administered web instrument designed to work in browsers and on mobile devices.

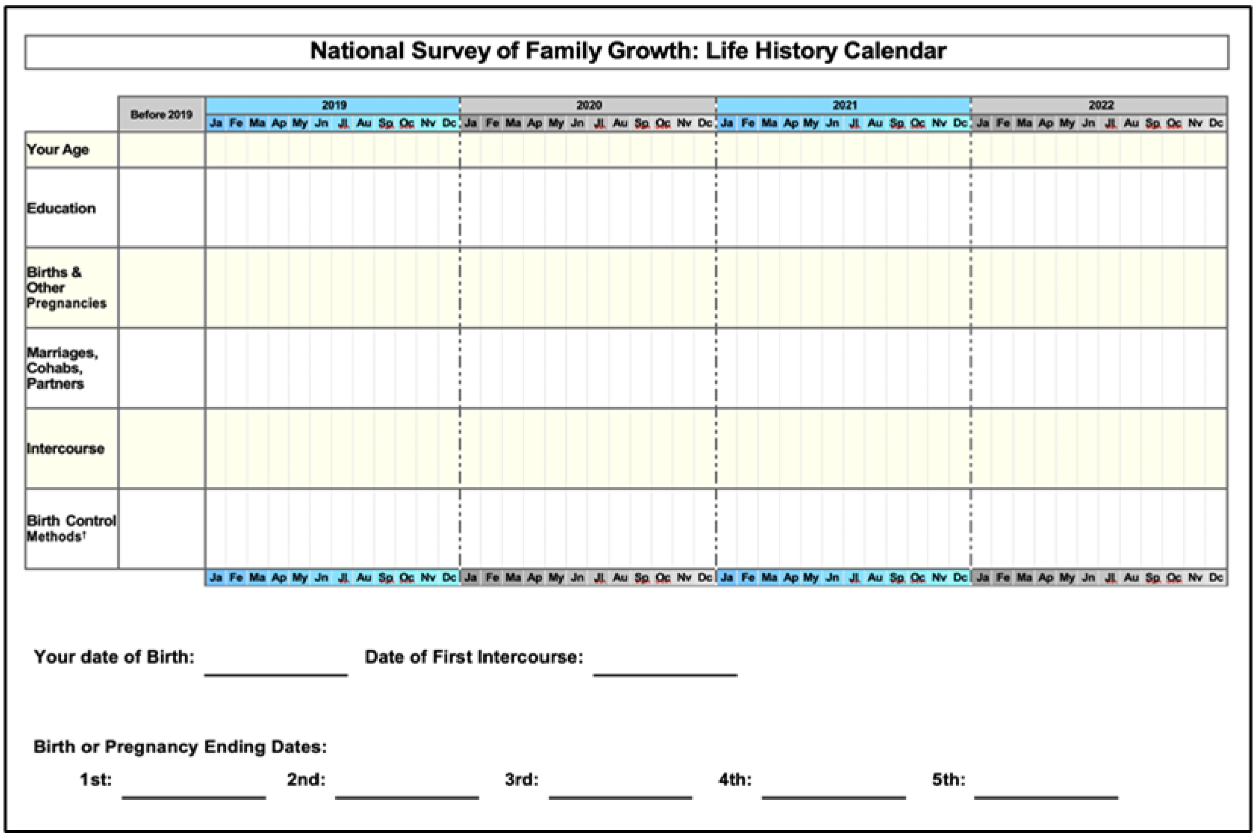
3. Technical Description

3.1 LHC Implementation

The NSFG instrument collects detailed information on events, such as pregnancy outcomes, sexual activity, and contraceptive use, by month over the past four years. Figure 1 shows the *paper* LHC used in face-to-face interviews for 2022.

> back to Table of Contents

Figure 1. NSFG Paper Life History Calendar



National Survey of Family Growth: Life History Calendar

	Before 2019	2019	2020	2021	2022
		Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc	Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc	Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc	Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc
Your Age					
Education					
Births & Other Pregnancies					
Marriages, Cohabs, Partners					
Intercourse					
Birth Control Methods†					
		Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc	Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc	Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc	Ja Fe Ma Ap My Jn Jl Au Sp Oc Nv Dc

Your date of Birth: ____________ Date of First Intercourse: ____________

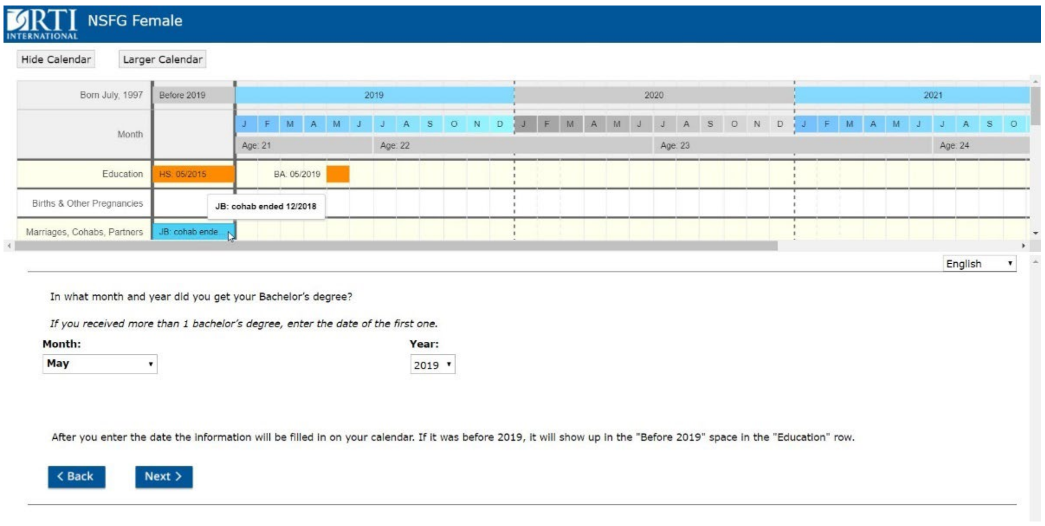
Birth or Pregnancy Ending Dates:

1st: ____________ 2nd: ____________ 3rd: ____________ 4th: ____________ 5th: ____________

Years and months appear across the top and question categories along the left side, forming a grid in which respondents can mark life events. The calendar covers the current year and three previous years. A column toward the left labelled "before 2019" is for recording relevant events that took place prior to the reference period of interest.

Figure 2 is an example of our *electronic* version incorporated into the NSFG Blaise 5 questionnaire.

Figure 2. NSFG Electronic Life History Calendar Embedded with Blaise Questionnaire



Born July, 1997	Before 2019	2019	2020	2021
Month		J F M A M J J A S O N D	J F M A M J J A S O N D	J F M A M J J A S O
		Age: 21 … Age: 22	… Age: 23	… Age: 24
Education	HS: 05/2015	BA: 05/2019		
Births & Other Pregnancies				
Marriages, Cohabs, Partners	JB: cohab ende			

JB: cohab ended 12/2018

In what month and year did you get your Bachelor's degree?

If you received more than 1 bachelor's degree, enter the date of the first one.

Month: May

Year: 2019

After you enter the date the information will be filled in on your calendar. If it was before 2019, it will show up in the "Before 2019" space in the "Education" row.

> back to Table of Contents

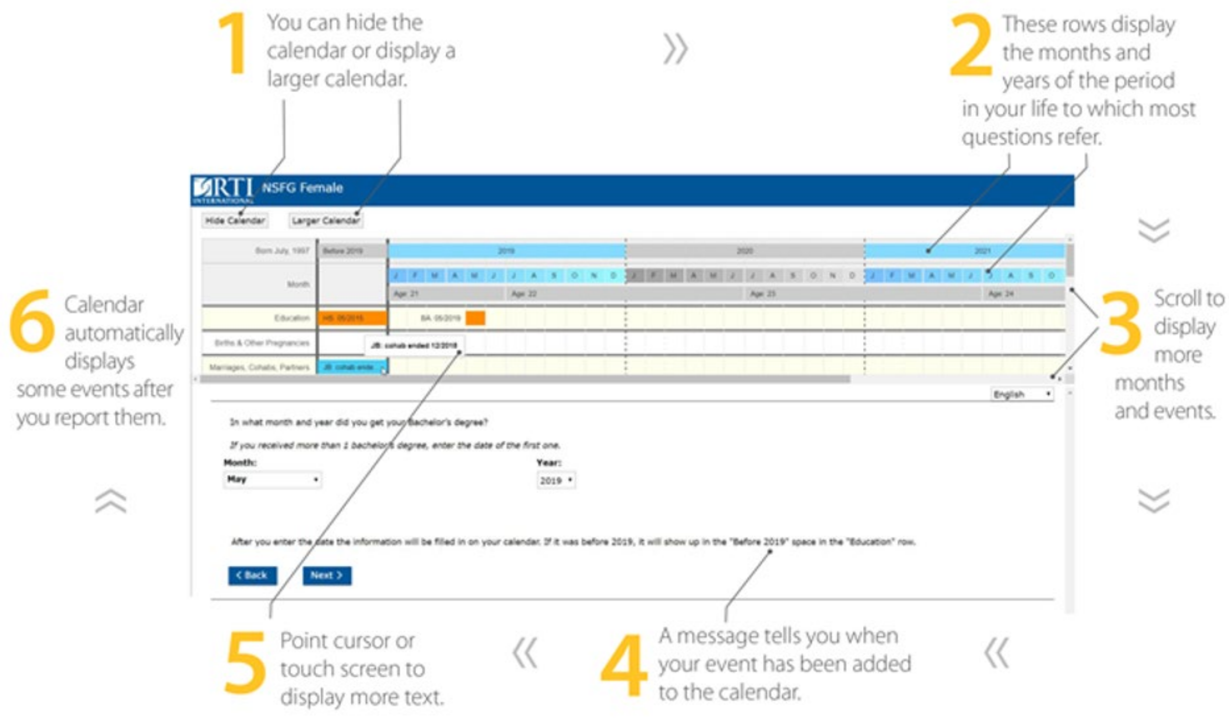
Everything below the central scroll bar is programmed in Blaise, while above the bar is custom. A design decision was made to have the calendar be *above* the question and scrollable, so that the calendar content is large enough to read but does not take all the screen space. The electronic calendar is not a data entry tool, but rather a graphical representation of responses that are entered into the Blaise survey instrument and that correspond to the questions presented on each screen with the calendar view. Every part of the screen is also available in Spanish—buttons, labels, month abbreviations, etc.

A tutorial, shown in Figure 3, is presented to respondents to help them understand how they can control and use the calendar. It has many elements designed to replicate the paper version, and some enhancements that are only possible with an electronic version. Of note, at the upper right is one control to hide/show the calendar and another to display it in full screen (i.e., without any Blaise content). Along the top are the "guide rows" with years and months, similar to the paper version. The respondent's age is shown and updates according to their date of birth. The "Before" column is also present as in the paper version. The years displayed adjust according to the current year.

For visibility, a design decision was made to display only a portion of the calendar by default. Scroll bars allow viewing more months and question categories.

As shown in Step 5, calendar items have hover text for further content. The calendar events stay visible for the remainder of the questionnaire, including if the respondent backs up.

Figure 3. Respondent Instructions


1 You can hide the calendar or display a larger calendar.
2 These rows display the months and years of the period in your life to which most questions refer.
3 Scroll to display more months and events.
4 A message tells you when your event has been added to the calendar.
5 Point cursor or touch screen to display more text.
6 Calendar automatically displays some events after you report them.
RTI INTERNATIONAL
NSFG Female
Hide Calendar
Larger Calendar
Born July, 1997
Before 2019
2019
2020
2021
Month
Age 21
Age 22
Age 23
Age 24
Education
HS: 05/2015
BA: 05/2019
Births & Other Pregnancies
JB: cohab ended 12/2018
Marriages, Cohabs, Partners
JB: cohab ende
English
< Back
Next >

In what month and year did you get your bachelor's degree?

If you received more than 1 bachelor's degree, enter the date of the first one.

Month: May

Year: 2019

After you enter the date the information will be filled in on your calendar. If it was before 2019, it will show up in the "Before 2019" space in the "Education" row.

3.2 LHC Architecture

The Blaise 5 site is a Single Page Application. We added custom elements and scripts to that page to house and control the calendar. The LHC itself is hosted in a separate website on the same server, which includes the chart. This is partly because Blaise 5 and the LHC have different reload requirements. For example, changing languages requires us to reload the chart, but Blaise updates itself without reloading

> back to Table of Contents

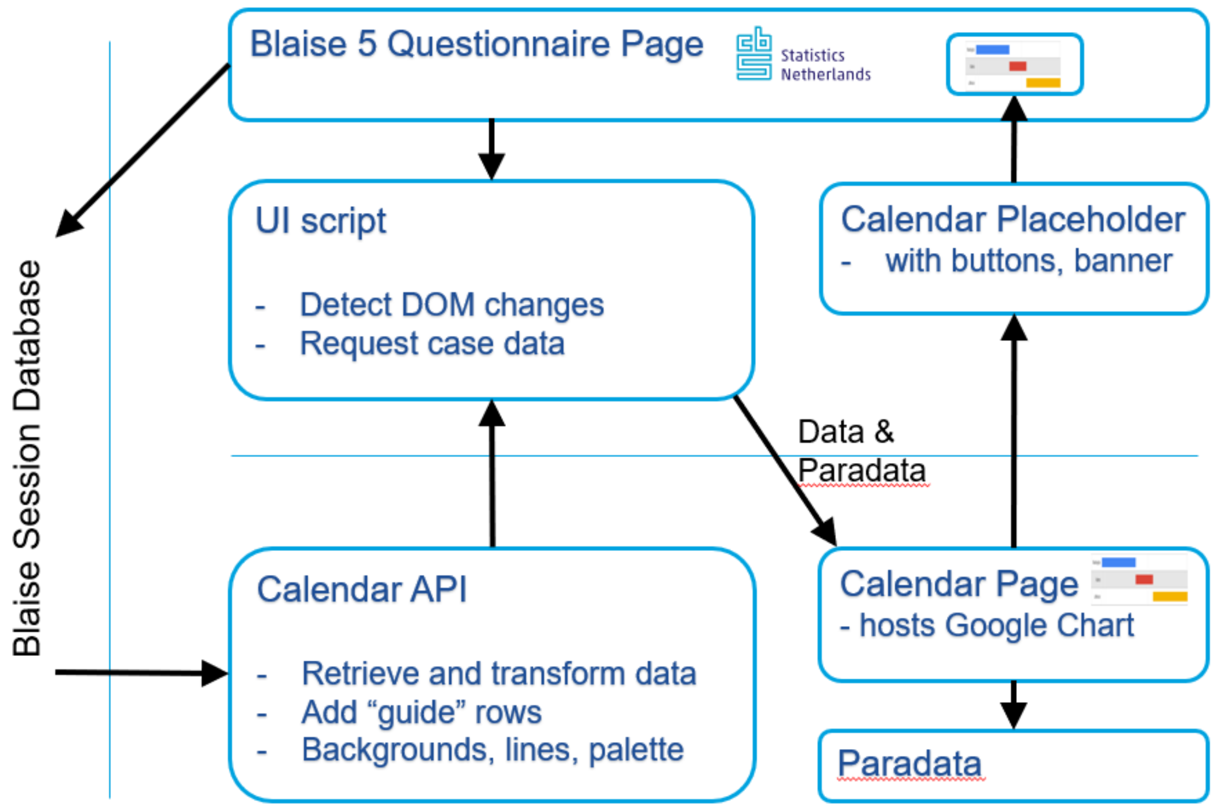
the Blaise page. It also maintains a separation of concerns—of the user interface controls and monitoring versus the chart functionality.

Figure 4 represents the structure. At the top of the diagram is the questionnaire. Above the center line are changes we made to the webpage that Blaise generates. We create a placeholder (upper right) where the calendar can be inserted, along with the custom buttons. The UI script (at center left) monitors the Blaise page for changes to the item, page, case ID, language selection, and so on. It then requests updated data from the calendar application (lower left).

Below the line is the separate calendar application. It gets data from the Blaise database and transforms it, adds data for "guide" rows (such as years and months), and performs several UI customizations, including grid line weights, styles, and colors.

The Calendar Page (lower right) is primarily a container for the chart, but it also receives messages and writes logs to a paradata file when the buttons are used.

Figure 4. Application Structure



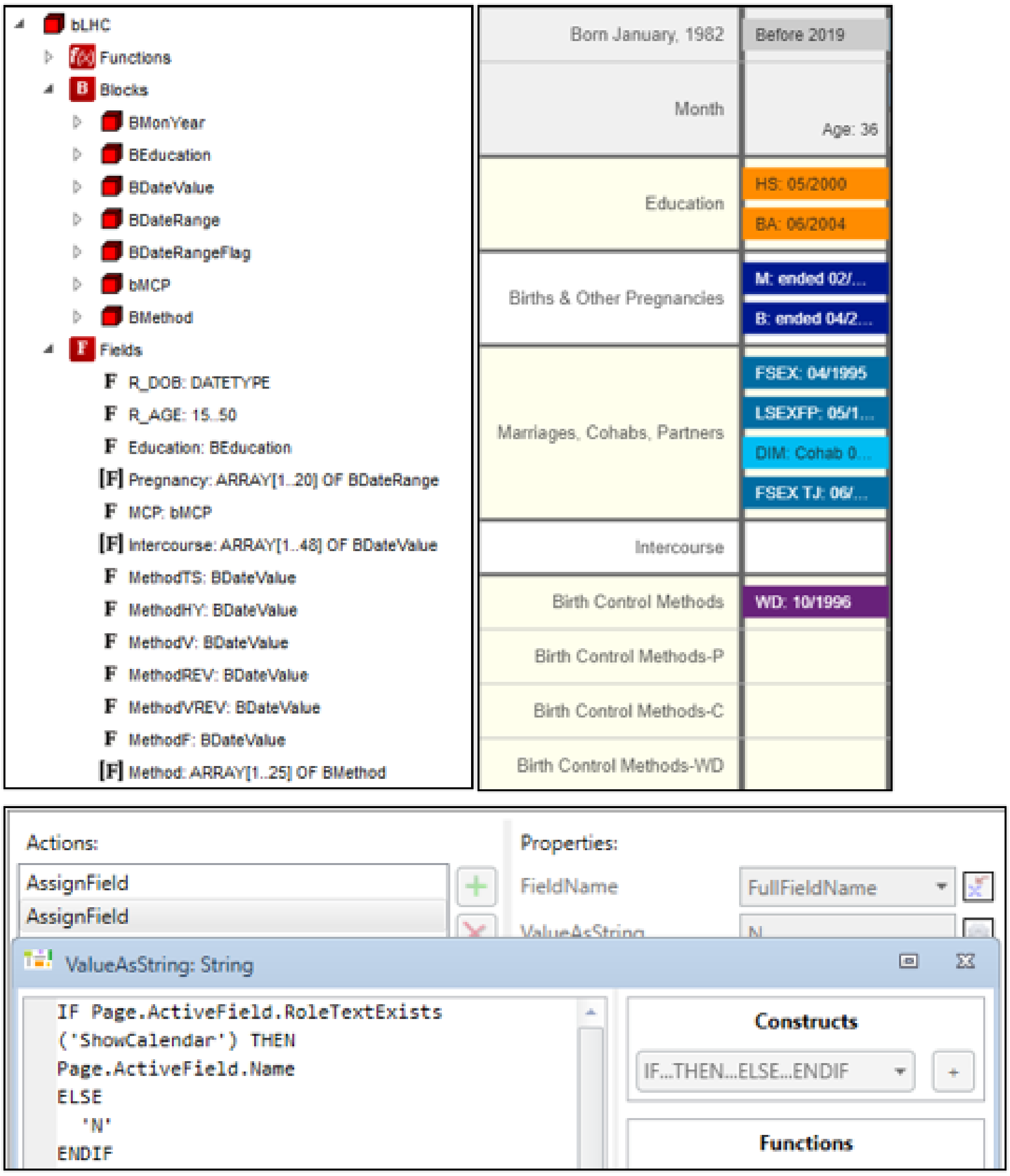
3.3 Blaise Interaction—Data & Design

A few factors are important to the interaction of the LHC with the Blaise instrument. First, it is essential to use the Blaise session database. Only the session database has the latest values as they are being entered.

> back to Table of Contents

Second, Blaise offers API calls to request values by item, such as with GetBlaiseItem(). For comparatively large numbers of items, such as the use of specific contraceptives by month over a period of years, we found this to be too slow. Instead, we access an entire block of items in one call to GetField() and loop through them if the structure is known. For that reason, having a reliable data structure is critical. The items displayed in the calendar are spread throughout the questionnaire, and questionnaires change over time. Since we want a stable interface between Blaise and the calendar, we created a Blaise block named "LHC" (Figure 5) that has copies of the values the calendar will need to access. Any time one of those items is answered, its value is copied to the LHC block as part of the Blaise programming. This way, they are in a reliable location in the database.

Figure 5. Blaise Block Defined for Calendar Interaction



bLHC
- Functions
- Blocks
  - BMonYear
  - BEducation
  - BDateValue
  - BDateRange
  - BDateRangeFlag
  - bMCP
  - BMethod
- Fields
  - R_DOB: DATETYPE
  - R_AGE: 15..50
  - Education: BEducation
  - Pregnancy: ARRAY[1..20] OF BDateRange
  - MCP: bMCP
  - Intercourse: ARRAY[1..48] OF BDateValue
  - MethodTS: BDateValue
  - MethodHY: BDateValue
  - MethodV: BDateValue
  - MethodREV: BDateValue
  - MethodVREV: BDateValue
  - MethodF: BDateValue
  - Method: ARRAY[1..25] OF BMethod

Born January, 1982	Before 2019
Month	Age: 36
Education	HS: 05/2000 BA: 06/2004
Births & Other Pregnancies	M: ended 02/... B: ended 04/2...
Marriages, Cohabs, Partners	FSEX: 04/1995 LSEXFP: 05/1... DIM: Cohab 0... FSEX TJ: 06/...
Intercourse	
Birth Control Methods	WD: 10/1996
Birth Control Methods-P	
Birth Control Methods-C	
Birth Control Methods-WD	

Actions:
AssignField
AssignField

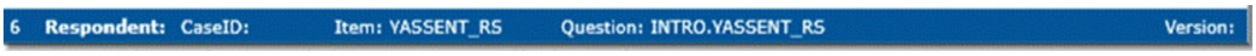
Properties:
FieldName: FullFieldName
ValueAsString: N

ValueAsString: String

```
IF Page.ActiveField.RoleTextExists
('ShowCalendar') THEN
Page.ActiveField.Name
ELSE
  'N'
ENDIF
```

The Blaise Resource Database was adjusted to work smoothly with the LHC. As mentioned, the script monitors the Blaise page for changes. This is done via a footer that is invisible to the respondent. A visible version is shown in Figure 6. The script reads the footer to get the case ID, page and item IDs, and so on as they change. It also contains a directive of whether to display the entire calendar container (which, for example, should not be visible on the screen before the calendar is introduced to the respondent), including the show/hide and full-screen buttons.

Figure 6. Blaise Footer (Normally Invisible)



6 **Respondent:** CaseID: Item: YASSENT_RS Question: INTRO.YASSENT_RS Version:

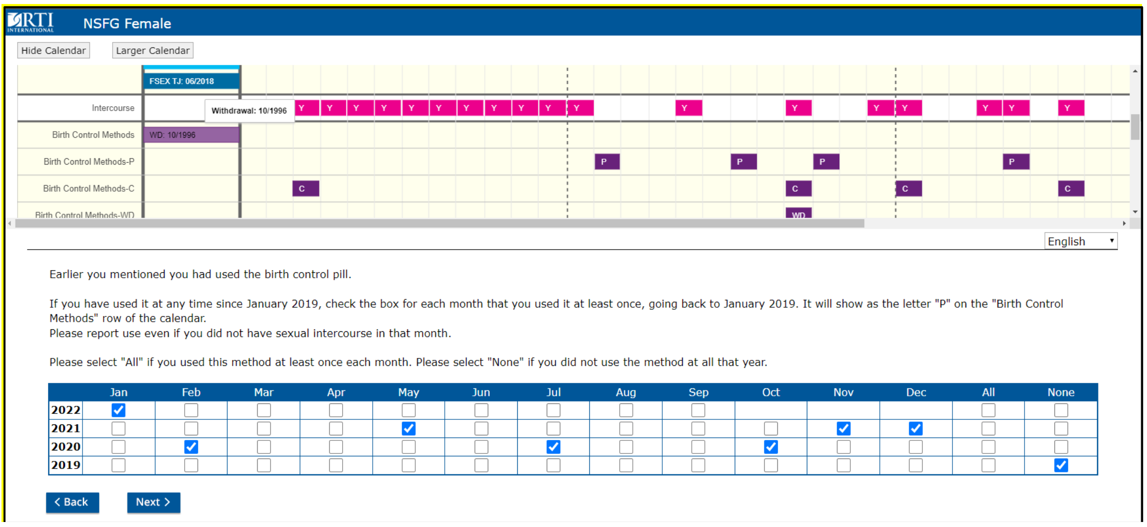
3.4 Collecting and Displaying Many Data Points across Years

One section in the instrument collects data for up to 48 months: every month in the past three years and up to the interview month in the current year. A set of more than 20 questions is asked about contraceptive use in the respondent's life. To facilitate this task, the respondent is presented with a grid where she clicks the box for the month(s) with sexual activities, and a checkmark appears in the box. Then, for each selected contraceptive method, the respondent is asked to mark the months when it was used (Figure 7).

A Blaise special procedure was developed to define columns and rows in the grid for the question. A hard check is triggered in that procedure if any inconsistency is detected between the selected month(s) and the previously reported sexual activity. An additional check may be executed if the respondent was pregnant in the selected month(s). The LHC is updated with the collected information and the respondent can see all birth control methods in one place, each on its own row, with letters designating the methods used during the given month (e.g., "P" for pill, "C" for condom).

The Blaise procedure assigns row names for each specific year, replacing the standard row titles and column numbers that are normally assigned. The procedure is also responsible for assigning data to fields in the LHC. With supporting modifications to the Resource Database, the procedure selectively hides checkboxes depending on the current date and respondent age; that is, for future months in the row for the current year (as in the top row of the table in Figure 7), as well as for past months when the respondent has already indicated she was not using contraceptives.

Figure 7. Condensed Interface for Collecting and Displaying Many Data Items



Earlier you mentioned you had used the birth control pill.

If you have used it at any time since January 2019, check the box for each month that you used it at least once, going back to January 2019. It will show as the letter "P" on the "Birth Control Methods" row of the calendar.
Please report use even if you did not have sexual intercourse in that month.

Please select "All" if you used this method at least once each month. Please select "None" if you did not use the method at all that year.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	All	None
2022	✓	☐	☐	☐	☐	☐	☐	☐	☐				☐	☐
2021	☐	☐	☐	☐	✓	☐	☐	☐	☐	☐	✓	✓	☐	☐
2020	☐	✓	☐	☐	☐	☐	✓	☐	☐	✓	☐	☐	☐	☐
2019	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	✓

< Back   Next >

> back to Table of Contents

4. Electronic Calendar Evaluation

To better understand the use of a self-administered electronic LHC, and the life history calendar in general, we embedded debriefing questions at the end of the NSFG female instrument asked of both web and in-person interviewed respondents. Because mode of data collection is confounded with calendar mode, this design did not allow for the direct comparison of estimates between web and face-to-face (electronic vs. paper), but it presents an initial feasibility test that allows us to better assess the proportion of web respondents who utilized the calendar, where it was used, and its ease of use.

To supplement the respondent debriefing questions, we examined paradata, such as screen size and actions taken by respondents (e.g., minimize or enlarge the calendar, hide or show the calendar).

4.1 Paradata on Calendar Use

During the first year of data collection, we collected paradata on the usage of the show/hide and full-screen buttons. In Figure 8, some respondent actions are outlined for visibility. We can look at the "show" and "full-screen" buttons together as ways to view the calendar. As outlined in blue, the overall number of respondents who ever used the show or full-screen buttons or ever used the hide button were about the same. About half used both options (green outline). About 25% showed the calendar and never hid it, while about 25% hid it and never showed it.

Figure 8. Calendar Use—Showing and Hiding

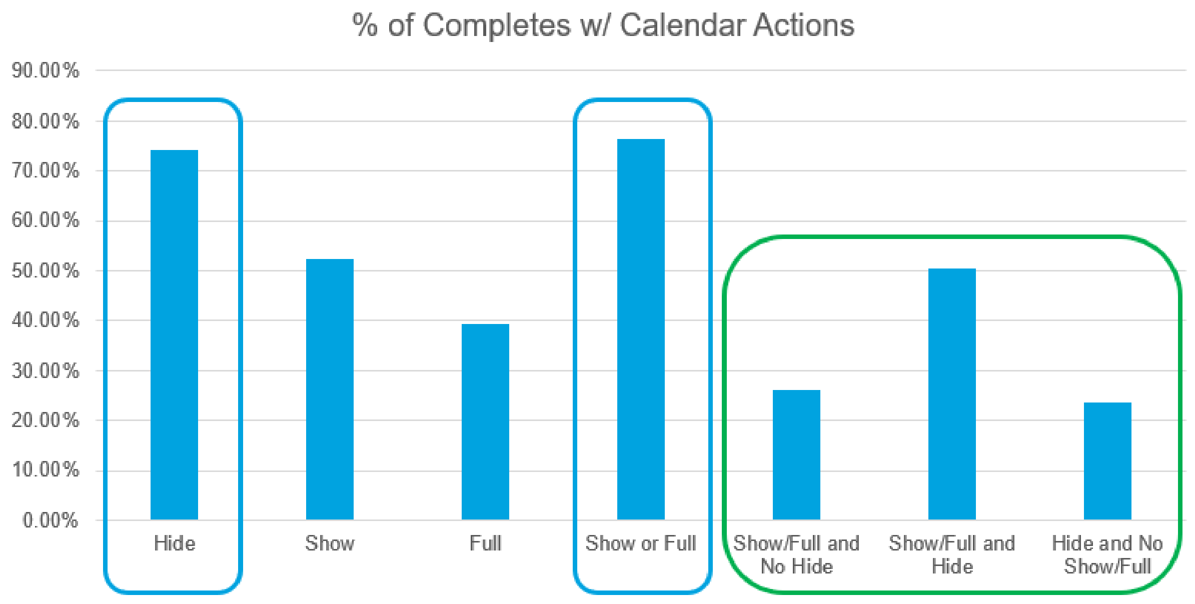
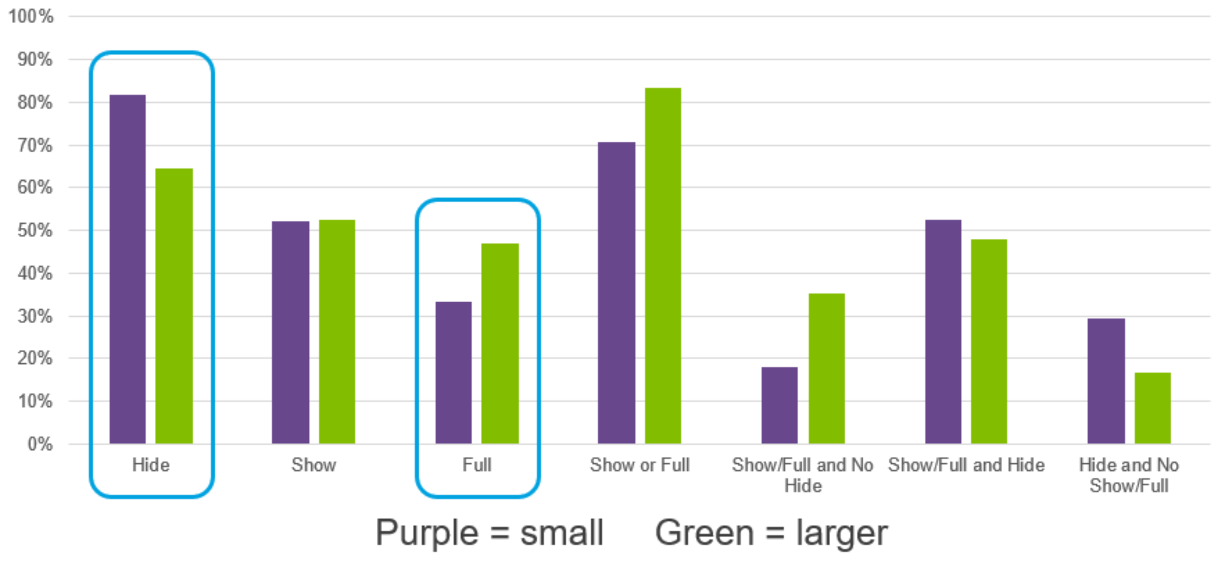


Figure 9 examines differences between screen sizes. Respondents using smaller screens were, predictably, more likely to hide the calendar and less likely to use the full-screen option. Here, small is defined as having a screen width of less than or equal to 600 pixels.

> back to Table of Contents

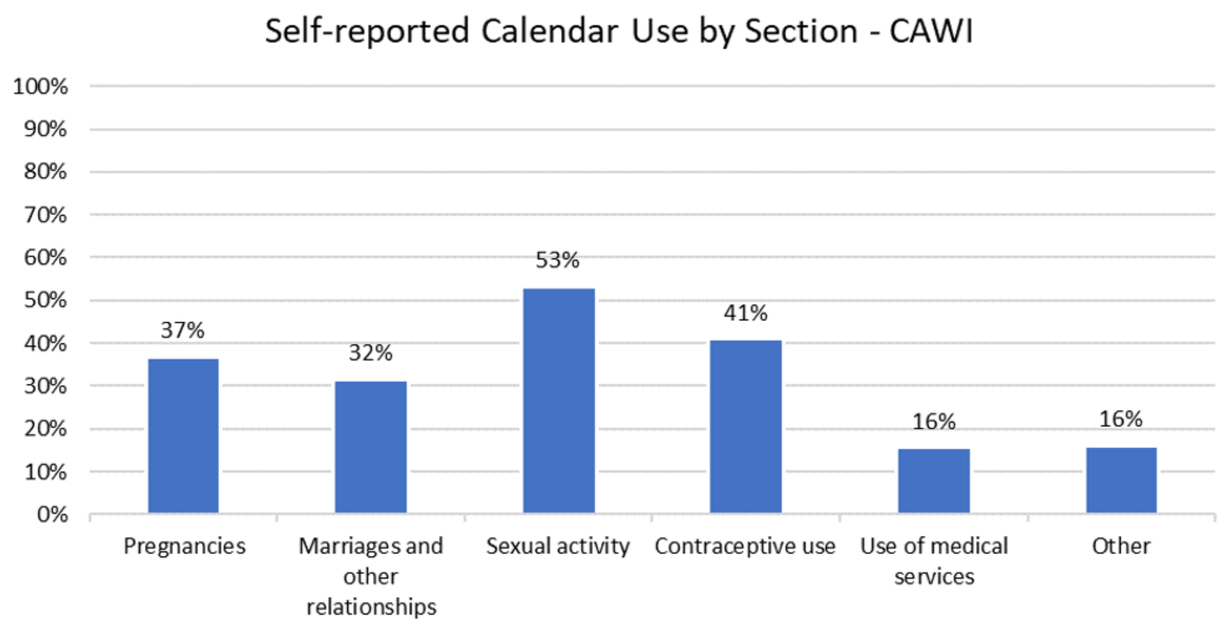
Figure 9. Calendar Use and Screen Size Comparison



4.2 Self-Reported Frequency of LHC Use

We asked respondents about the kinds of questions in which they found the calendar useful. Some results are presented in Figure 10.

Figure 10. Calendar Use by Section

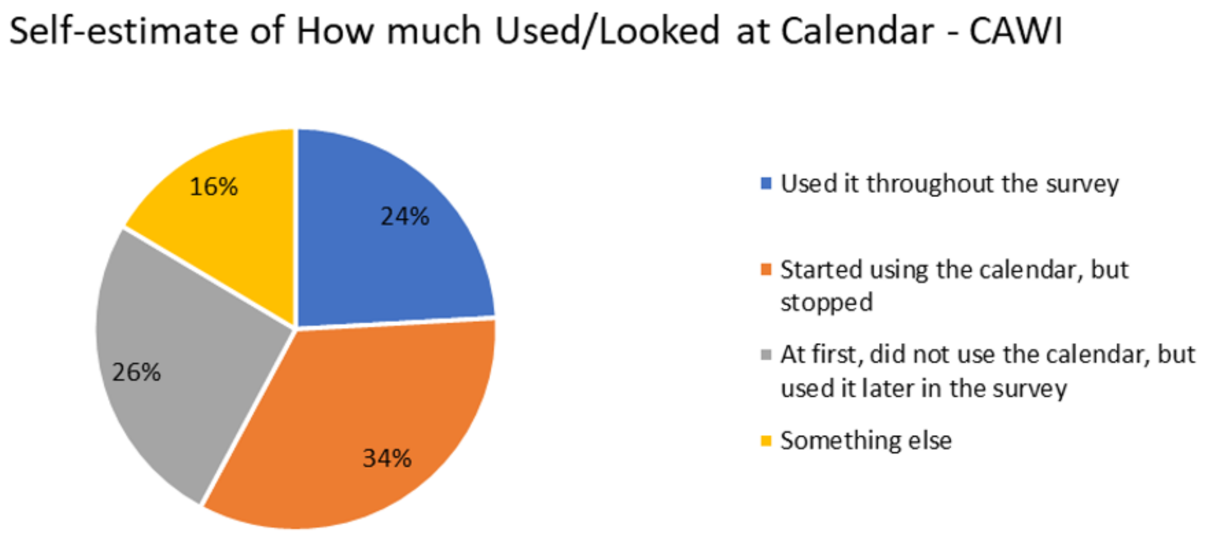


More than half reported using it in the section on sexual activity. About 40% used the LHC for questions about contraceptive use, followed by pregnancy and relationship sections in the 30% range.

> back to Table of Contents

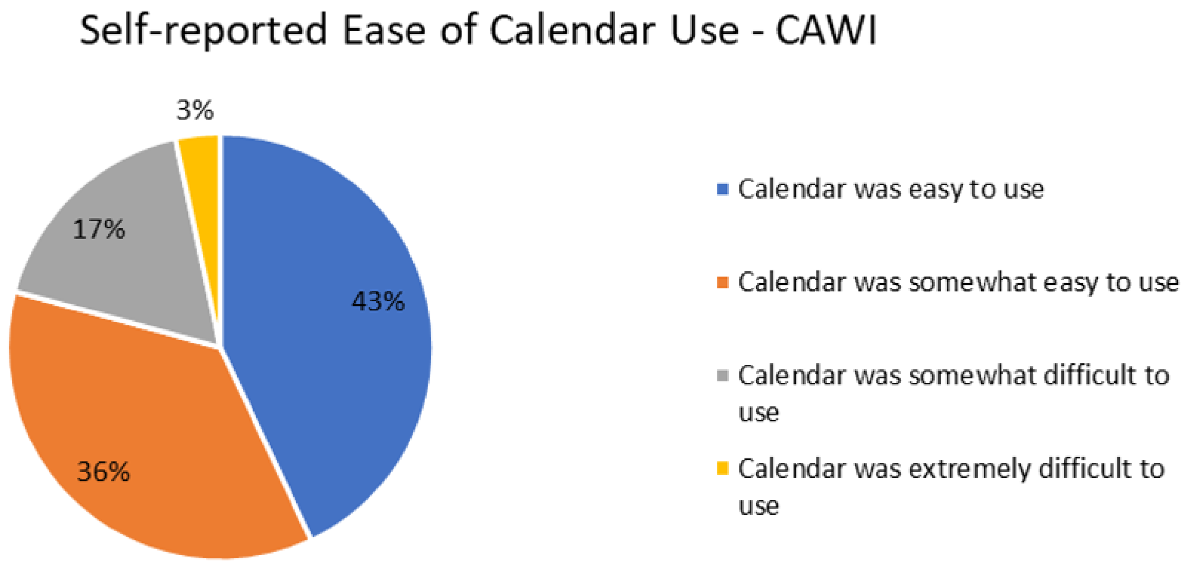
We also asked about the consistency of using the calendar (Figure 11). About a third of respondents (orange slice) started using it, and then stopped. But about a quarter (blue) used it throughout the survey and about another quarter (gray) didn't use it at first, then began using it later.

Figure 11. Calendar Use



Finally, we asked about the ease of use. Results are shown in Figure 12. Just 3% found it extremely difficult. While 17% found it somewhat difficult, 79% said it was either easy or somewhat easy to use.

Figure 12. Ease of Calendar Use



> back to Table of Contents

5. Conclusion

Blaise integration was successful, with customizations for usability and performance. We used the GetField() Blaise API with indexing on larger blocks of data for improved performance. Custom data structures storing copies of calendar data allowed us to align and freeze the programmatic interface between the Blaise and calendar APIs.

Customizations to the Blaise instrument and environment enabled better user interface design and improved integration with the calendar. Enhancements to the Blaise Resource Database control the visibility of columns in grid data entry. Special Blaise procedures control the display of data entry elements and assign data to specific fields.

While respondents sitting in person with an interviewer reported using the LHC significantly more than CAWI respondents (34% reported not using the calendar in year one vs. 78% on the web), the LHC allowed us to continue providing respondents with this memory aid during that period, as well as for the ongoing data collection in which nearly 75% of interviews are completed on the web.

Those who reported using the electronic calendar mostly utilized it in the sections on sexual activity and contraceptive use, followed by pregnancies and marriages/cohabitation. We detected significant differences of calendar use by age, ethnicity, and education, with the youngest respondents (15–18 years old) being the least likely to use the calendar, those with higher education being more likely to use their own calendar or app, and Hispanics being more likely to use the calendar.

The electronic LHC seemed to be used more sporadically by respondents relative to the paper calendar—more than half of the CAWI respondents (60%) reported using the calendar at some point, but not throughout the survey. In contrast, only 38% of CAPI respondents reported using the calendar at some point, and at least half used it throughout the survey.

6. References

Belli, R. F., Shay, W. L., & Stafford, F. P. (2001). Event history calendars and question list surveys: A direct comparison of interviewing methods. *Public Opinion Quarterly*, *65*(1), 45–74.

> back to Table of Contents

System To System Communication

Ralph van Geenen, Statistics Netherlands
WWW.CBS.NL/PS
ANNUAL / PRODUCTION STATISTICS RGS

1. Abstract

Save yourself time!

For the business questionnaire ‘Production Statistics,’ we have created a functionality that imports data into the Blaise questionnaire. The data is existing data in the general accountancy software (e.g., ‘exact Online’). This program can generate a ‘general calculating schema’ file (RCSFI file), which contains 40 to 60 percent of the information we are looking for.

Since April 2023, we have implemented a second method for importing data. More specifically, we use a third-party website, which can directly log in to several accountancy software packages. This implies that the manual import of an XML file is not needed anymore.

1.1 The Old Technique

- First, we use the ‘upload’ functionality of Blaise 5.8 and store the XML file in a ‘blob’ datatype.
- Second, we use a separate (WCF) service with use of the ‘alien’ function of Blaise to store that data on an ‘internal’ CBS server.
- Third, the service transforms the information into PS variables. The RGS data contains elements that have a 1:N relation to our information request. The service cumulates the information.
- Fourth, we use another (WCF) service to retrieve the information and convert it into a variable in the questionnaire.
- Finally, we present a recap of the imported data. In case the respondent thinks: “No! This isn’t correct,” he has the option to either clear the questionnaire and try again or choose not to utilize this functionality.

1.2 The New Technique

The respondent selects his accountancy package from the list, logs in to the package, gives permission to use the data, and finally ‘imports’ it into the questionnaire.

1.3 Presentation

In the presentation, I will provide a demo of the questionnaire. If someone is interested in the code, we can schedule an appointment to further discuss this.

> back to Table of Contents

2. Introduction

2.1 The RCSFI Import into the Blaise Questionnaire/Response Burden

The thought behind this is to simplify this process for the respondent. Due to the implementation of this functionality, respondents now have the capability to import data, thereby alleviating the need for manual data entry. This new approach is more based on the bookkeeping perspective.

2.2 Historical Background

Several years ago, CBS considered the utilization of bookkeeping data in questionnaires.
Since 2016, there has been and continues to be a national standard of business reporting (SBR). This, however, is not directly usable for questionnaires. Therefore, the Reference Classification System of Financial Information (RCSFI) was implemented.

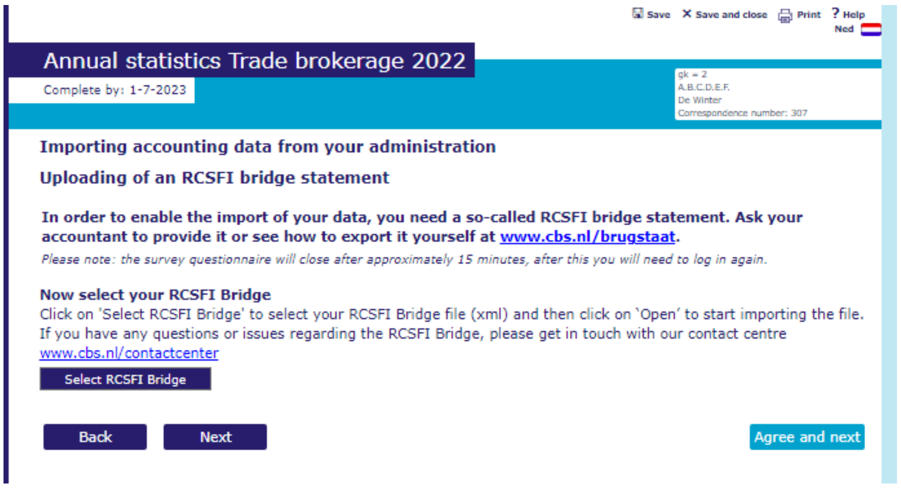
- RCSFI, or Referentie GrootboekSchema (RGS) in Dutch
- Developed in a public-private partnership as a nonmandatory standard in order to integrate and automate administrative processes
- In the Netherlands, there is no legal prescribed format for bookkeeping—only specific reports
- RCSFI contains references for all ledgers for different reports of the Dutch government
- Furthermore, it can also be used for all kinds of reports, dashboards, KPIs, etc.
- RCSFI codes are (a part of) variables in the questionnaire
- A cross-table links all RCSFI codes to a variable in the questionnaire

In 2020, the RCSFI was implemented for the first time in the production statistics. The aim was to achieve a 75 percent prefilling rate. Unfortunately, the required and desired information for CBS was not available in the SBR/RGS standard. We, however, persevered and strongly believed that to ease the process for the respondent, everything that can be imported and used should be imported in the questionnaire.

> back to Table of Contents

2.3 2021

Figure 1. The Manual Import into The Questionnaire—Since PS20 (April 2021)



Annual statistics Trade brokerage 2022

Complete by: 1-7-2023

gk = 2
A.B.C.D.E.F.
De Winter
Correspondence number: 307

Importing accounting data from your administration

Uploading of an RCSFI bridge statement

In order to enable the import of your data, you need a so-called RCSFI bridge statement. Ask your accountant to provide it or see how to export it yourself at www.cbs.nl/brugstaat.

Please note: the survey questionnaire will close after approximately 15 minutes, after this you will need to log in again.

Now select your RCSFI Bridge
Click on 'Select RCSFI Bridge' to select your RCSFI Bridge file (xml) and then click on 'Open' to start importing the file.
If you have any questions or issues regarding the RCSFI Bridge, please get in touch with our contact centre www.cbs.nl/contactcenter

In the IBUD in April 2022, I presented this part.

1. Log in to the questionnaire and choose the option ‘import a RCSFI file.’
2. An instruction appears on the screen and tells you the steps that should be taken to complete the import.
3. In the bookkeeping software, you first have to export a file (RCSFI bridge) and save it on the desktop.
4. Go back to the questionnaire—if there was an automatic log off, log in again.
5. Push the button ‘select RCSFI Bridge.’
6. Select the file.
7. Import the file.
8. If successful, a summary appears.

2.4 2022

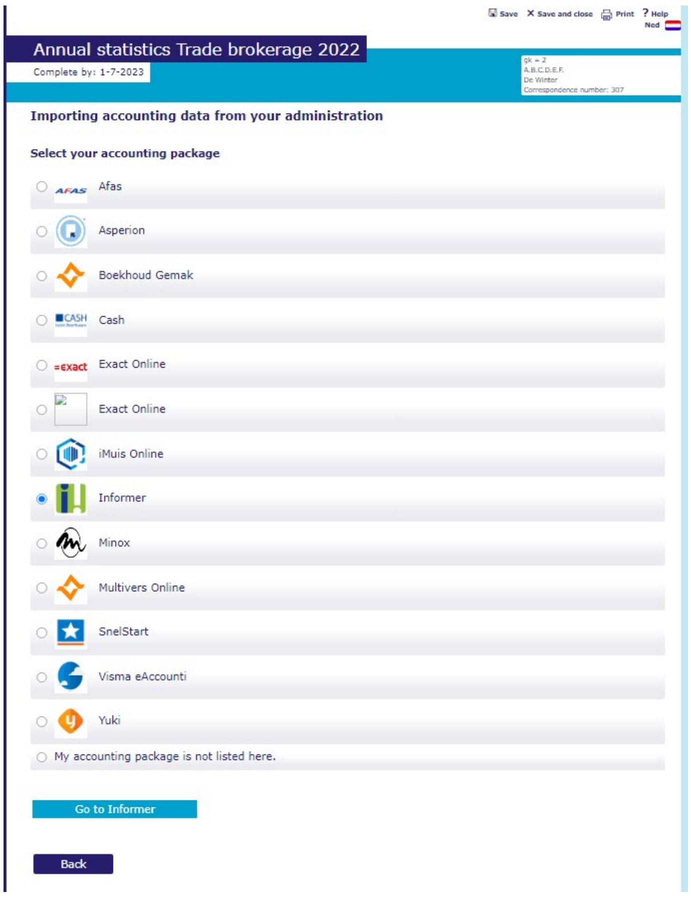
The manual import, more communication, and a video with instructions—Since PS21 (April 2022)

The same process as mentioned above, but with a video instruction included.

> back to Table of Contents

2.5 2023

The connected way—system to system—since PS22 (April 2023), a Dutch Transaction Ideal Method



Annual statistics Trade brokerage 2022

Complete by: 1-7-2023

gk = 2
A.B.C.D.E.F.
De Winter
Correspondence number: 307

Importing accounting data from your administration

Select your accounting package

- ( ) Afas
- ( ) Asperion
- ( ) Boekhoud Gemak
- ( ) Cash
- ( ) Exact Online
- ( ) Exact Online
- ( ) iMuis Online
- [✓ Informer]
- ( ) Minox
- ( ) Multivers Online
- ( ) SnelStart
- ( ) Visma eAccounti
- ( ) Yuki
- ( ) My accounting package is not listed here.

In this solution, the respondents can log in to their own bookkeeping software accounts and 'donate' their information to the questionnaire/Statistic Netherlands.

1. Choose your bookkeeping software. There are a few in the Netherlands. Almost all bookkeeping software businesses are supported.
2. Log in with your own credentials.
3. Select your bookkeeping (correct financial year, correct business in case of accountancy office).
4. Push the button—import.
5. If successful, you can go back to the questionnaire (close the tab) and refresh the questionnaire screen.
6. If successful, a summary appears.

3. Used Techniques

Videos: We have implemented an instruction video to show respondents how to correctly use the RCSFI. This video describes the way the data is exported from the bookkeeping software and imported into the questionnaire. Based on the results, we could conclude that none of the respondents watched the instruction video. Therefore, we do not use this anymore.

> back to Table of Contents

Alien Procedures: To send datafiles to a web service, to retrieve data from a web service.

Redirects: Redirect from web browser to the Speedbooks website. This includes some parameters for identification (e.g., respondent number, financial year, and the accountancy software package). This ensures that the Speedbook logs in to the right package.

WCF Services: This is the web service that processes the RCSFI data and converts the bookkeeping data into usable questionnaire data (key value pairs).

RESTAPI: A modern way of communication that is used between Speedbooks/bookkeeping and the RGS Service. In the future, this will also serve as a substitute technique for the WCF RGS Service.

4. Next Steps/Future Goals

Our aim is to use the shortened version of the PS for small businesses. This shortened version can be fully—or to be more precise, 100 percent—completed by the RGS import function.

The RCSFI file contains more information than we need in the questionnaire. This enables the statistician/analyst to use more specific and detailed information. However, this raises the following important question: Are we permitted to use all this specific information?

In the past, we also did other projects like this. One of them was the John Deere project. At that project, we connected to an external John Deere API (third party), did an 'OAuth Authentication' process within, and got farm data out of the cloud that directly came from the John Deere farm machines. This project was stopped because of the quality of the data. Nevertheless, the process was useful to us and to Blaise.

> back to Table of Contents

Replacing Manipula in Blaise 5

Siu Chong Wan and Ryan Webb, Westat

Presenter: Siu Chong Wan

We have done a lot of tricky things in Blaise 4 by calling Manipula procedures from within the instrument that the data model alone cannot handle, for example, showing dialog, calling a third-party program and importing its output into the in-memory data of the ongoing interview, and starting different instruments in order with conditions. In Blaise 5, a lot of these can be accomplished with the use of the resource database.

Although Blaise 5 Manipula is still a very powerful tool, Manipula process scripts cannot be used in browsers. This paper discusses how, with the resource database, hopefully we can have similar flexibility in both windows and browsers on some occasions.

1. Things We Tried So Far

1.1 Replacing Manipula Dialog

In Blaise 4, we used the Maniplus Dialog to display text information, display images and let the user browse through it, obtain information from the user, and enter data directly into a Blaise data set. To do this, we tied the Manipula procedure that defined the dialog box to the type in the data model properties.

In Blaise 5, with the resource database, is the Maniplus Dialog a necessity for this purpose? A Page Template can display extra elements with label control, image control, and even video control. This is useful if we do not want to maintain multiple projects and packages to run Manipula procedures. However, that is not to say that displaying a page template from the resource database can totally replicate the experience of displaying a Maniplus Dialog.

A Maniplus Dialog can overlap the data entry window as a second window; the use of a master page template or a custom page template leads to a new page in the same window. While both can display the same elements and collect the same data, visually they are two different experiences. With a Maniplus Dialog, the user can tell they are being diverted to a new window while they can still see the original data entry window in the background; with a new page (even supported by a different page template for a different look and feel) in the same window, everything just seems to be part of the same flow to the user. Where one's preference lies depends on the situation and the purpose of the page.

In addition, Maniplus Dialog still has the advantage of being able to define precise dialog height and width that we cannot do in a page template.

1.2 Running Third-Party Programs

In Blaise 4, we used the Maniplus Run function to execute external applications. To do so during runtime, we tied the Manipula procedure in the data model properties to the type that needs to run the third-party application.

The introduction of the StartLocalProcess action in Blaise 5.13 opened the door to a whole new world. The StartLocalProcess action is very handy for the purpose of running third-party applications. It eliminated the needs for a Manipula procedure to run external applications. In the middle of a survey, we can introduce something that cannot be programmed in Blaise, or something that simply can be programmed more efficiently outside of Blaise.

> back to Table of Contents

With the Blaise 5 SessionData API, we should be able to create external programs that pass data between the Blaise data in the current session and the external programs. What happens in Vegas does not have to stay in Vegas anymore. We might be able to bring home the money.

However, it is also worth noting that, like Manipula, the StartLocalProcess action does not work in browsers.

1.3 Flexible Routing

We always count on Blaise to follow the rules. In a Blaise survey, a field cannot be on route more than once in the rules. Rules are checked constantly to ensure data integrity. That is a strength. However, this can also become inflexible on occasion. When needed, we would like to not follow the rules.

In Blaise 4, the easiest we could do might be the INVOLVING instruction in an edit check. But that has to be presented as an error, not a choice. Of course, the next thing we could always count on was, again, Manipula because we could use the Maniplus instruction SETACTIVEFIELD.

In Blaise 5, with various resource database actions, it is relatively easy to change the flow of routing when necessary. For instance, rather than asking the respondent to move back many pages to change something, we can let them press a button to jump back to a certain point. With the GotoField action in Blaise 5, we can provide respondents with the opportunity to jump between questions without triggering any errors or maintaining Manipula scripts.

However, this happens while the rules are still running constantly. So we need to keep in mind that what we are doing outside of the rules still needs to be in compliance with the rules.

1.4 Running Multiple Surveys in Order

In Blaise 4, when we need to run multiple instruments together in a certain order, we would use the Maniplus Edit function/method to start multiple data entry sessions in one sitting. This usually happens outside of the instrument in Manipula.

In Blaise 5, we can use the StartSurvey action for this purpose without the help of Manipula. The StartSurvey action can be used in more than just the OnEnd event. Using the StartSurvey action in the OnEnd event, we can easily run multiple surveys from start to end in one order. Using the StartSurvey action in the OnLoad event of a page, or an event triggered by a layout control such as an OnClick event of a button, we do not have to wait until a survey is completed before the next survey starts. The current session will end and the survey data will be saved before the StartSurvey action is performed.

We can still see the value of controlling the flow of multiple surveys in Manipula with the Edit function/method if it is preferable to keep the programming of that flow in one place, that is, one Manipula project. However, using the StartSurvey action outside of Manipula gives us the flexibility of when, where, and how different surveys start consecutively. More significantly, the StartSurvey action also works in web surveys, while Manipula still cannot.

2. Examples

2.1 Replacing Manipula Dialog Boxes with Parallel Blocks

In Blaise 4, we used Maniplus Dialog to collect roster member information in a dialog box, where we could offer options of actions like add, edit, and delete. In this example, we use a combination of parallel blocks and resource database actions to collect a roster of medicines. This involves off-route flows that are transparent to the respondents.

> back to Table of Contents

2.1.1 Data Model

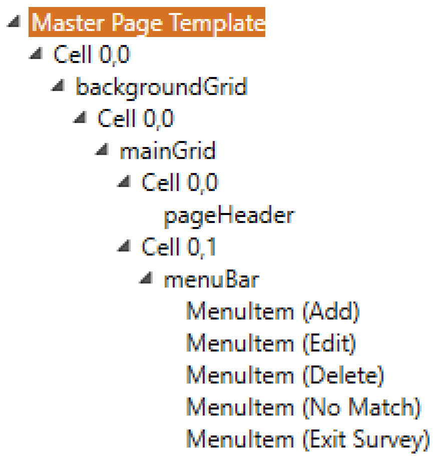
In the data model, we define a parallel block for each individual "dialog" path.

```
SETTINGS
   PARALLEL
      AddBlock
      EditBlock
      DeleteBlock
```

Each parallel block adds/edits/deletes medicines and saves the medicines back to the Main parallel block. As a result, the data collected in the parallel blocks are disposable.

2.1.2 Resource Database

In the resource database, we can use menu bar items, buttons, or keyboard shortcuts to provide access to the parallel blocks. In our example, we use menu bar items with a shortcut property to lead to the parallel blocks and other off-route flows.



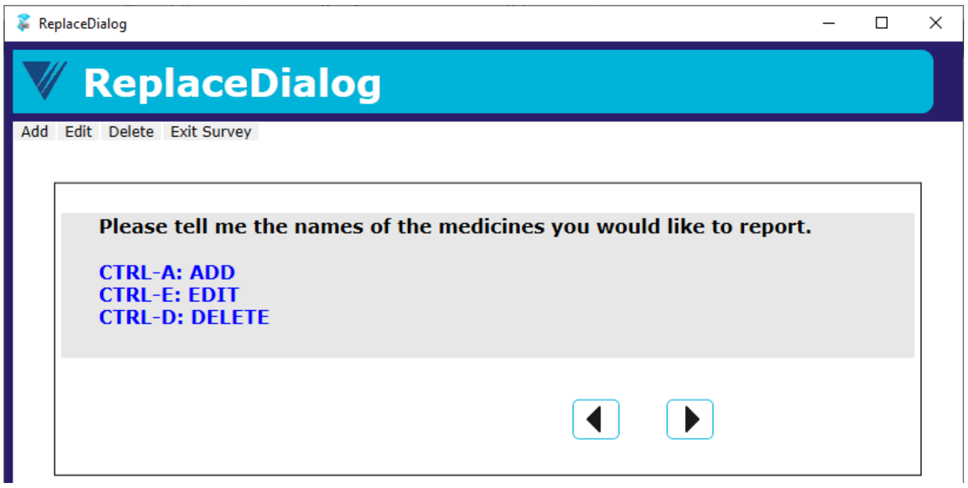
```
<MenuItem Text="Add" Shortcut="Ctrl+A" Visibility="{Expression IF
ENDSWITH(State.ActiveFieldName, 'Roster') THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action SetParallel('AddBlock')}" />

<MenuItem Text="Edit" Shortcut="Ctrl+E" Visibility="{Expression IF
ENDSWITH(State.ActiveFieldName, 'Roster') THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action SetParallel('EditBlock')}" />

<MenuItem Text="Delete" Shortcut="Ctrl+D" Visibility="{Expression IF
ENDSWITH(State.ActiveFieldName, 'Roster') THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action SetParallel('DeleteBlock')}" />

<MenuItem Text="No Match" Shortcut="Ctrl+Z" Visibility="{Expression IF
Page.ActiveField.TypeName = 'TLookupString' THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action AssignField('NoMatchButton', '1',
{Expression 1});Save();GotoField('AddBlock.PMedNoMatchString')}" />
```

> back to Table of Contents

Sending the user to a parallel block is the easy part. Clearing out the data from the parallel blocks while keeping the data in the Main parallel could take some effort. Of course, we can always create arrays of the parallel blocks so that each call of the parallel block is a new instance, rather than clearing out anything. But that also means we would be keeping duplicated data that are not necessary, unless it is desirable to keep the history of all the add/edit/delete instances.

2.1.3 Result

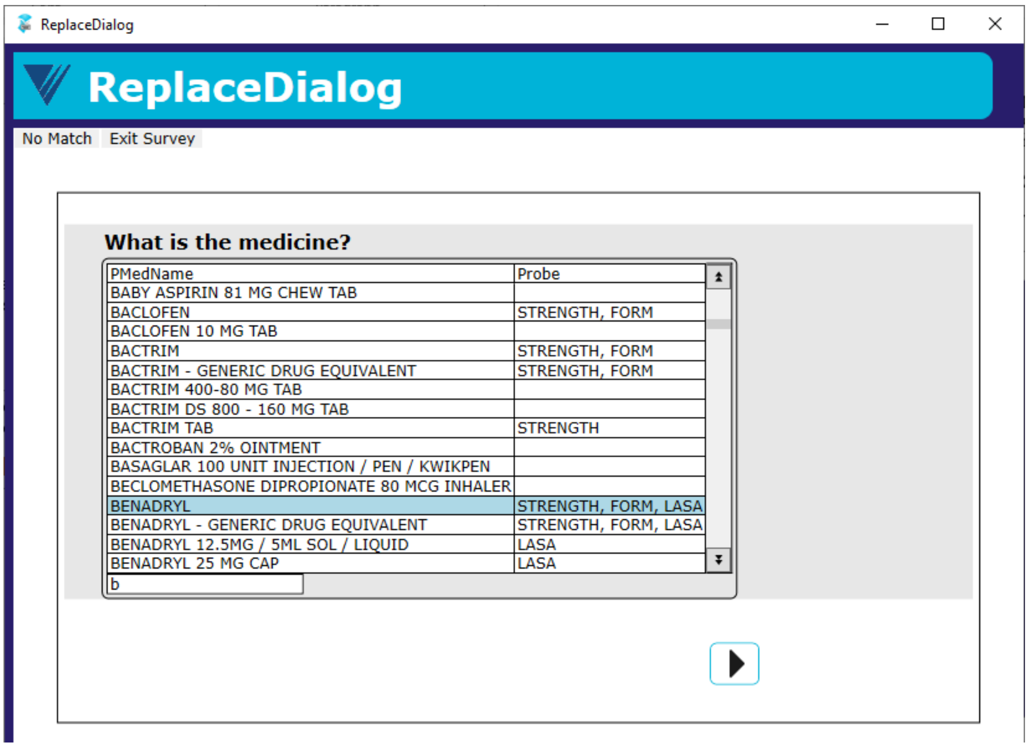
We first start in the Main parallel.



Please tell me the names of the medicines you would like to report.

CTRL-A: ADD
CTRL-E: EDIT
CTRL-D: DELETE

Either CTRL-A or the “Add” on the menu bar leads us to the parallel AddBlock.



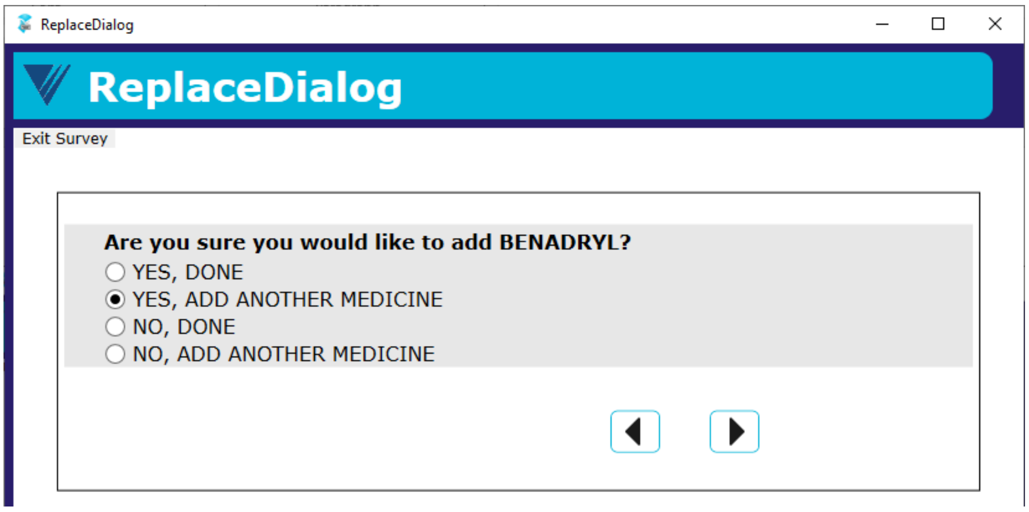
What is the medicine?

PMedName	Probe
BABY ASPIRIN 81 MG CHEW TAB	
BACLOFEN	STRENGTH, FORM
BACLOFEN 10 MG TAB	
BACTRIM	STRENGTH, FORM
BACTRIM - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
BACTRIM 400-80 MG TAB	
BACTRIM DS 800 - 160 MG TAB	
BACTRIM TAB	STRENGTH
BACTROBAN 2% OINTMENT	
BASAGLAR 100 UNIT INJECTION / PEN / KWIKPEN	
BECLOMETHASONE DIPROPIONATE 80 MCG INHALER	
BENADRYL	STRENGTH, FORM, LASA
BENADRYL - GENERIC DRUG EQUIVALENT	STRENGTH, FORM, LASA
BENADRYL 12.5MG / 5ML SOL / LIQUID	LASA
BENADRYL 25 MG CAP	LASA

b

> back to Table of Contents

Select BENADRYL and click the forward arrow.

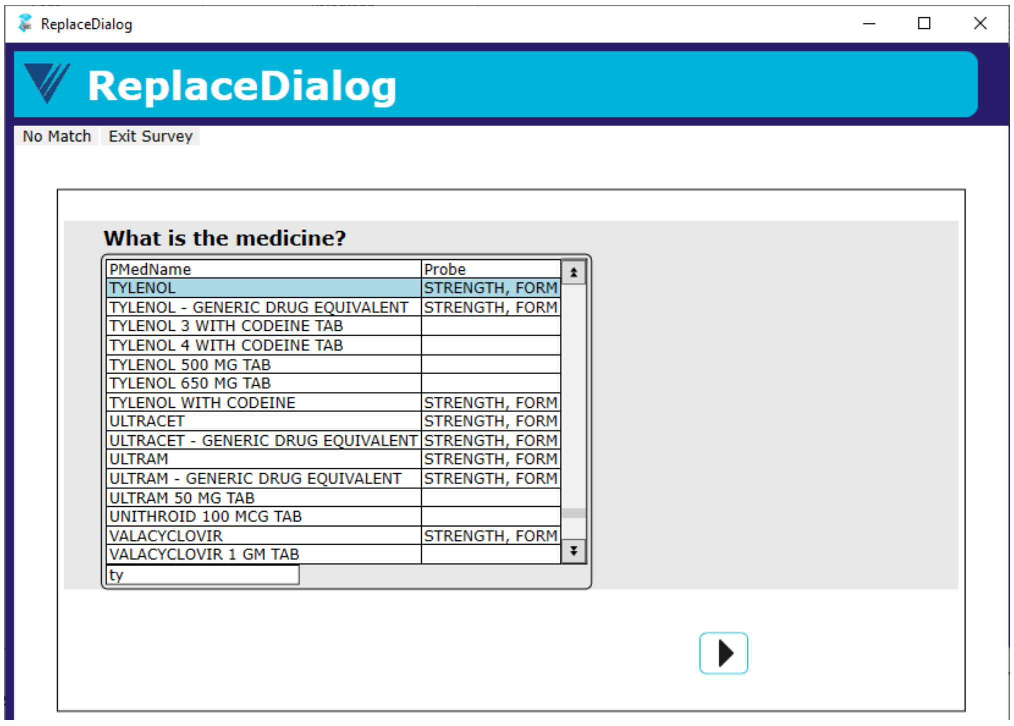

ReplaceDialog

Exit Survey

Are you sure you would like to add BENADRYL?

- ○ YES, DONE
- ● YES, ADD ANOTHER MEDICINE
- ○ NO, DONE
- ○ NO, ADD ANOTHER MEDICINE

Confirm adding and plan to add another medicine.


ReplaceDialog

No Match   Exit Survey

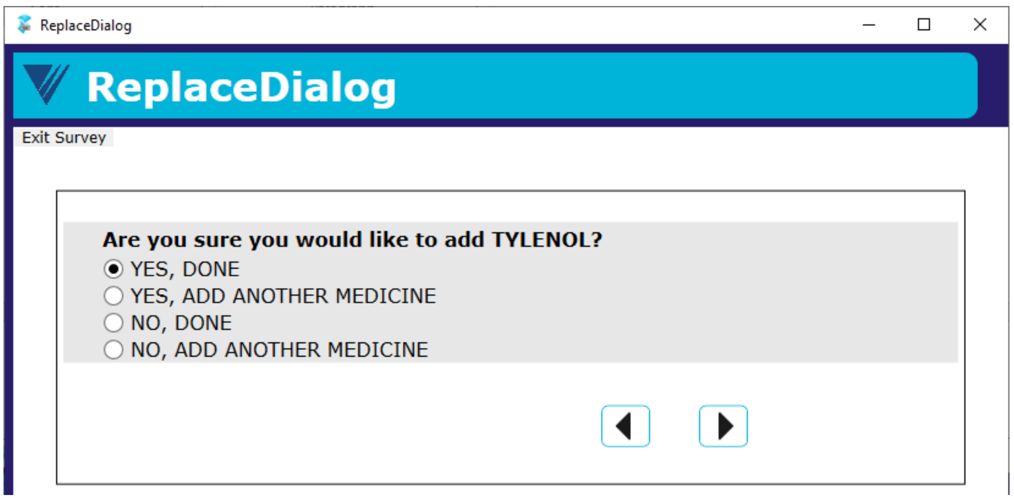
What is the medicine?

PMedName	Probe
TYLENOL	STRENGTH, FORM
TYLENOL - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
TYLENOL 3 WITH CODEINE TAB	
TYLENOL 4 WITH CODEINE TAB	
TYLENOL 500 MG TAB	
TYLENOL 650 MG TAB	
TYLENOL WITH CODEINE	STRENGTH, FORM
ULTRACET	STRENGTH, FORM
ULTRACET - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
ULTRAM	STRENGTH, FORM
ULTRAM - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
ULTRAM 50 MG TAB	
UNITHROID 100 MCG TAB	
VALACYCLOVIR	STRENGTH, FORM
VALACYCLOVIR 1 GM TAB	

ty

Select TYLENOL this time and move forward.

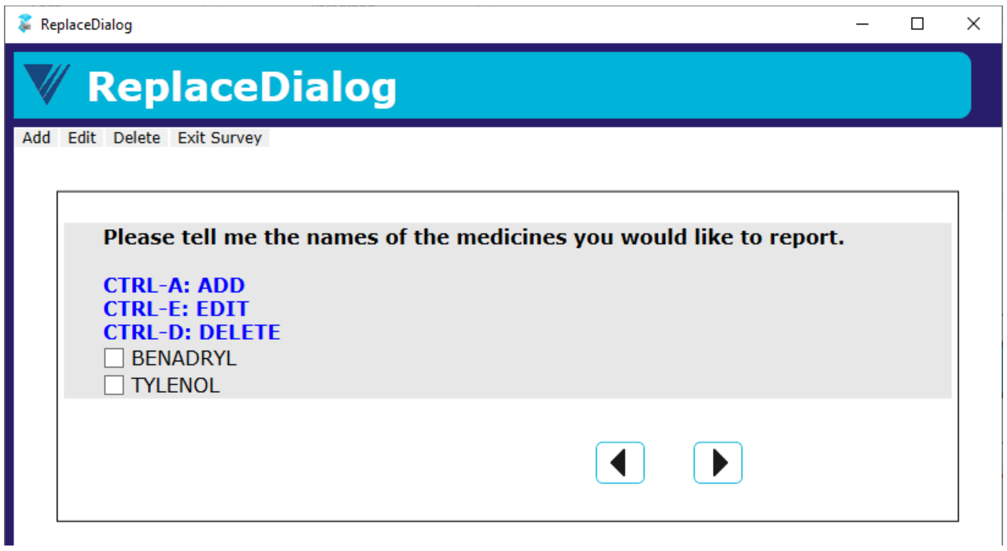
> back to Table of Contents



Are you sure you would like to add TYLENOL?

● YES, DONE
○ YES, ADD ANOTHER MEDICINE
○ NO, DONE
○ NO, ADD ANOTHER MEDICINE

The end of the parallel AddBlock leads us back to the Main parallel.

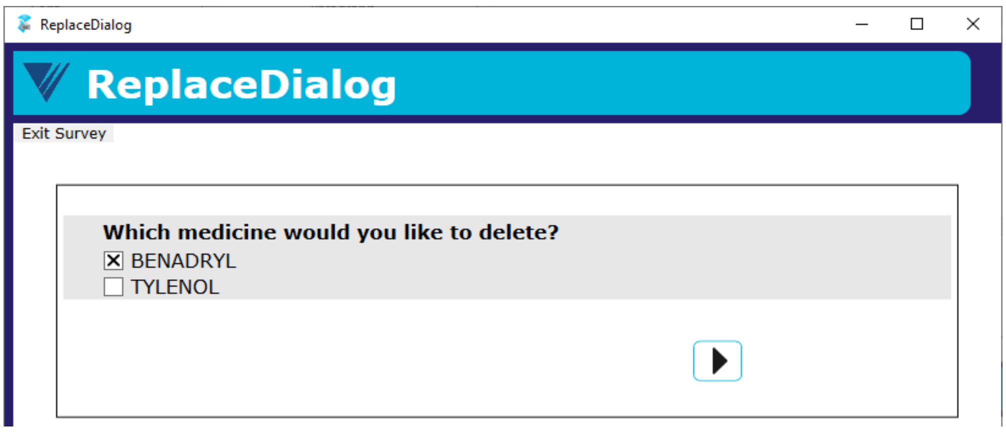


Please tell me the names of the medicines you would like to report.

CTRL-A: ADD
CTRL-E: EDIT
CTRL-D: DELETE
☐ BENADRYL
☐ TYLENOL

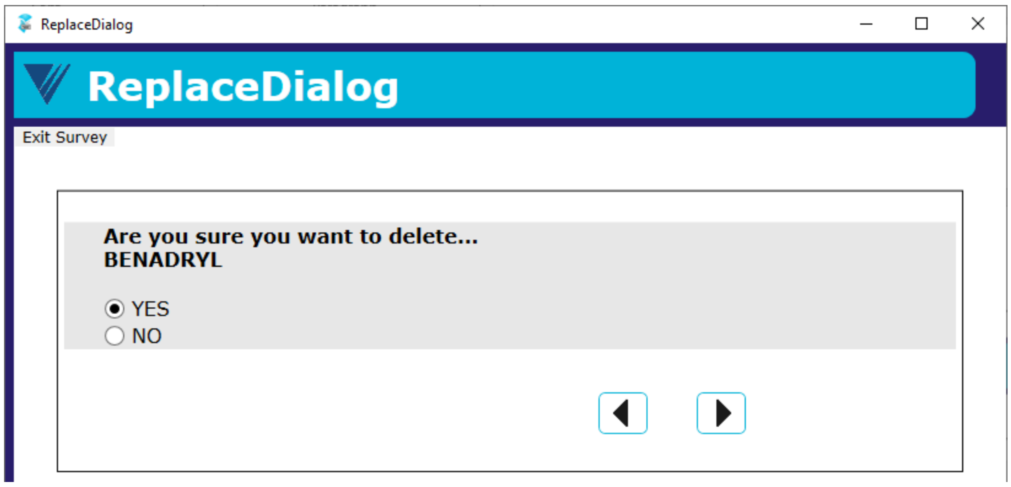
Either CTRL-D or "Delete" on the menu bar leads us to the parallel DeleteBlock.

> back to Table of Contents



Which medicine would you like to delete?
☒ BENADRYL
☐ TYLENOL

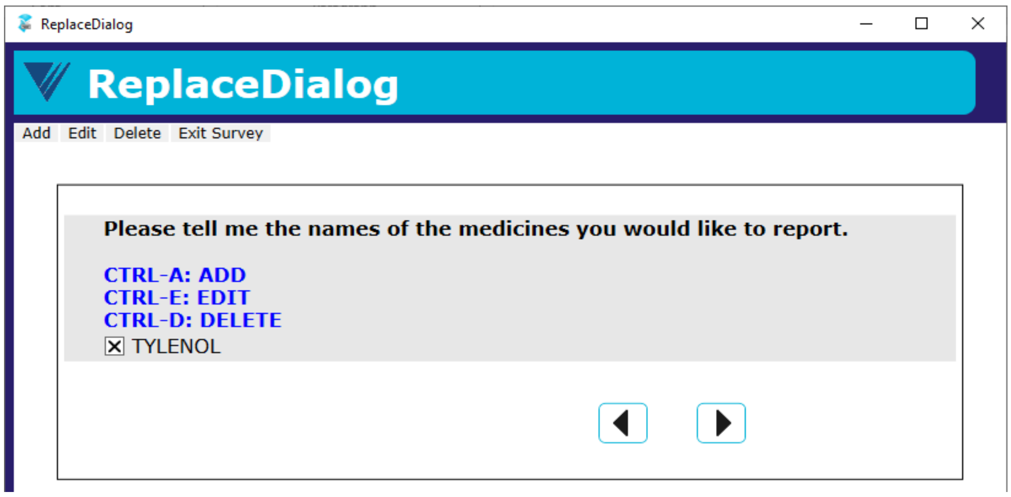
Select BENADRYL to delete.



Are you sure you want to delete...
BENADRYL

[✓ YES]
NO

The end of the parallel DeleteBlock leads us back to Main parallel.

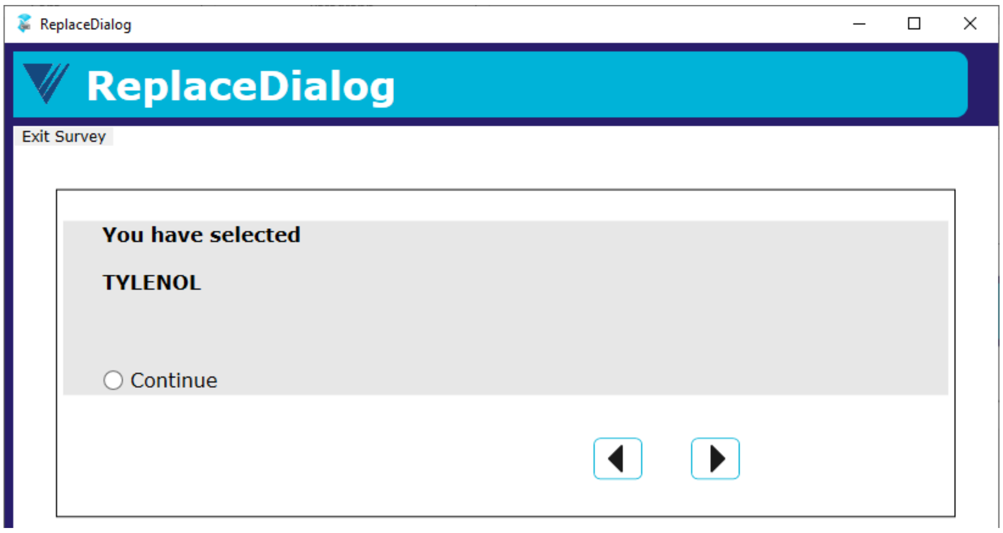


Please tell me the names of the medicines you would like to report.

CTRL-A: ADD
CTRL-E: EDIT
CTRL-D: DELETE
☒ TYLENOL

> back to Table of Contents

Click the forward arrow to the next page in the Main parallel.



You have selected

TYLENOL

○ Continue

We see the medicines that we added but not the ones we deleted.

2.2 Off-Route to Go Back to an Earlier Question

In this example, we collect family household member information on the phone one by one. When the user stops adding more people, we ask them to confirm. When the confirmation is denied, rather than asking the user to keep pressing the "back" button to move backward to the beginning to correct the answers, we lead the user back to the page where the first household member was collected.

To the respondents, this happens when they simply answer the questions. They do not need to use extra buttons or keys to go off-route.

2.2.1 Resource Database

In the resource database, we created master page template PageGoToFieldMobile that goes to a certain field on load. We are hardcoding the field being redirected here for simplicity. However, with the addition of a page template parameter or a conditional action, this page can easily serve the purpose of redirecting to more than one field.

```
<MasterPageTemplate DesignWidth="320" DesignHeight="480"
OnLoad="{Action
GotoField('FSQ.PersonTable.HHRoster[1].FSQ010IntroMobile')}">
```

2.2.2 Data Model

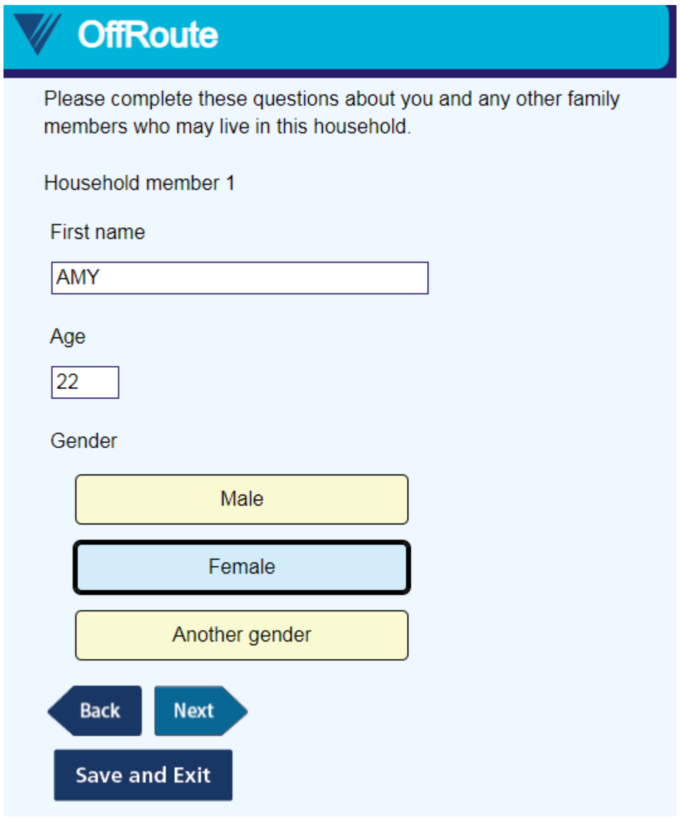
In the data model, after confirmation is denied, we route to a dummy page that uses the above mentioned PageGoToFieldMobile page template to go back to the first household member collected.

```
FSQ050 {FSQ050}
IF FSQ050 = No THEN
   FSQ050No
ENDIF
```

> back to Table of Contents

```
LAYOUTSET "Phone"
   AT FSQ050No MASTERPAGE TEMPLATE "PageGoToFieldMobile"
```

2.2.3 Result


Please complete these questions about you and any other family members who may live in this household.

Household member 1

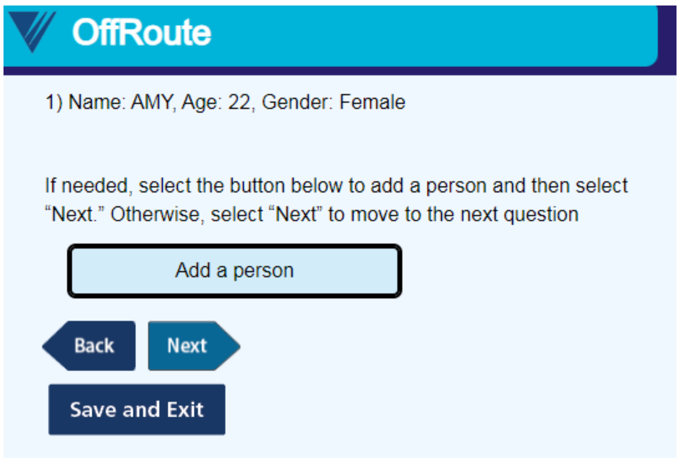
First name

AMY

Age

22

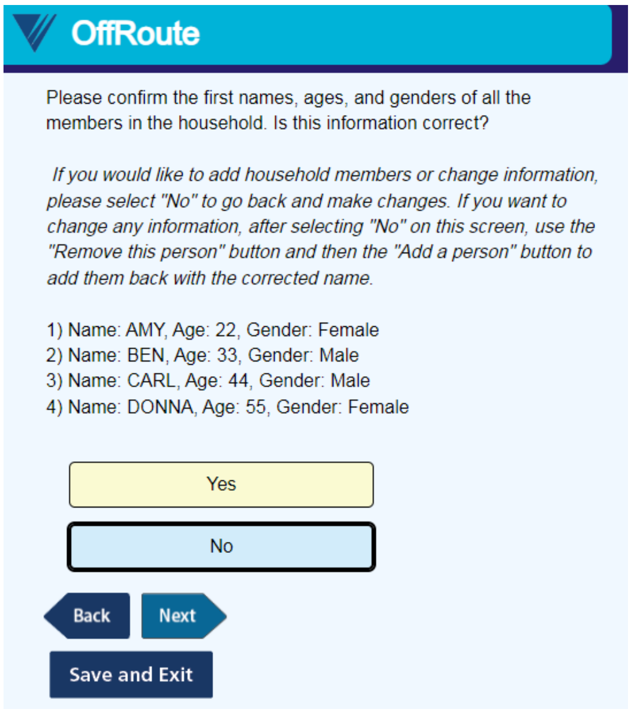
Gender


1) Name: AMY, Age: 22, Gender: Female

If needed, select the button below to add a person and then select “Next.” Otherwise, select “Next” to move to the next question

And a few more household members later …

> back to Table of Contents


OffRoute

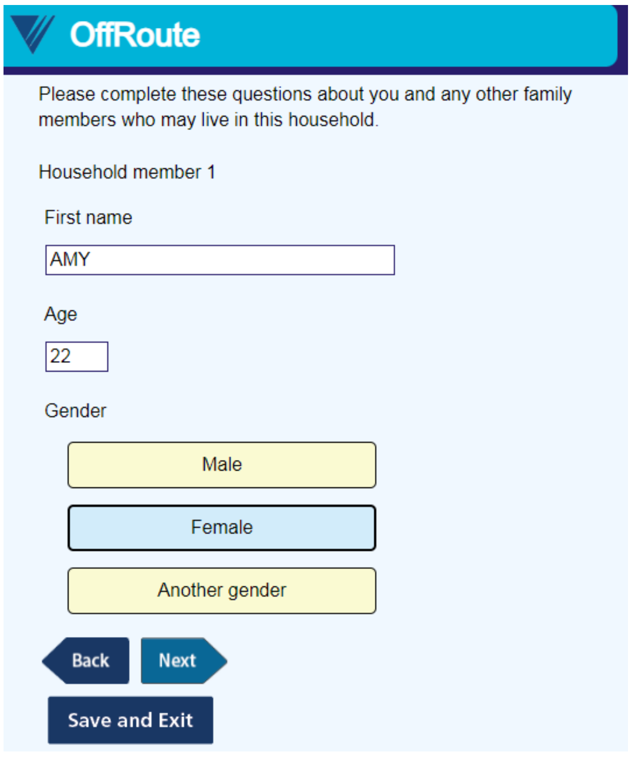
Please confirm the first names, ages, and genders of all the members in the household. Is this information correct?

If you would like to add household members or change information, please select "No" to go back and make changes. If you want to change any information, after selecting "No" on this screen, use the "Remove this person" button and then the "Add a person" button to add them back with the corrected name.

1) Name: AMY, Age: 22, Gender: Female
2) Name: BEN, Age: 33, Gender: Male
3) Name: CARL, Age: 44, Gender: Male
4) Name: DONNA, Age: 55, Gender: Female

Yes [✓ No]

After the answer "No" is selected and the "Next" button is clicked, we are led back to the first household member collected.


OffRoute

Please complete these questions about you and any other family members who may live in this household.

Household member 1

First name: AMY

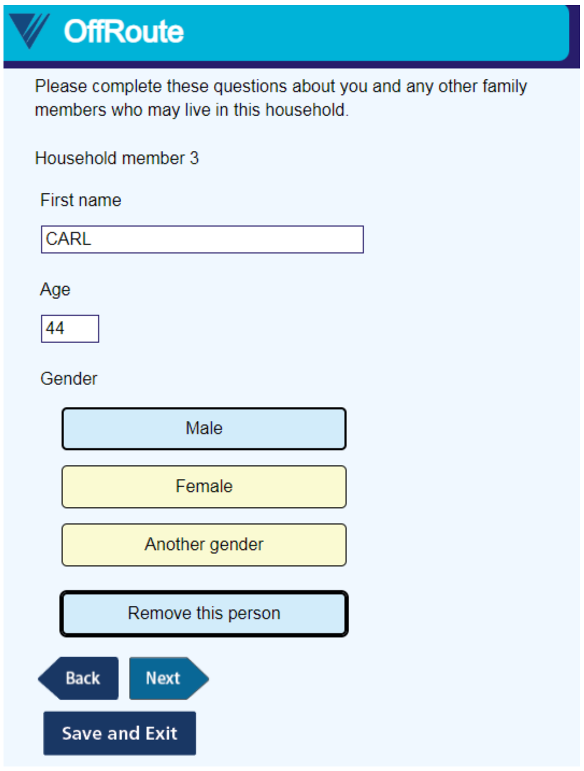
Age: 22

Gender: Male [✓ Female] Another gender

Now, let us remove Carl.

> back to Table of Contents

We click the "Next" button a few times to move forward to Carl's page.


OffRoute

Please complete these questions about you and any other family members who may live in this household.

Household member 3

First name

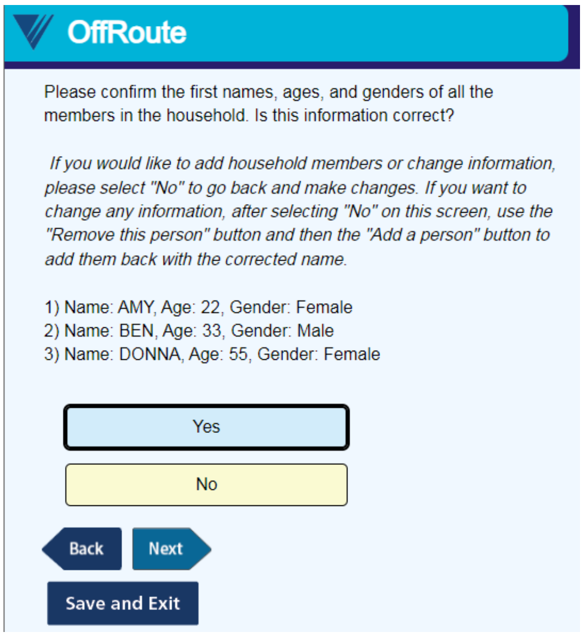
CARL

Age

44

Gender

We click the "Remove this person" button, and then we click the "Next" button a few times to move back to the confirmation page.


OffRoute

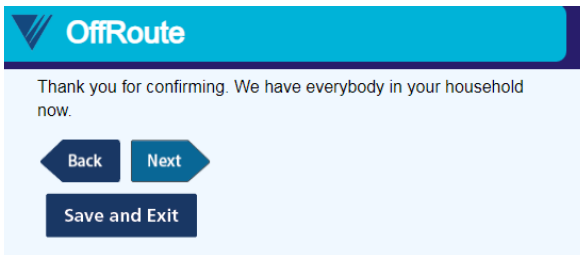
Please confirm the first names, ages, and genders of all the members in the household. Is this information correct?

If you would like to add household members or change information, please select "No" to go back and make changes. If you want to change any information, after selecting "No" on this screen, use the "Remove this person" button and then the "Add a person" button to add them back with the corrected name.

1) Name: AMY, Age: 22, Gender: Female
2) Name: BEN, Age: 33, Gender: Male
3) Name: DONNA, Age: 55, Gender: Female

Now we click the "Yes" button to confirm and move forward.

> back to Table of Contents



Thank you for confirming. We have everybody in your household now.

It routes to the next page.

To make this more sophisticated and less for the respondent to navigate, we can also make the collected household member roster available as an enumerated type question for the respondent to choose which household member they would like to go back to change.

3. Conclusions

With the resource database, it is relatively easy to run third-party programs, navigate off-route, or even start multiple surveys given different conditions consecutively. We just need to be mindful of covering all grounds, especially when we try to program routing outside of the rules.

However, while we can program similar functionality and collect the same data without Manipula Dialog, I cannot say we can really recreate the experience of a Manipula Dialog box yet without Manipula. It is because there are controls that we do not have outside of Manipula Dialog like sizing of the window. Moreover, the laying over of windows cannot happen within one survey project.

Unfortunately, Manipula still does not work in browsers. So the web experience would still be somewhat limited when compared to the Windows DEP experience.

> back to Table of Contents

Replacing Manipula in Blaise 5

Siu Chong Wan and Ryan Webb, Westat

Presenter: Siu Chong Wan

We have done a lot of tricky things in Blaise 4 by calling Manipula procedures from within the instrument that the data model alone cannot handle, for example, showing dialog, calling a third-party program and importing its output into the in-memory data of the ongoing interview, and starting different instruments in order with conditions. In Blaise 5, a lot of these can be accomplished with the use of the resource database.

Although Blaise 5 Manipula is still a very powerful tool, Manipula process scripts cannot be used in browsers. This paper discusses how, with the resource database, hopefully we can have similar flexibility in both windows and browsers on some occasions.

1. Things We Tried So Far

1.1 Replacing Manipula Dialog

In Blaise 4, we used the Maniplus Dialog to display text information, display images and let the user browse through it, obtain information from the user, and enter data directly into a Blaise data set. To do this, we tied the Manipula procedure that defined the dialog box to the type in the data model properties.

In Blaise 5, with the resource database, is the Maniplus Dialog a necessity for this purpose? A Page Template can display extra elements with label control, image control, and even video control. This is useful if we do not want to maintain multiple projects and packages to run Manipula procedures. However, that is not to say that displaying a page template from the resource database can totally replicate the experience of displaying a Maniplus Dialog.

A Maniplus Dialog can overlap the data entry window as a second window; the use of a master page template or a custom page template leads to a new page in the same window. While both can display the same elements and collect the same data, visually they are two different experiences. With a Maniplus Dialog, the user can tell they are being diverted to a new window while they can still see the original data entry window in the background; with a new page (even supported by a different page template for a different look and feel) in the same window, everything just seems to be part of the same flow to the user. Where one’s preference lies depends on the situation and the purpose of the page.

In addition, Maniplus Dialog still has the advantage of being able to define precise dialog height and width that we cannot do in a page template.

1.2 Running Third-Party Programs

In Blaise 4, we used the Maniplus Run function to execute external applications. To do so during runtime, we tied the Manipula procedure in the data model properties to the type that needs to run the third-party application.

The introduction of the StartLocalProcess action in Blaise 5.13 opened the door to a whole new world. The StartLocalProcess action is very handy for the purpose of running third-party applications. It eliminated the needs for a Manipula procedure to run external applications. In the middle of a survey, we can introduce something that cannot be programmed in Blaise, or something that simply can be programmed more efficiently outside of Blaise.

> back to Table of Contents

With the Blaise 5 SessionData API, we should be able to create external programs that pass data between the Blaise data in the current session and the external programs. What happens in Vegas does not have to stay in Vegas anymore. We might be able to bring home the money.

However, it is also worth noting that, like Manipula, the StartLocalProcess action does not work in browsers.

1.3 Flexible Routing

We always count on Blaise to follow the rules. In a Blaise survey, a field cannot be on route more than once in the rules. Rules are checked constantly to ensure data integrity. That is a strength. However, this can also become inflexible on occasion. When needed, we would like to not follow the rules.

In Blaise 4, the easiest we could do might be the INVOLVING instruction in an edit check. But that has to be presented as an error, not a choice. Of course, the next thing we could always count on was, again, Manipula because we could use the Maniplus instruction SETACTIVEFIELD.

In Blaise 5, with various resource database actions, it is relatively easy to change the flow of routing when necessary. For instance, rather than asking the respondent to move back many pages to change something, we can let them press a button to jump back to a certain point. With the GotoField action in Blaise 5, we can provide respondents with the opportunity to jump between questions without triggering any errors or maintaining Manipula scripts.

However, this happens while the rules are still running constantly. So we need to keep in mind that what we are doing outside of the rules still needs to be in compliance with the rules.

1.4 Running Multiple Surveys in Order

In Blaise 4, when we need to run multiple instruments together in a certain order, we would use the Maniplus Edit function/method to start multiple data entry sessions in one sitting. This usually happens outside of the instrument in Manipula.

In Blaise 5, we can use the StartSurvey action for this purpose without the help of Manipula. The StartSurvey action can be used in more than just the OnEnd event. Using the StartSurvey action in the OnEnd event, we can easily run multiple surveys from start to end in one order. Using the StartSurvey action in the OnLoad event of a page, or an event triggered by a layout control such as an OnClick event of a button, we do not have to wait until a survey is completed before the next survey starts. The current session will end and the survey data will be saved before the StartSurvey action is performed.

We can still see the value of controlling the flow of multiple surveys in Manipula with the Edit function/method if it is preferable to keep the programming of that flow in one place, that is, one Manipula project. However, using the StartSurvey action outside of Manipula gives us the flexibility of when, where, and how different surveys start consecutively. More significantly, the StartSurvey action also works in web surveys, while Manipula still cannot.

2. Examples

2.1 Replacing Manipula Dialog Boxes with Parallel Blocks

In Blaise 4, we used Maniplus Dialog to collect roster member information in a dialog box, where we could offer options of actions like add, edit, and delete. In this example, we use a combination of parallel blocks and resource database actions to collect a roster of medicines. This involves off-route flows that are transparent to the respondents.

> back to Table of Contents

2.1.1 Data Model

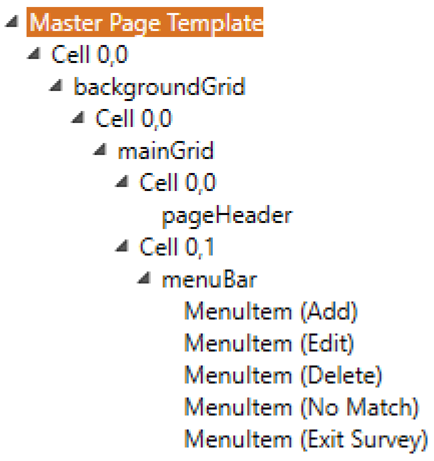
In the data model, we define a parallel block for each individual “dialog” path.

```
SETTINGS
   PARALLEL
      AddBlock
      EditBlock
      DeleteBlock
```

Each parallel block adds/edits/deletes medicines and saves the medicines back to the Main parallel block. As a result, the data collected in the parallel blocks are disposable.

2.1.2 Resource Database

In the resource database, we can use menu bar items, buttons, or keyboard shortcuts to provide access to the parallel blocks. In our example, we use menu bar items with a shortcut property to lead to the parallel blocks and other off-route flows.



```
◢ Master Page Template
  ◢ Cell 0,0
    ◢ backgroundGrid
      ◢ Cell 0,0
        ◢ mainGrid
          ◢ Cell 0,0
              pageHeader
          ◢ Cell 0,1
            ◢ menuBar
                MenuItem (Add)
                MenuItem (Edit)
                MenuItem (Delete)
                MenuItem (No Match)
                MenuItem (Exit Survey)
```

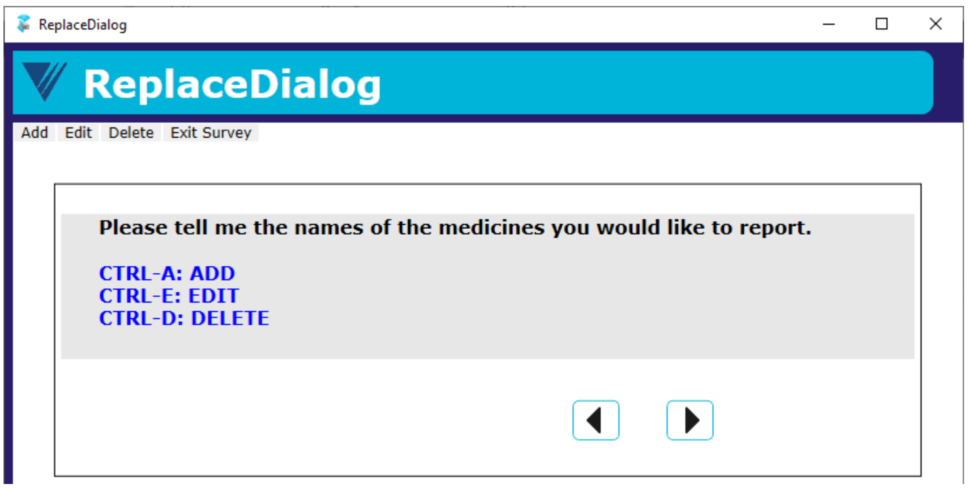
```
<MenuItem Text="Add" Shortcut="Ctrl+A" Visibility="{Expression IF
ENDSWITH(State.ActiveFieldName, 'Roster') THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action SetParallel('AddBlock')}" />

<MenuItem Text="Edit" Shortcut="Ctrl+E" Visibility="{Expression IF
ENDSWITH(State.ActiveFieldName, 'Roster') THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action SetParallel('EditBlock')}" />

<MenuItem Text="Delete" Shortcut="Ctrl+D" Visibility="{Expression IF
ENDSWITH(State.ActiveFieldName, 'Roster') THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action SetParallel('DeleteBlock')}" />

<MenuItem Text="No Match" Shortcut="Ctrl+Z" Visibility="{Expression IF
Page.ActiveField.TypeName = 'TLookupString' THEN 'Visible' ELSE
'Collapsed' ENDIF}" OnClick="{Action AssignField('NoMatchButton', '1',
{Expression 1});Save();GotoField('AddBlock.PMedNoMatchString')}" />
```

> back to Table of Contents

Sending the user to a parallel block is the easy part. Clearing out the data from the parallel blocks while keeping the data in the Main parallel could take some effort. Of course, we can always create arrays of the parallel blocks so that each call of the parallel block is a new instance, rather than clearing out anything. But that also means we would be keeping duplicated data that are not necessary, unless it is desirable to keep the history of all the add/edit/delete instances.

2.1.3 Result

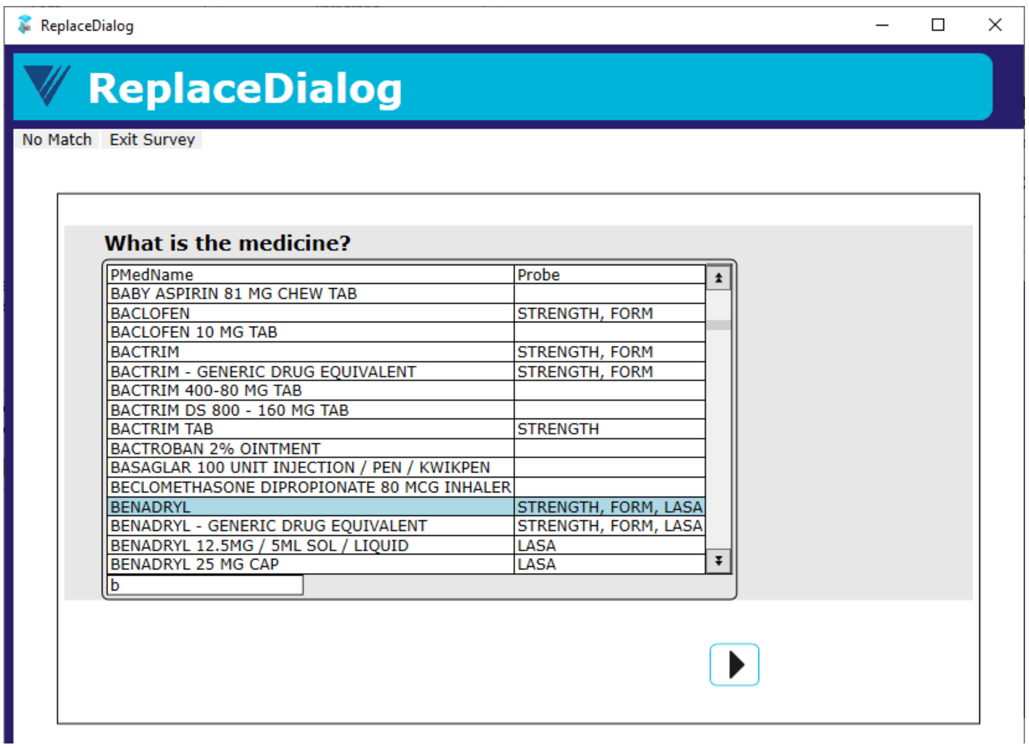
We first start in the Main parallel.



Please tell me the names of the medicines you would like to report.

CTRL-A: ADD
CTRL-E: EDIT
CTRL-D: DELETE

Either CTRL-A or the “Add” on the menu bar leads us to the parallel AddBlock.



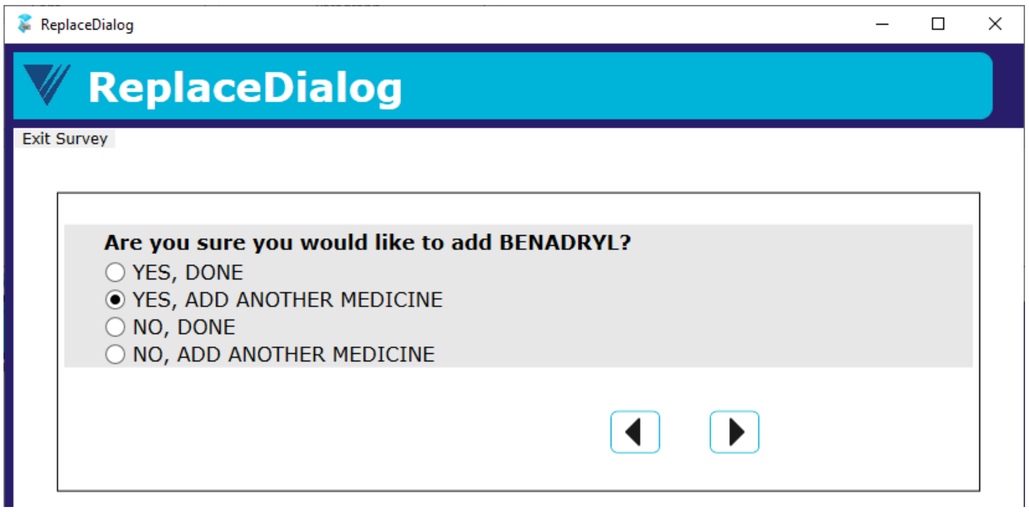
What is the medicine?

PMedName	Probe
BABY ASPIRIN 81 MG CHEW TAB	
BACLOFEN	STRENGTH, FORM
BACLOFEN 10 MG TAB	
BACTRIM	STRENGTH, FORM
BACTRIM - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
BACTRIM 400-80 MG TAB	
BACTRIM DS 800 - 160 MG TAB	
BACTRIM TAB	STRENGTH
BACTROBAN 2% OINTMENT	
BASAGLAR 100 UNIT INJECTION / PEN / KWIKPEN	
BECLOMETHASONE DIPROPIONATE 80 MCG INHALER	
BENADRYL	STRENGTH, FORM, LASA
BENADRYL - GENERIC DRUG EQUIVALENT	STRENGTH, FORM, LASA
BENADRYL 12.5MG / 5ML SOL / LIQUID	LASA
BENADRYL 25 MG CAP	LASA

b

> back to Table of Contents

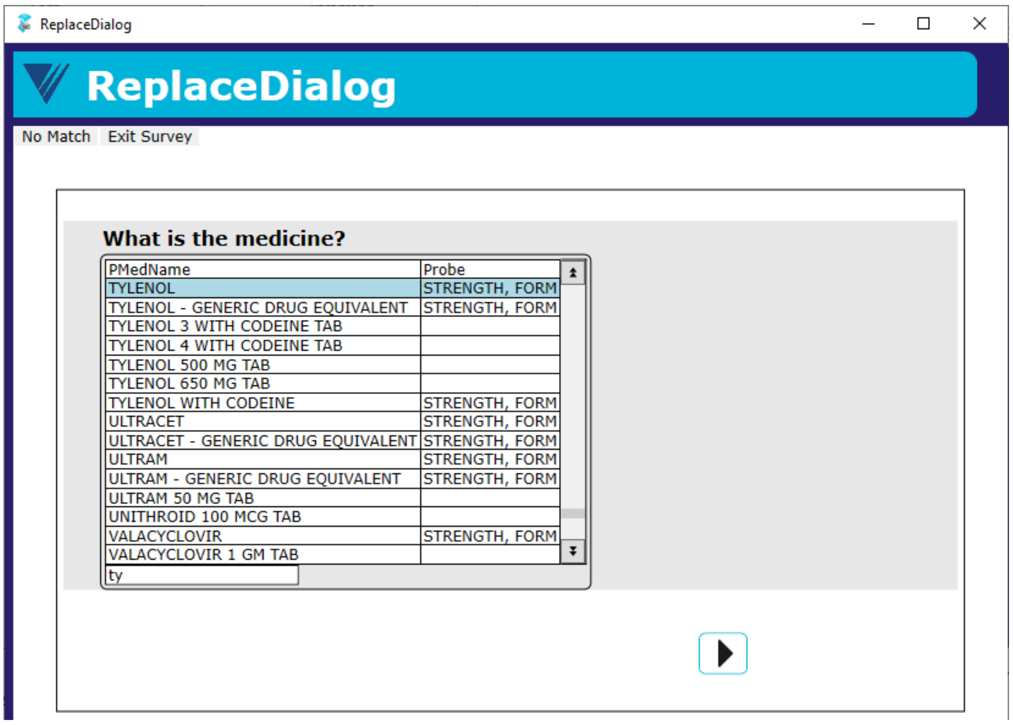
Select BENADRYL and click the forward arrow.



Are you sure you would like to add BENADRYL?

- ○ YES, DONE
- ● YES, ADD ANOTHER MEDICINE
- ○ NO, DONE
- ○ NO, ADD ANOTHER MEDICINE

Confirm adding and plan to add another medicine.



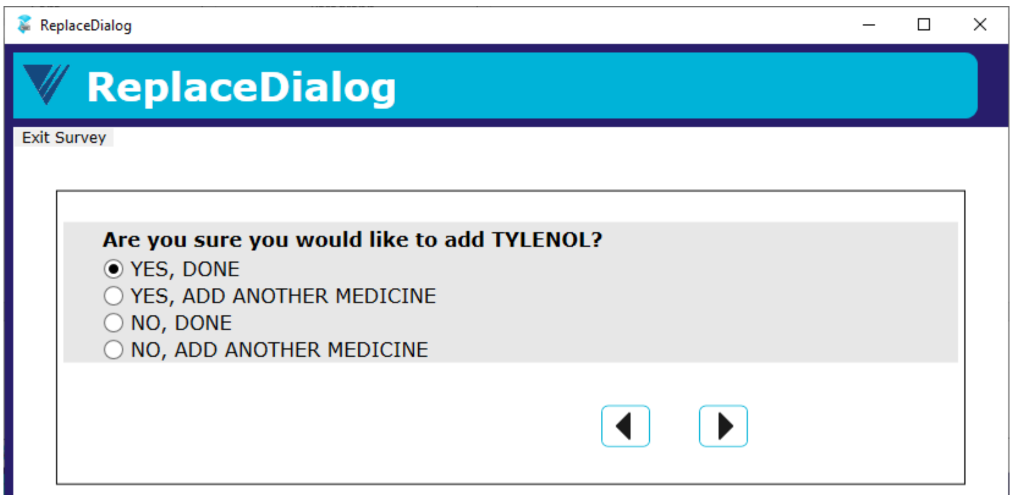
What is the medicine?

PMedName	Probe
TYLENOL	STRENGTH, FORM
TYLENOL - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
TYLENOL 3 WITH CODEINE TAB	
TYLENOL 4 WITH CODEINE TAB	
TYLENOL 500 MG TAB	
TYLENOL 650 MG TAB	
TYLENOL WITH CODEINE	STRENGTH, FORM
ULTRACET	STRENGTH, FORM
ULTRACET - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
ULTRAM	STRENGTH, FORM
ULTRAM - GENERIC DRUG EQUIVALENT	STRENGTH, FORM
ULTRAM 50 MG TAB	
UNITHROID 100 MCG TAB	
VALACYCLOVIR	STRENGTH, FORM
VALACYCLOVIR 1 GM TAB	

ty

Select TYLENOL this time and move forward.

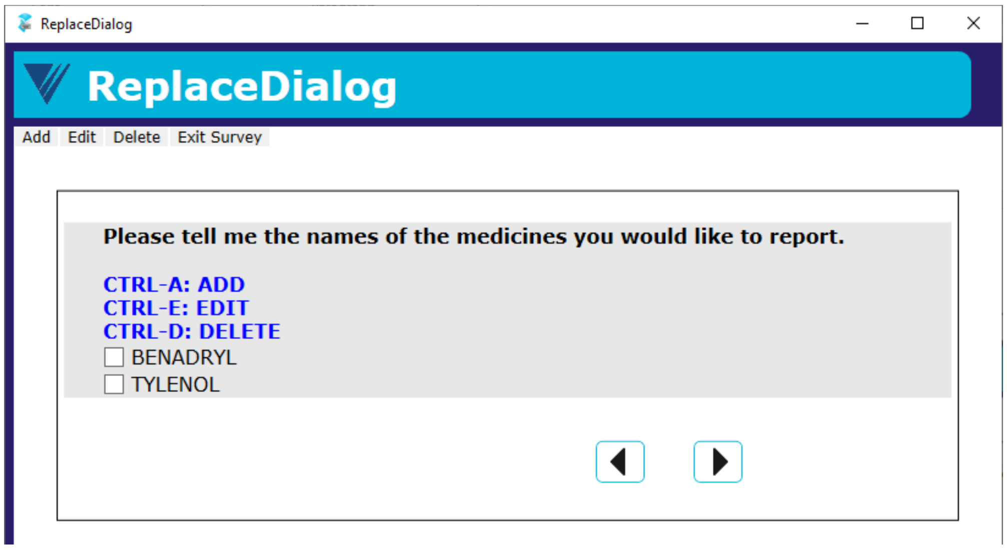
> back to Table of Contents



Are you sure you would like to add TYLENOL?

[✓ YES, DONE]
YES, ADD ANOTHER MEDICINE
NO, DONE
NO, ADD ANOTHER MEDICINE

The end of the parallel AddBlock leads us back to the Main parallel.



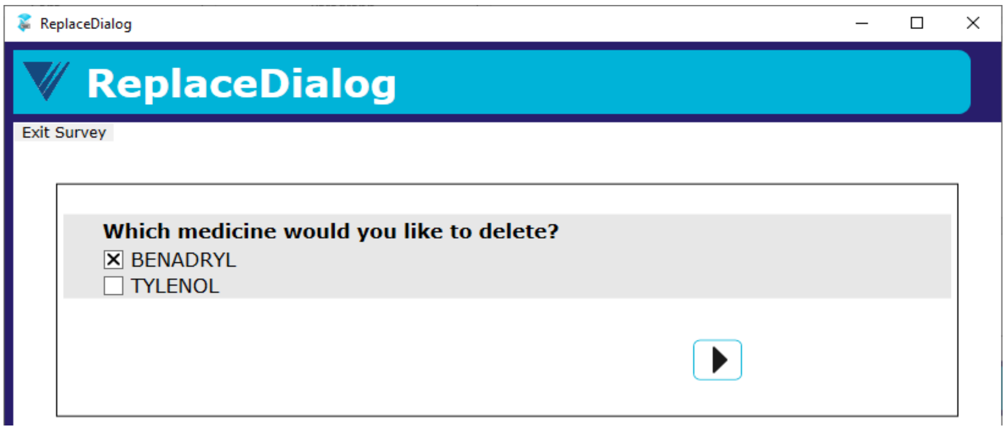
Please tell me the names of the medicines you would like to report.

CTRL-A: ADD
CTRL-E: EDIT
CTRL-D: DELETE

☐ BENADRYL
☐ TYLENOL

Either CTRL-D or “Delete” on the menu bar leads us to the parallel DeleteBlock.

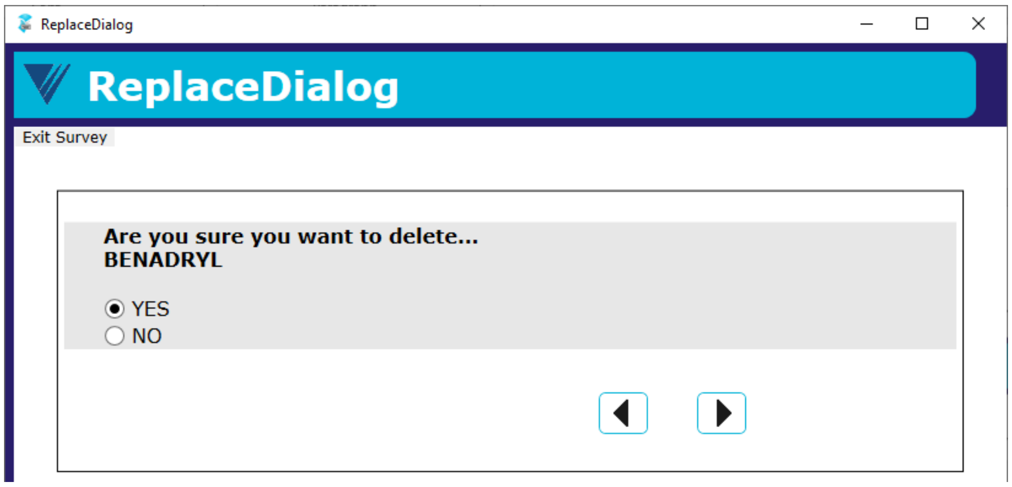
> back to Table of Contents



Which medicine would you like to delete?

☒ BENADRYL
☐ TYLENOL

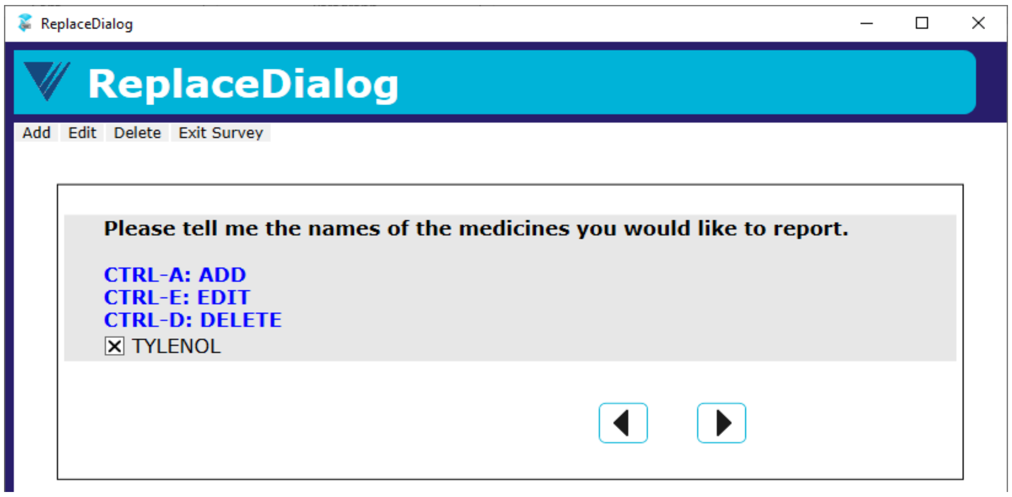
Select BENADRYL to delete.



Are you sure you want to delete...
BENADRYL

● YES
○ NO

The end of the parallel DeleteBlock leads us back to Main parallel.

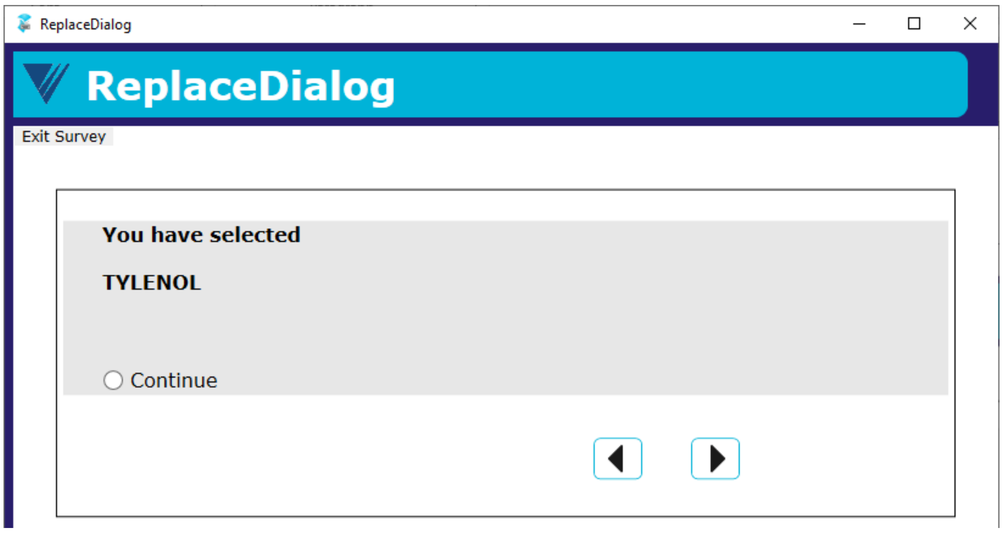


Please tell me the names of the medicines you would like to report.

CTRL-A: ADD
CTRL-E: EDIT
CTRL-D: DELETE
☒ TYLENOL

> back to Table of Contents

Click the forward arrow to the next page in the Main parallel.



You have selected

TYLENOL

○ Continue

We see the medicines that we added but not the ones we deleted.

2.2 Off-Route to Go Back to an Earlier Question

In this example, we collect family household member information on the phone one by one. When the user stops adding more people, we ask them to confirm. When the confirmation is denied, rather than asking the user to keep pressing the "back" button to move backward to the beginning to correct the answers, we lead the user back to the page where the first household member was collected.

To the respondents, this happens when they simply answer the questions. They do not need to use extra buttons or keys to go off-route.

2.2.1 Resource Database

In the resource database, we created master page template PageGoToFieldMobile that goes to a certain field on load. We are hardcoding the field being redirected here for simplicity. However, with the addition of a page template parameter or a conditional action, this page can easily serve the purpose of redirecting to more than one field.

```
<MasterPageTemplate DesignWidth="320" DesignHeight="480"
OnLoad="{Action
GotoField('FSQ.PersonTable.HHRoster[1].FSQ010IntroMobile')}">
```

2.2.2 Data Model

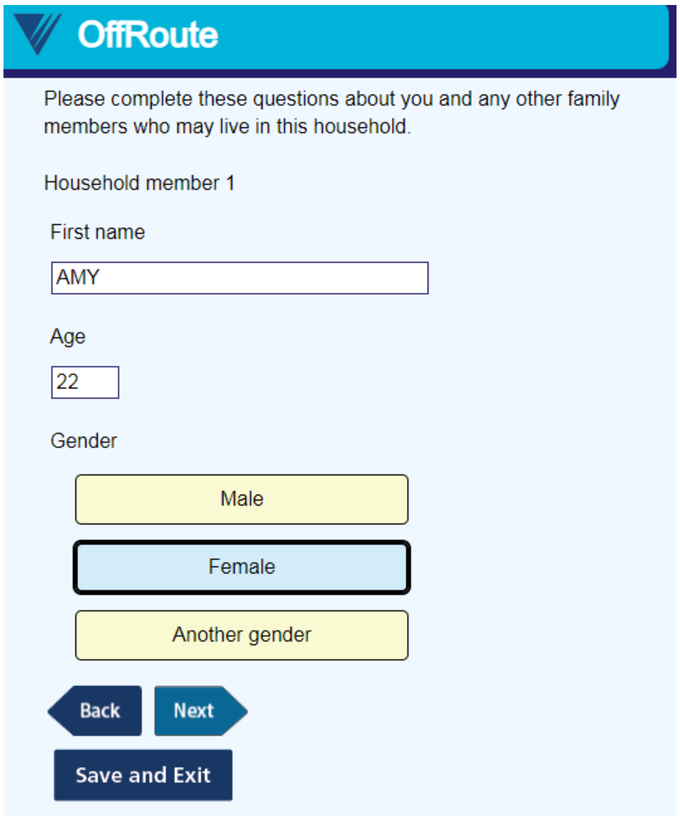
In the data model, after confirmation is denied, we route to a dummy page that uses the above mentioned PageGoToFieldMobile page template to go back to the first household member collected.

```
FSQ050 {FSQ050}
IF FSQ050 = No THEN
   FSQ050No
ENDIF
```

> back to Table of Contents

```
LAYOUTSET "Phone"
   AT FSQ050No MASTERPAGE TEMPLATE "PageGoToFieldMobile"
```

2.2.3 Result


OffRoute

Please complete these questions about you and any other family members who may live in this household.

Household member 1

First name

AMY

Age

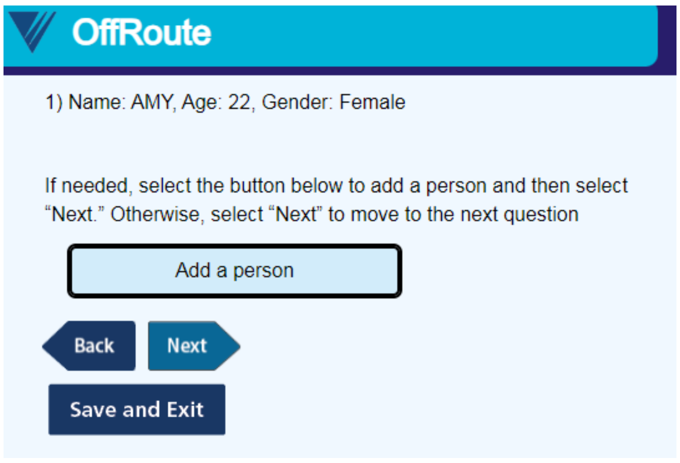
22

Gender

Male

[✓ Female]

Another gender

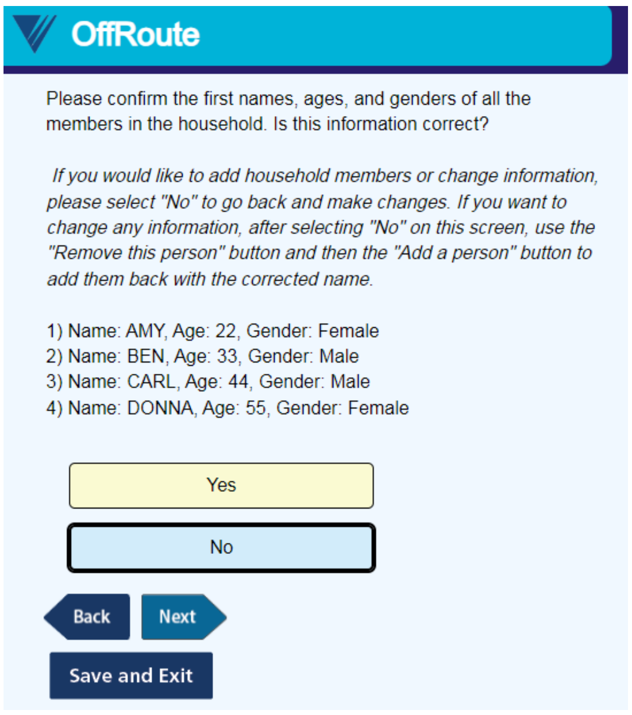

OffRoute

1) Name: AMY, Age: 22, Gender: Female

If needed, select the button below to add a person and then select "Next." Otherwise, select "Next" to move to the next question

And a few more household members later …

> back to Table of Contents



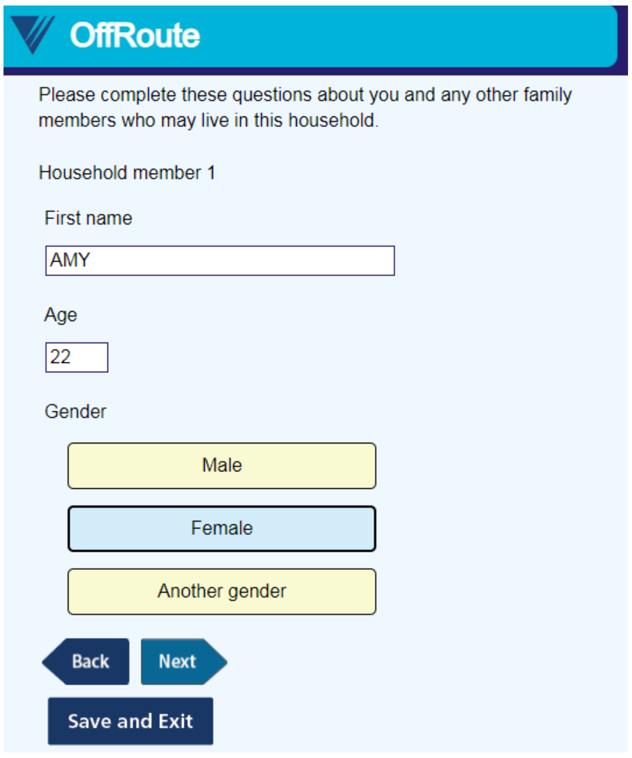
Please confirm the first names, ages, and genders of all the members in the household. Is this information correct?

If you would like to add household members or change information, please select "No" to go back and make changes. If you want to change any information, after selecting "No" on this screen, use the "Remove this person" button and then the "Add a person" button to add them back with the corrected name.

1) Name: AMY, Age: 22, Gender: Female
2) Name: BEN, Age: 33, Gender: Male
3) Name: CARL, Age: 44, Gender: Male
4) Name: DONNA, Age: 55, Gender: Female

Yes [✓ No]

After the answer "No" is selected and the "Next" button is clicked, we are led back to the first household member collected.



Please complete these questions about you and any other family members who may live in this household.

Household member 1

First name: AMY

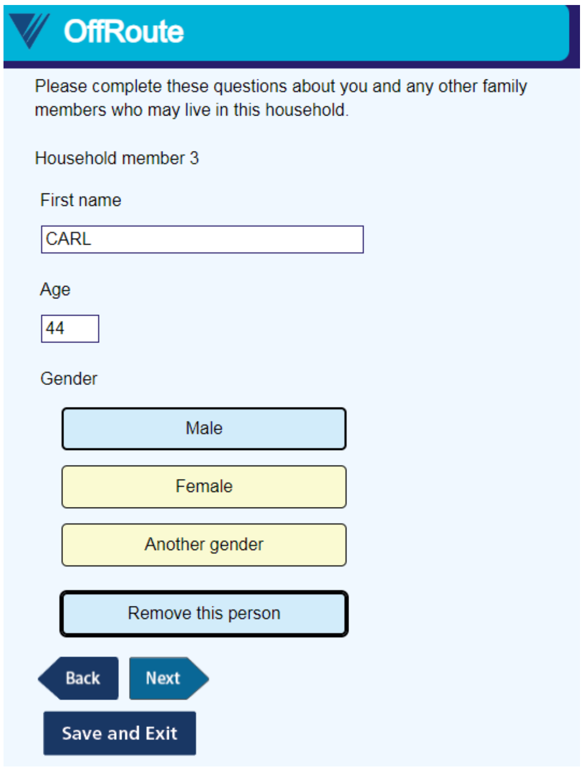
Age: 22

Gender: Male [✓ Female] Another gender

Now, let us remove Carl.

> back to Table of Contents

We click the “Next” button a few times to move forward to Carl’s page.


Please complete these questions about you and any other family members who may live in this household.

Household member 3

First name

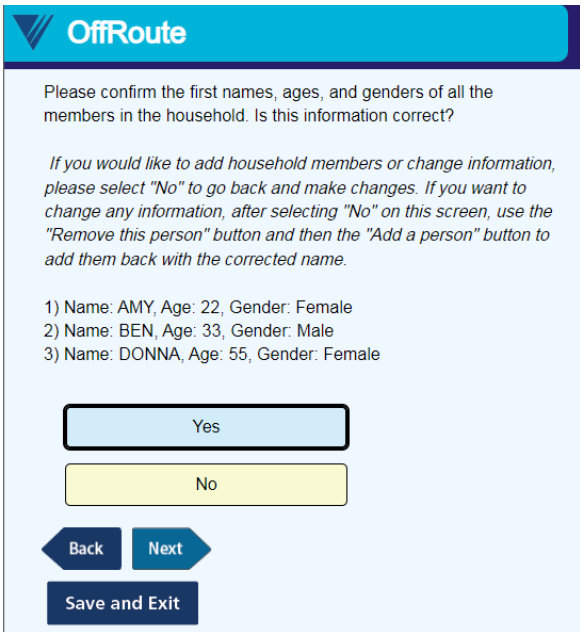
CARL

Age

44

Gender

We click the “Remove this person” button, and then we click the “Next” button a few times to move back to the confirmation page.

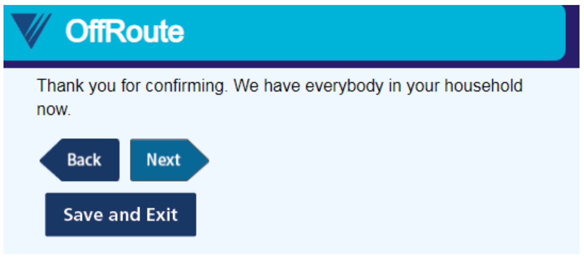

Please confirm the first names, ages, and genders of all the members in the household. Is this information correct?

If you would like to add household members or change information, please select "No" to go back and make changes. If you want to change any information, after selecting "No" on this screen, use the "Remove this person" button and then the "Add a person" button to add them back with the corrected name.

1) Name: AMY, Age: 22, Gender: Female
2) Name: BEN, Age: 33, Gender: Male
3) Name: DONNA, Age: 55, Gender: Female

Now we click the “Yes” button to confirm and move forward.

> back to Table of Contents



Thank you for confirming. We have everybody in your household now.

It routes to the next page.

To make this more sophisticated and less for the respondent to navigate, we can also make the collected household member roster available as an enumerated type question for the respondent to choose which household member they would like to go back to change.

3. Conclusions

With the resource database, it is relatively easy to run third-party programs, navigate off-route, or even start multiple surveys given different conditions consecutively. We just need to be mindful of covering all grounds, especially when we try to program routing outside of the rules.

However, while we can program similar functionality and collect the same data without Manipula Dialog, I cannot say we can really recreate the experience of a Manipula Dialog box yet without Manipula. It is because there are controls that we do not have outside of Manipula Dialog like sizing of the window. Moreover, the laying over of windows cannot happen within one survey project.

Unfortunately, Manipula still does not work in browsers. So the web experience would still be somewhat limited when compared to the Windows DEP experience.

> back to Table of Contents

How Creating a Pipeline for Automatically Analyzing and Sharing Paradata Facilitated the Ability to Make Data-Driven Adjustments to Improve Data Collections

Elise Alstad, Erdal Kilicdogan, Sara Grimstad, Katharina Rossbach, Gezim Seferi, Marta Krawczynska, Statistics Norway

1. Abstract

Paradata has proven key to managing dynamic data collections, as we can use the paradata to identify and measure nonresponse and measurement errors. However, paradata contains large amounts of unstructured data, and it can be resource-intensive to extract and structure the data. We therefore found that we needed a robust and efficient process for extracting, cleaning, storing, and analyzing paradata. Using Python with Google Cloud Storage services, we created a data pipeline that synchronizes paradata from Blaise daily, parses it, and stores it in the cloud. By having up-to-date paradata, we can gain insightful information about the data collection process as it progresses. For instance, using Audit Trail data with sample data, we assess nonresponse bias for different demographic groups. Likewise, by using Dial History data, we analyze the outcome of each call and initiate measures based on their effectiveness. To make information about the data collection available to all stakeholders, irrespective of coding experience, results from the paradata analysis are shared to an internal web page that is updated daily. By creating an automatic pipeline that allows colleagues to evaluate the data collection process, we have made it easier to make data-driven decisions that adjust for bias and measurement errors. Because we use Python, which is an open-source programming language, aspects of our pipeline can be implemented by others without any cost. Similarly, we hope that sharing the journey of how we created the pipeline and the benefits we saw from it can be useful for other Blaise users.

2. Introduction

Data collection demands significant resources in terms of both cost and time, and it is therefore important to understand how to improve data collection processes. Paradata can be used to understand respondent behavior and enhance survey questionnaires to gain higher survey quality, as well as to monitor and identify areas for improvement to effectively manage data collections dynamically (Kreuter, Couper, & Lyberg, 2010). At Statistics Norway, we have used paradata from Dial History and Audit Trail, and we will refer to these data sources when we use the term paradata. These data sources include all call records, actions that the interviewer and respondents perform in the survey, and the timestamps for these actions. Post-survey analyses of paradata will help us understand how to improve the data collection process for next time. But by using real-time, or nearly real-time, paradata, we can help stakeholders to make changes during data collection in ongoing surveys (Schouten, Peytchev, & Wagner, 2017). While there are numerous uses of paradata (see Hunt, 2016; Cheung et al., 2016; Kreuter, Couper, & Lyberg, 2010) at Statistics Norway, we have focused on developing the use of paradata in two main areas:

- Improving survey questionnaires by understanding respondent behavior in surveys.
- Dynamic management of surveys by monitoring and adjusting for nonresponse bias.

When we started using paradata, the initial step was to export paradata from Blaise, and here we noticed challenges concerning the sheer volume of data and database access. Querying Audit Trail data from large surveys demanded a substantial share of the server capacity, resulting in slower loading time for interviewers. Secondly, we faced challenges related to database access because colleagues outside of the Blaise developer team required access to the paradata. However, as the database with the schemas for Audit Trail also included other schemas, it was unideal that too many individuals would be given access

> back to Table of Contents

to the database. Subsequently, only a limited number of colleagues were given access and could export Audit Trail and Dial History data. Moreover, the individuals who could export paradata had to be cautious about the amount of data they exported while also aiming to export the data during periods when few interviewers were active to avoid slow loading speed for interviewers. Thus, we realized that we needed to develop an alternative solution that would facilitate easier access to paradata.

The team tasked with utilizing paradata had several objectives, one of which was to provide colleagues in the department with an up-to-date overview of the data collection. We started by developing Python programs that would generate data collection reports using call history data from Dial History. However, the program reports were not widely used, mainly due to some colleagues perceiving Python as a barrier due to their unfamiliarity with the programming language. Additionally, considering the sensitivity of paradata, we wished to provide stakeholders with reports without giving them access to the data itself, which was required to run the program reports. To achieve this, we recognized the need to create a way to share reports without requiring users to run the program or access the data. Thus, we set out to create a web page with aggregated results over the data collection.

Furthermore, as some colleagues wished to analyze the data themselves, we saw that starting from scratch with each analysis would be inefficient. We therefore developed small Python program modules that could assist users with analyzing the paradata. To ensure accessibility and to encourage more reuse of code, all team members work within one GitHub repository. By developing programs that should work on all instruments and require little coding expertise to run, we have created a versatile solution that can accommodate the needs of colleagues who are involved in different parts of the data collection process. By establishing this pipeline, project managers would receive better support in dynamically managing data collection, and survey methodologists and Blaise developers could more easily access and analyze paradata, facilitating data-driven decisions to improve surveys.

One of Statistics Norway's digitalization strategies includes implementing Cloud Services and making the shift toward using open-source programming languages. Thus, when creating this pipeline, we decided to use Google Cloud Services and develop the programs in Python. However, aspects of this pipeline can be implemented without the use of Cloud Services. The first part of the article illustrates an overview of the pipeline structure. Next, we share some experiences of how paradata has been used by project managers and survey methodologists to dynamically manage data collection and to improve survey questionnaires. Finally, we summarize the positive effects we have seen from implementing the pipeline and discuss our plans for enhancing the pipeline.

3. Paradata Pipeline

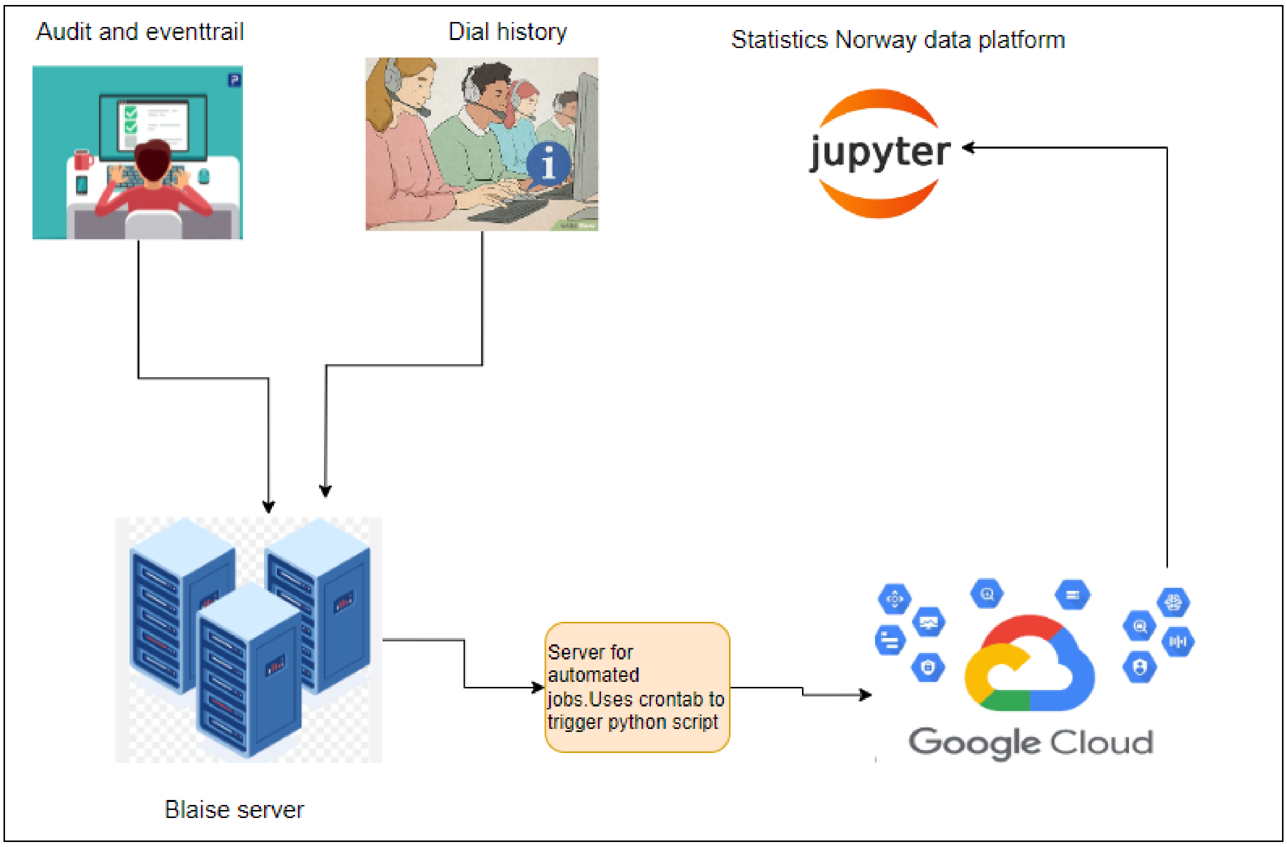
In this section, we will explain how the pipeline is set up. The program code we have used for the pipeline is included in the Appendix. The initial step of the pipeline involves exporting data from our on-premises Blaise server. Next, data is exported to a storage system in the Google Cloud Platform (GCP). Then, the following steps of the pipeline are completed using solutions from Statistics Norway's new cloud-based data platform (DAPLA). In short, the bullet points below and Figure 1 summarize the steps of our pipeline:

- Exporting data from Blaise to GCP
- Transferring data from the source bucket to the production bucket in GCP
- Developing reports and programs in JupyterLab
- Sharing daily results to the internal web page

Dial History data is parsed in Step 1, and Audit Trail data is parsed in Step 2.

> back to Table of Contents

Figure 1. Pipeline from Blaise to GCP


Audit and eventtrail
Dial history
Statistics Norway data platform
jupyter
Server for automated jobs.Uses crontab to trigger python script
Google Cloud
Blaise server

Before explaining the steps in the pipeline, we will first outline the storage structure we use in GCP.

3.1 Using the Google Cloud Platform Structure

In the GCP, a "bucket" is a unit for containing data that serves the same purpose as a folder. All data in Google Cloud must be stored and organized within these buckets. Essentially, a bucket functions as a structured folder, facilitating the organization and management of different data files. Moreover, we have two buckets for different data structures: unmanipulated and manipulated data. By using different buckets for unmanipulated and manipulated data, we can keep backups of the unmanipulated data where only a few individuals have access. Since the unmanipulated data are the basis for all our analysis and manipulation, having this data in a separate bucket ensures that we have a reliable reference point.

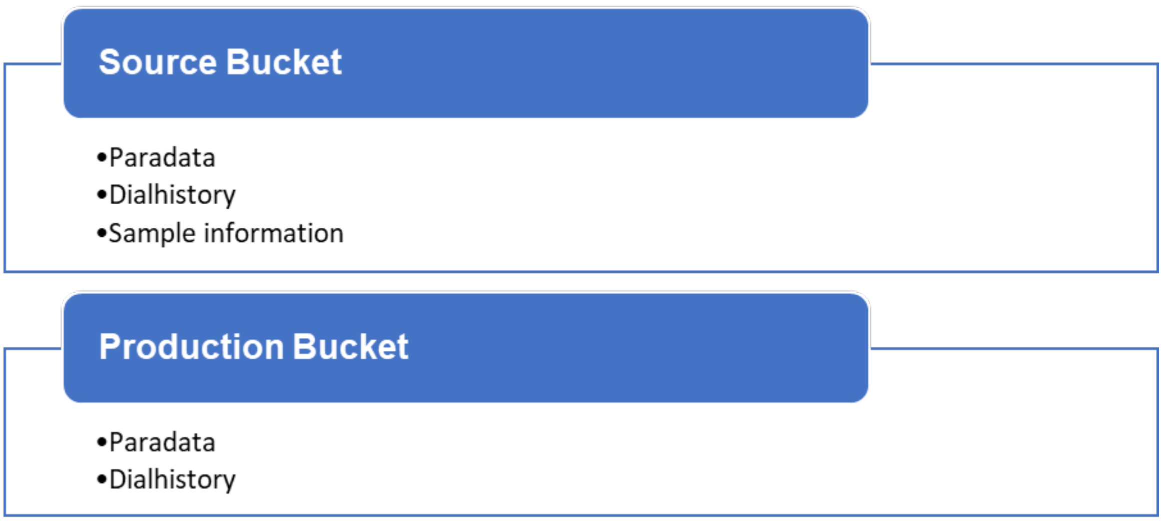
3.1.1 Source Bucket

The source bucket contains unmanipulated paradata from Blaise and select categorical sample data variables, such as age group, region, education level, and gender. Only two people have permission to read, write, or delete files in this bucket.

3.1.2 Production Bucket

In the production bucket, we store manipulated data from the source bucket and data with the select sample variables. Only colleagues who need to use these data are given access to read, write, or delete data in this bucket.

> back to Table of Contents



3.2 Exporting Data from Blaise to Google Cloud Platform

The paradata files are exported from the Blaise databases and then stored as CSV files on an on-premises server. To export the paradata from Blaise, we use Code 1 (see the Appendix) for Audit Trail data and Code 2 (see the Appendix) for Dial History. We use Crontab to automatically schedule the export of the paradata each night. We then use GCP's tool "transfer job" to automatically transfer the data from our on-premises server to the source bucket in GCP. Once this step of the pipeline is completed, Audit Trail data and Dial History data are stored as CSV files in the source bucket.

3.3 Transferring Data from the Source Bucket to the Production Bucket in GCP

The next step in the pipeline is to transfer data in GCP from the source bucket and store it as Parquet files in the production bucket. We have utilized a solution created by Statistics Norway's data platform team that uses Cloud Run in GCP to automate this step. The solution allows a team to write a script that is triggered when a new file is added to the source bucket. For the Audit Trail data, we have developed a script (see Code 3 in the Appendix) that reads the data from the source bucket, parses the data, and then stores the parsed data in the production bucket. Similarly, for Dial History data, we have developed a script that reads the CSV file from the source bucket and stores it as a Parquet file in the production bucket.

3.3.1 Parsing Audit Trail Data

As the Audit Trail data from Blaise is inadequate for effective analysis, we have parsed the paradata to a structure more suitable for analysis using a script we refer to as the paradata parser (see Code 3 in the Appendix). In the unmanipulated Audit Trail–level data, shown in Table 1, we can see that the "Content" column contains information about the action. The content of the "Content" column is structured as XML-formatted data.

To extract and structure the information in the "Content" column, we use the Python XML package xml.etree.ElementTree. The paradata parser script processes the XML content in the "Content" column of the input dataframe, extracting attributes and tags, and creates a new dataframe with the columns "event", "KeyValue", "TimeStamp", "SessionId", and "InstrumentId", in addition to columns from the attributes as shown in Tables 2 and 3. The parser iterates through each row, creating a new column for each new attribute and adding the attribute's value to the corresponding row. Each tag represents an event type and is included in the "event" column. The paradata parser organizes the data into the structure shown in

> back to Table of Contents

Tables 2 and 3. To enhance readability, the data is split into two separate tables. Also, the column “OS” would contain more information, but for viewing purposes, the text has been shortened.

Table 1. Audit Trail Data from Blaise

TimeStamp	Content	KeyValue	SessionId	InstrumentId
2023-02-28 08:14:40	<StartSessionEvent Width="414" Height="707" Device="Browser" Browser="Safari" Language="en" Platform="HTML" OS="Mozilla/5.0 (iPhone)" />	123aaa	{abc-12345}	{defg-678910-abc}
2023-02-28 08:15:59	<UpdatePageEvent LayoutSetName="SSB_Small_Touch" PageIndex="3" />	123aaa	{abc-12345}	{defg-678910-abc}
2023-02-28 08:15:59	<EnterFieldEvent FieldName="skjema. Tilfreds" AnswerStatus="Empty" />	123aaa	{abc-12345}	{defg-678910-abc}
2023-02-28 08:16:00	<LeaveFieldEvent FieldName="skjema. Tilfreds" Value="6" AnswerStatus="Response" />	123aaa	{abc-12345}	{defg-678910-abc}
2023-02-28 08:16:01	<ActionEvent Action="NextField()" ControlID="la_2kaba_7" />	123aaa	{abc-12345}	{defg-678910-abc}

Tables 2 and 3. Parsed Audit Trail Data

Width	Height	Device	Browser	Language	Platform	OS	event	KeyValue	TimeStamp
414	707	Browser	Safari	en	HTML	Mozilla/5.0 (iPhone)	StartSessionEvent	123aaa	2023-02-28 08:14:40
							UpdatePageEvent	123aaa	2023-02-28 08:15:59
							EnterFieldEvent	123aaa	2023-02-28 08:15:59
							LeaveFieldEvent	123aaa	2023-02-28 08:16:00
							ActionEvent	123aaa	2023-02-28 08:16:01

SessionId	InstrumentId	LayoutSetName	PageIndex	FieldName	AnswerStatus	Value	Action	ControlID
{abc-12345}	{defg-678910-abc}							
{abc-12345}	{defg-678910-abc}	SSB_Small_Touch	3					
{abc-12345}	{defg-678910-abc}			skjema.Tilfreds	Empty			
{abc-12345}	{defg-678910-abc}			skjema.Tilfreds	Response	6		
{abc-12345}	{defg-678910-abc}						NextField()	la_2kaba_7

An advantage of the paradata parser, in contrast to programs that are dependent on regular expressions or fixed string/column position to extract information from the “Content” column, is that the paradata parser automatically creates new columns as it iterates over the data and encounters new attributes. This flexibility ensures that the program can process data with different Audit Trail–level settings. The paradata parser program has worked with data that has “Page”, “Field”, and “Keyboard” as the Audit Trail–level setting. By utilizing the paradata parser, the pipeline can process all our surveys, each potentially having different Audit Trail–level configurations, thus ensuring flexibility.

Briefly summarized, each night, data is extracted from Blaise using Crontab and then stored in the GCP source bucket. Subsequently, the paradata parser script (see Code 3 in the Appendix) and the solution offered by Statistic Norway’s data platform team are employed to parse and transfer the data to the production bucket. As a result, we have updated data that is ready to be analyzed each day.

> back to Table of Contents

3.4 Developing Reports and Programs in JupyterLab

We access and analyze the paradata in a JupyterLab environment within Statistics Norway's cloud-based data platform. JupyterLab's interactive computing environment allows us to combine code, visualizations, and explanatory text in a single Jupyter Notebook. The development of new program reports and our web page is done in the JupyterLab environment in Statistics Norway's cloud-based data platform.

As we focus on developing scripts that contain standardized code that work with all surveys, all team members work within one GitHub repository and aim to develop analyses that can be run on various surveys. Moreover, we use Poetry, a tool for package management in Python, to ensure that every team member works with the same set of libraries. By using Statistics Norway's JupyterLab environment and using GitHub for version control, team members find it easy to collaborate on developing code.

When we initially started to analyze paradata, we often found ourselves repeatedly writing and running code for the same or similar tasks when analyzing the data. Therefore, our team decided to focus on code modularization. Code modularization involves creating modules, which are self-contained units of code that promote reusability and organization. Thus, we aim to identify repeated tasks and write this into modules. The infrastructure of our GitHub repository is set up to be flexible for the user, with modules stored within one folder so they can be imported and used when the user is working in the Jupyter Notebook. Thus, code does not need to be copied and pasted between users or Jupyter Notebooks. For instance, two modules are frequently used when analyzing paradata: data query and paradata manipulation.

As data from each day is stored as separate files in the production bucket, we have created a program (see Code 5 in the Appendix) that queries all the data for a specific survey. The program allows colleagues to query all the data for the specific survey anddata in between two dates, before a specific date, or after a specific date.

Moreover, the data obtained from the paradata parser may sometimes be insufficient for direct analysis, so we made a module (see Code 4 in the Appendix) to manipulate the paradata, creating a more suitable structure for analysis. The module performs various operations, such as sorting observations by SessionId and TimeStamp, filling values of PageIndex and FieldName, and creating a new column labeled "diff_time" to calculate time frames between consecutive observations within each session. The Tables 3 and 4 show how the paradata looks after applying the module. For viewing purposes, the dataset is split into two tables.

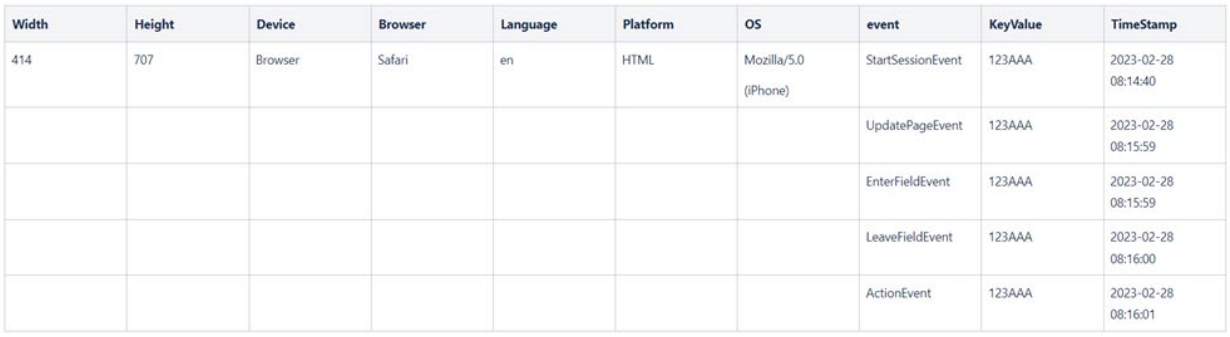
As we aim to continually enhance manipulation processes and conduct quality checks on our programs, we wished to minimize manipulation of the paradata in Step 2 of our pipeline and to instead perform data manipulation in the JupyterLab environment in Step 3. This approach ensures uniformity in the structure of data within the production bucket while we continually develop our code for processing and analyzing paradata.

3.5 Sharing Daily Results to Internal Web Page

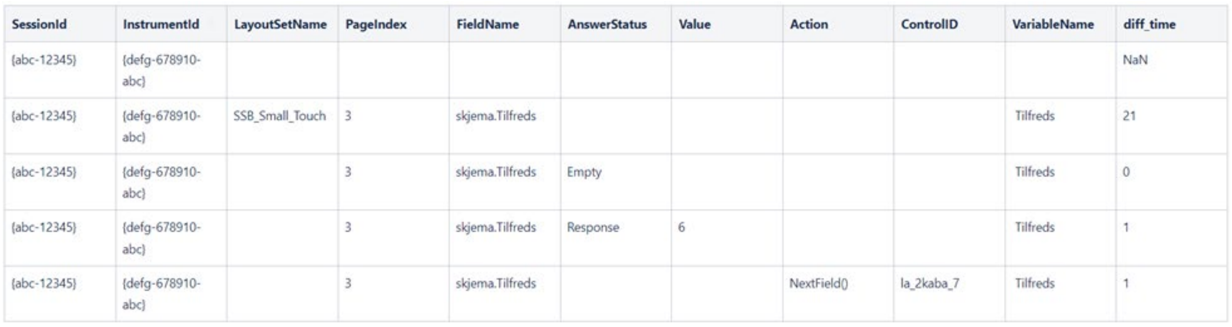
We wished to make it easier for project managers and field staff to see an updated overview of the data collection. Moreover, it was important that seeing daily results should not require any coding skills or access to the actual data. Thus, we decided to create a web page where we could share reports with key metrics for the data collection.

> back to Table of Contents

Tables 3 and 4. Manipulated Paradata



Width	Height	Device	Browser	Language	Platform	OS	event	KeyValue	TimeStamp
414	707	Browser	Safari	en	HTML	Mozilla/5.0 (iPhone)	StartSessionEvent	123AAA	2023-02-28 08:14:40
							UpdatePageEvent	123AAA	2023-02-28 08:15:59
							EnterFieldEvent	123AAA	2023-02-28 08:15:59
							LeaveFieldEvent	123AAA	2023-02-28 08:16:00
							ActionEvent	123AAA	2023-02-28 08:16:01



SessionId	InstrumentId	LayoutSetName	PageIndex	FieldName	AnswerStatus	Value	Action	ControlID	VariableName	diff_time
{abc-12345}	{defg-678910-abc}									NaN
{abc-12345}	{defg-678910-abc}	SSB_Small_Touch	3	skjema.Tilfreds					Tilfreds	21
{abc-12345}	{defg-678910-abc}		3	skjema.Tilfreds	Empty				Tilfreds	0
{abc-12345}	{defg-678910-abc}		3	skjema.Tilfreds	Response	6			Tilfreds	1
{abc-12345}	{defg-678910-abc}		3	skjema.Tilfreds			NextField()	la_2kaba_7	Tilfreds	1

We developed the internal web page by using Quarto[1] and host it with GitHub Pages.[2] Quarto is an open-source solution that combines markdown text and executable Python or R code. When rendering a Quarto file, code is executed, and the output of the code and the markdown text is rendered to HTML files. The collection of HTML files is combined into a complete web page that is hosted with GitHub Pages. As the web page is static, we update it every day by rendering and publishing the complete GitHub repository using Quarto's render and publish commands. By using Quarto, we can utilize the Python reports we create in Step 3 in the JupyterLab environment and then transfer the code to Quarto files for our web page.

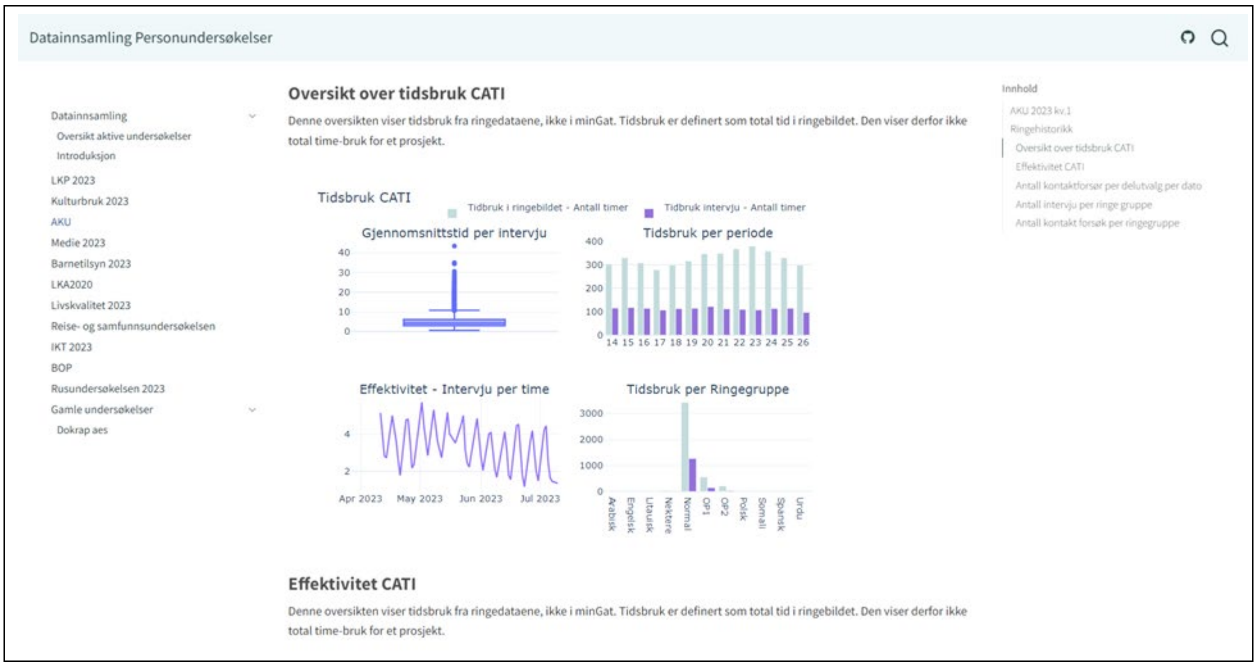
Figure 2 shows a screenshot of our web page with key metrics for ongoing surveys. Each survey instrument is represented on the left, and the user can click on the page to view the instrument they wish to see. The figure shows four different figures of CATI time use for the instrument. On the right, we can see an index of the possible reports available for the instrument.

Quarto and GitHub Pages were chosen due to their ease of use, thereby reducing the need for IT support during setup. Furthermore, GitHub Pages guarantees restricted access solely to authenticated employees of Statistics Norway via their GitHub accounts. While we aim to build a more comprehensive solution in the future, such as an interactive dashboard application, the current solution accomplishes our main objective: sharing results and an overview of the data collection process to colleagues without the need for coding or access to the data.

[1] https://quarto.org/
[2] https://docs.github.com/en/enterprise-cloud@latest/pages/getting-started-with-github-pages/about-github-pages

> back to Table of Contents

Figure 2. Screenshot of Our Internal Web Page



Datainnsamling Personundersøkelser

Datainnsamling
Oversikt aktive undersøkelser
Introduksjon
LKP 2023
Kulturbruk 2023
AKU
Medie 2023
Barnetilsyn 2023
LKA2020
Livskvalitet 2023
Reise- og samfunnsundersøkelsen
IKT 2023
BOP
Rusundersøkelsen 2023
Gamle undersøkelser
Dokrap aes

Oversikt over tidsbruk CATI

Denne oversikten viser tidsbruk fra ringedataene, ikke i minGat. Tidsbruk er definert som total tid i ringebildet. Den viser derfor ikke total time-bruk for et prosjekt.

Effektivitet CATI

Denne oversikten viser tidsbruk fra ringedataene, ikke i minGat. Tidsbruk er definert som total tid i ringebildet. Den viser derfor ikke total time-bruk for et prosjekt.

Innhold
AKU 2023 kv.1
Ringehistorikk
Oversikt over tidsbruk CATI
Effektivitet CATI
Antall kontaktforsør per delutvalg per dato
Antall intervju per ringe gruppe
Antall kontakt forsøk per ringegruppe

4. How The Pipeline Is Used

Creating the pipeline allows colleagues to see daily updates of the data collection. Additionally, because it is easier to access structured paradata, colleagues can complete their own ad hoc analysis or use predefined programs to run reports. The section below includes examples of how colleagues use paradata to evaluate survey questionnaires and dynamically manage data collection.

4.1 Survey Project Managers—Dynamic Management of Surveys

An important responsibility for survey project managers is to continually monitor and manage data collection, for example, by adapting to the amount of interviewer resources available. Having daily updated paradata and sample information can be very beneficial for survey managers. As we have had several survey project managers involved in the development of the pipeline, they have effectively started analyzing paradata and developing reports for the web page.

We have used Dial History data to create key metrics that are used to monitor the data collection process. Dial History data contains information about the most important indicators used to supervise the data collection, such as the number of interviews conducted, response rate among respondents who have been contacted at least once, the number of dropouts, and the average interview time. The average interview time is often estimated during testing, but because it may vary from the actual interview time, it is beneficial to check the average interview time early in the data collection process. Since interviewers inform respondents about the estimated duration of the interview when asking them to participate in the survey, it is important that the estimated interview time is close to the actual interview time.

We used paradata in the Living Conditions Survey 2023 to assess nonresponse bias during the data collection period, which helped us initiate measures to try to reduce that bias. When comparing the distribution in the gross sample with the distribution in the net sample, we can say something about nonresponse bias. By using paradata, combined with grouped sample data for gender, age, and education level, we made a visualization showing nonresponse bias for these characteristics. People aged 25 to 44

> back to Table of Contents

and people with lower education levels were underrepresented in the net sample. We therefore decided to offer incentives for people from these two groups to increase response rates. Without paradata combined with grouped sample data being updated on the web page daily, we would not have been able to monitor the nonresponse bias continually and initiate the exact measures we did, in addition to measuring the effect in retrospect.

In the same survey, we decided to reduce the maximum number of possible calls to respondents because we saw from paradata that the last calls yielded very few interviews compared with the large share of nonresponses. Paradata enabled us to demonstrate the result of each call in an accessible manner, and we could adjust the daybatch settings accordingly. Explicitly, we reduced the maximum number of calls from 15 to 12 in the Blaise CATI dashboard as the 13th, 14th and 15th call proved to be unfruitful in yielding interviews. This insight was particularly important, as we had limited interviewer resources that demanded efficient time allocation.

4.2 Survey Project Managers—Evaluating Survey Questionnaires

We have also used paradata to understand the flow of the Travel and Shopping Survey. Norwegians' travel and shopping habits abroad were previously included as a minor part of a larger questionnaire conducted through telephone interviews. Later, the survey transitioned to a dedicated web-based platform, resulting in significant restructuring and rephrasing of questions. In connection with this, we used the paradata foundation to test a hypothesis about underreporting.

The mode shift and restructuring of questions led to the hypothesis of underreporting for the number of trips taken abroad, as the survey's burden on respondents increases in line with the number of trips. Respondents are first asked how many trips abroad they have taken in the last month, and then they are asked follow-up questions for each trip. The questions are often identical and repeated for each trip. Response burden is often associated with long completion time, poorly formulated questions, and nearly identical repetitive questions (Sharp & Frankel, 1983). Considering this, it is conceivable that respondents may understand the logic of the Blaise questionnaire and reduce the reported number of trips to avoid survey burden and the number of repetitive questions.

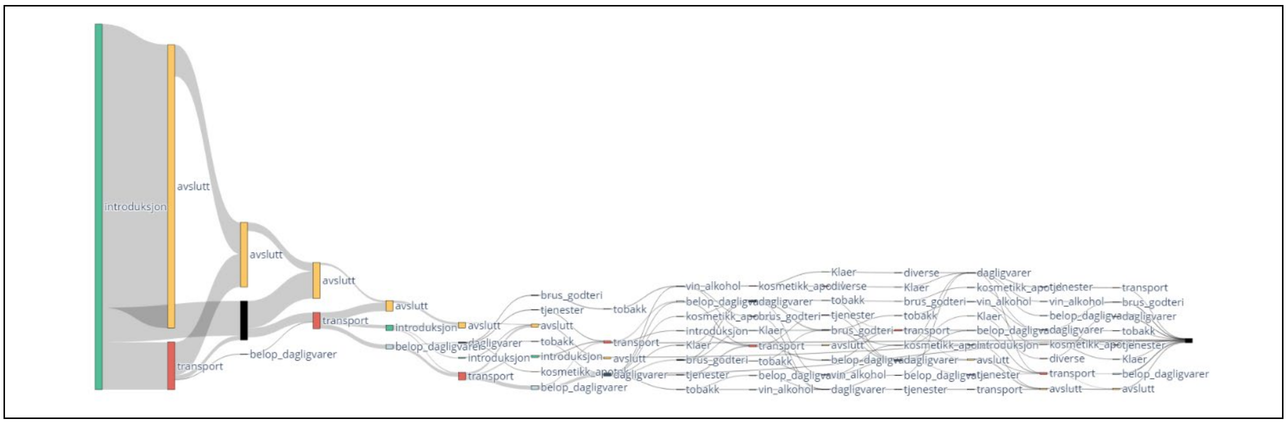
In summary, we utilized paradata to test the hypothesis of underreporting by examining respondents' navigation and user journey through the questionnaire. After unpacking, rows were grouped by respondents and timestamp and were ranked in chronological order. By doing so, it became possible to track the user journey of respondents during questionnaire completion. However, due to the volume of information, it was crucial to find a visualization method that would present the user journey of all respondents in a straightforward manner.

Considering the hypothesis, we used a Sankey diagram, as it provides a clear visual representation of the transition between survey questions, revealing bottlenecks where the respondent either drops off, stalls, or goes back to a previous question. Lastly, if the user journey has multiple paths or options, the Sankey diagram can demonstrate how these paths diverge and converge.

Figure 3 shows a visualization of all recorded user journeys through the survey. As seen in the figure, many respondents move from the introduction to the transport questions, but upon closer examination of the diagram, it is evident that the introduction repeats later in the journey. This provided us with an indication that respondents navigate backward in the questionnaire. After closer inspection, we observed that respondents with more trips often adjusted the reported number of trips to avoid repetitive questions about each trip. This led to adjustments in survey questions and the introduction of a cap on the number of trips the respondent must report.

> back to Table of Contents

Figure 3. Survey Flow in the Travel and Shopping Survey—Sankey Diagram



Paradata proves to be well suited for analyses aimed at gaining insight into the user journey or flow of respondents in a questionnaire. Often, such an analysis can serve as an initial exploration to identify potential weaknesses in the questionnaire. Furthermore, such visualization provides a focus on where to allocate time and effort for both further analysis of reported data and coding of the Blaise questionnaire itself.

4.3 Survey Methodologists—Recruitment and Questionnaire Monitoring

To design new questionnaires or improve existing questionnaires, it is important for survey methodologists to assess how well the questions and questionnaire flow work for respondents to assure high data quality. Various qualitative methods, such as expert reviews, explorative and cognitive testing, and focus groups, are used to assess problems with the questions and improve them. Problems with the questions might lead to a large respondent burden, for example, due to sensitive questions, confusing question formulations, or no suitable answer options, which can lead to inadequate data quality. Quantitative data, such as paradata, can help to determine if problems detected in qualitative testing persist in the data collection. Similarly, paradata, combined with sample data, can give useful insights into whether a problem occurs for a specific group in the population.

Paradata have been available in Statistics Norway for a while, but with higher accessibility through the pipeline, survey methodologists can more easily use paradata. Paradata can be used for pilot surveys and user testing during the data collection process and after the data collection process is completed. Especially for pilot surveys, the use of paradata during the data collection process can be handy for recruiting specific people to focus groups. For instance, we can recruit those who took a long time to answer the survey, those who are registered with several sessions, or those who received several error messages when answering the questionnaire.

For evaluating the quality of questions or questionnaire flow of a survey, various paradata indicators can be used. It is important to keep in mind that analyzing paradata is time-consuming, and it is therefore advisable to map out the questions we want to analyze using paradata and choose which paradata indicators should be used for the analysis. For instance, indicators we often use to assess respondent burden and data quality are:

- Error messages
- Response time
- Previous page

> back to Table of Contents

Error messages can help the respondent avoid obvious mistakes or unanswered questions. Yet, it can be problematic if the error message is not comprehensible to the respondent or if the respondent receives many error messages while answering the survey. If many respondents receive error messages, this can lead to dropout or irritation and might affect the responses for the remaining questions. It may also be an indication of a poorly formulated question.

Analyzing response time has been studied extensively and a review of existing literature has been done by Vehovar and Čehovin (2023). A very short response time can indicate that a respondent does not read the question formulation thoroughly or not at all. On the other hand, a very short completion time could also mean that the respondent is just a particularly quick reader. A very long response time might indicate that the question is difficult to understand or answer. What determines if a response time is too long or too short depends on the type of question. Hence, researchers have to consider the question at hand to determine if a response time is too short or too long.

If many respondents revisit the previous page, it might indicate that there is a problem with the questionnaire flow, for instance, if there is a follow-up question on a new page but respondents require repetition of important information and must revisit the previous page. It might also indicate that the respondents perceive questions as too similar and therefore go backward in the questionnaire to find the difference between the questions.

In sum, paradata can help us to assess data quality and reduce response burden. Although paradata cannot replace qualitative work such as user testing, paradata serve as an additional help for survey methodologists to improve survey questionnaires.

5. Summary

In this article, we have outlined the process of exporting, cleaning, storing, and sharing paradata in our pipeline. The primary objectives for the team responsible for utilizing paradata was to continually provide stakeholders with up-to-date reports of the data collection and to make paradata more easily accessible for colleagues. We achieved these objectives by building this data pipeline. Within the pipeline, we have focused on code modularization and building program reports that can be run on several surveys. In addition to using program reports, colleagues can conduct their own analyses and develop program reports tailored to their specific needs. Our data collection department is composed of colleagues with varying levels of programming proficiency, and everyone has a critical role in data collection procedures. To ensure that everyone has the opportunity to view the outcomes of data collection and participate in enhancing its efficiency, irrespective of their programming proficiency, we share an overview of the data collection process on an internal web page. The examples of how our survey project managers and survey methodologists use paradata highlight the foundational role of our pipeline in shaping a range of crucial decisions. The pipeline facilities effective use of paradata to inform decisions about data collection processes and survey questionnaire development, thereby enhancing several aspects of the data collection.

6. The Way Forward

Since our pipeline project is in its early stages, we wish to improve certain aspects. Firstly, we wish to include more reports on our web page and build more supporting modules and programs for ad hoc analyses. To understand the needs of our colleagues and update them on our progress, we have regular sprint reviews where we show program reports we have developed and new additions to the webpage. These sprint reviews are opportunities for colleagues to suggest improvements and solutions that can help support them in their roles. Secondly, the steps in our pipeline are mostly automated, except for updating the web page. Thus, one of our priorities is to develop a solution that will automatically update our web page. Furthermore, we wish to create a dashboard that will give the user more flexibility with interactivity.

> back to Table of Contents

Our pipeline is a work in progress, and we expect to update and improve aspects of the pipeline as we discover bugs or find improvements. Thus, we suggest that interested readers contact us to see the updated versions of the code shown in this article.

7. References

Cheung, G., Piskorowski, A., Wood, L., & Peng, H. (2016). *Using survey paradata*. 17th International Blaise Users Group Conference, The Hague, Amsterdam, the Netherlands.

Hunt, J. (2016). *Using Audit Trail data to move from a black box to a transparent data collection process*. 17th International Blaise Users Group Conference, The Hague, Amsterdam, the Netherlands.

Kreuter, F., Couper, M., & Lyberg, L. (2010). *The use of paradata to monitor and manage survey data collection*. *American Statistical Association, Proceedings of the oint statistical meetings,* (pp. 282–296). Alexandria, VA, United States.

Schouten, B., Peytchev, A., & Wagner, J. (2017). *Adaptive survey design* (1st ed.). Chapman and Hall/CRC.

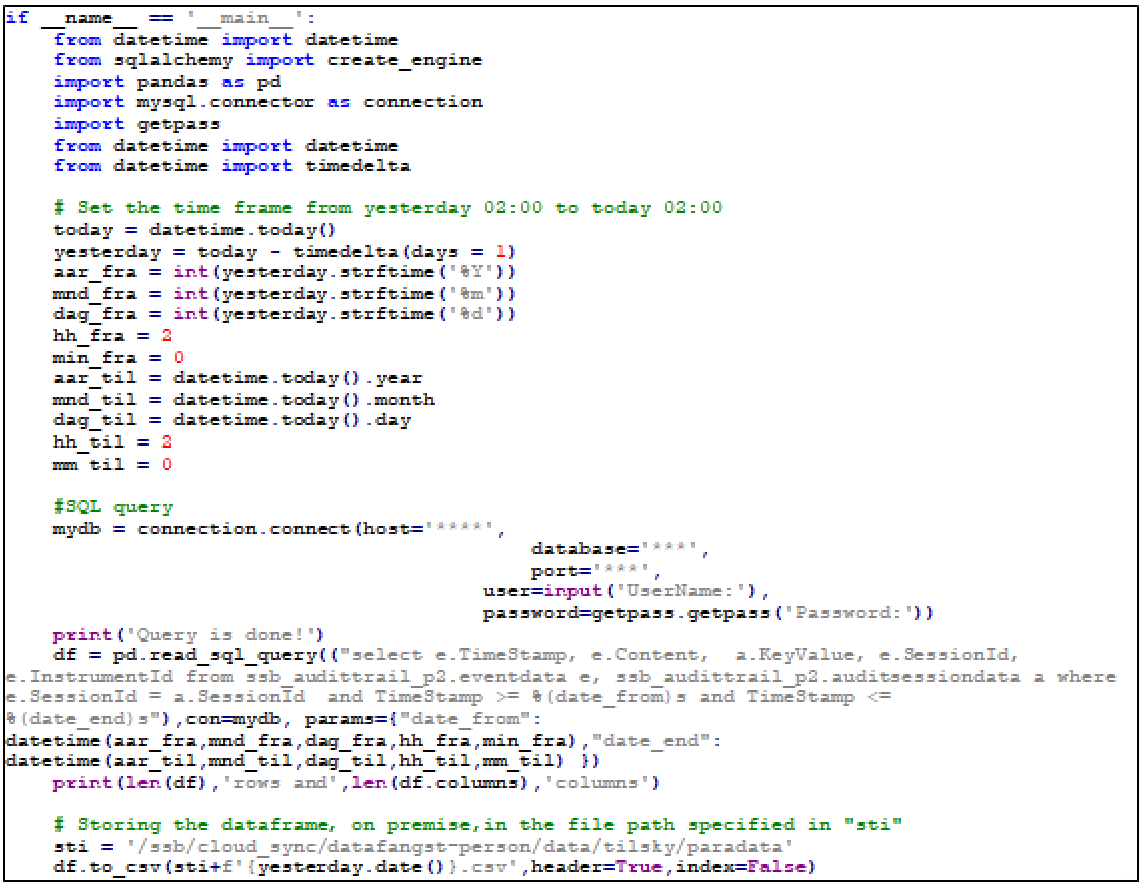
Sharp, L. M., & Frankel, J. (1983). Respondent burden: A test of some common assumptions. *Public Opinion Quarterly*, *47*(1), 36–53.

Vehovar, V., & Čehovin, G. (2023). *Direct paradata usage for analysis of response quality, respondent characteristics, and survey estimates: State-of-the-art review and typology of paradata* (Working Paper). Center for Social Informatics, University of Ljubljana. https://www.fdv.uni-lj.si/docs/default-source/cdi-doc/direct-paradata-usage-for-analysis-of-response-quality.pdf?sfvrsn=0%20

> back to Table of Contents

8. Appendix

Code 1. Exporting Audit Trail Data from Blaise



```
if __name__ == '__main__':
    from datetime import datetime
    from sqlalchemy import create_engine
    import pandas as pd
    import mysql.connector as connection
    import getpass
    from datetime import datetime
    from datetime import timedelta

    # Set the time frame from yesterday 02:00 to today 02:00
    today = datetime.today()
    yesterday = today - timedelta(days = 1)
    aar_fra = int(yesterday.strftime('%Y'))
    mnd_fra = int(yesterday.strftime('%m'))
    dag_fra = int(yesterday.strftime('%d'))
    hh_fra = 2
    min_fra = 0
    aar_til = datetime.today().year
    mnd_til = datetime.today().month
    dag_til = datetime.today().day
    hh_til = 2
    mm_til = 0

    #SQL query
    mydb = connection.connect(host='****',
                                            database='****',
                                            port='****',
                                        user=input('UserName:'),
                                        password=getpass.getpass('Password:'))
    print('Query is done!')
    df = pd.read_sql_query(("select e.TimeStamp, e.Content,  a.KeyValue, e.SessionId,
e.InstrumentId from ssb_audittrail_p2.eventdata e, ssb_audittrail_p2.auditsessiondata a where
e.SessionId = a.SessionId  and TimeStamp >= %(date_from)s and TimeStamp <=
%(date_end)s"),con=mydb, params={"date_from":
datetime(aar_fra,mnd_fra,dag_fra,hh_fra,min_fra),"date_end":
datetime(aar_til,mnd_til,dag_til,hh_til,mm_til) })
    print(len(df),'rows and',len(df.columns),'columns')

    # Storing the dataframe, on premise,in the file path specified in "sti"
    sti = '/ssb/cloud_sync/datafangst-person/data/tilsky/paradata'
    df.to_csv(sti+f'{yesterday.date()}.csv',header=True,index=False)
```

> back to Table of Contents

Code 2. Exporting and Preparing Dial History Data from Blaise

```
from datetime import datetime, timedelta
from sqlalchemy import create_engine
import pandas as pd
import mysql.connector as connection
import getpass
# Create connection
mydb = connection.connect(host='****', database='*****', user=username,
password=getpass.getpass('Password:')
current_time = datetime.now()
# Parse data for the current day
df = pd.read_sql_query(('select * from ****.dialhistory where StartTime >= %(date_from)s and
EndTime <= %(date_end)s")',con=mydb, params={"date_from": datetime(current_time.year, cur-
rent_time.month, current_time.day,0,0), "date_end": datetime(current_time.year,cur-
rent_time.month,
current_time.day,current_time.hour, current_time.minute) })
# Extract and expand each line
a_1  = df['AdditionalData'].str.extract(pat = '("styring.ToWhom" Status=\S*..\S*)', expand =
True)
a_2  = df['AdditionalData'].str.extract(pat = '("io_nummer" Status=\S*..\S*)', expand = True)
a_3  = df['AdditionalData'].str.extract(pat = '("innled" Status=\S*..\S*)', expand = True)
a_4  = df['AdditionalData'].str.extract(pat = '("intslutt" Status=\S*..\S*)', expand = True)
a_5  = df['AdditionalData'].str.extract(pat = '("intervjustatus" Status=\S*..\S*)', expand =
True)
a_6  = df['AdditionalData'].str.extract(pat = '("ioblokk.PeriodeNr" Status=\S*..\S*)', expand
= True)
a_7  = df['AdditionalData'].str.extract(pat = '("ioblokk.Delutvalg" Status=\S*..\S*)', expand
= True)
a_8  = df['AppointmentInfo'].str.extract(pat = '(<StartDate>\S*..\S*</StartDate>)', expand =
True)
a_9  = df['AppointmentInfo'].str.extract(pat = '(<StartTime>\S*..\S*</StartTime>)', expand =
True)
# Extract and expand new lines and give the column names
a1_value = a_1[0].str.extract(pat = '(Value=\S*)', expand = True)
a1_value['value_towhom']  = a1_value[0].str.extract('.*\'(.*)\'.*')
a2_value = a_2[0].str.extract(pat = '(Value=\S*)', expand = True)
a2_value['value_io_nummer']  = a2_value[0].str.extract('.*\"(.*)\".*')
a3_value = a_3[0].str.extract(pat = '(Value=\S*)', expand = True)
a3_value['value_innled']  = a3_value[0].str.extract('.*\"(.*)\".*')
a4_value = a_4[0].str.extract(pat = '(Value=\S*)', expand = True)
a4_value['value_intslutt']  = a4_value[0].str.extract('.*\"(.*)\".*')
a5_value = a_5[0].str.extract(pat = '(Value=\S*)', expand = True)
a5_value['value_intervjustatus']  = a5_value[0].str.extract('.*\"(.*)\".*')
a6_value = a_6[0].str.extract(pat = '(Value=\S*)', expand = True)
a6_value['value_ioblokk.PeriodeNr']  = a6_value[0].str.extract('.*\"(.*)\".*')
a7_value = a_7[0].str.extract(pat = '(Value=\S*)', expand = True)
a7_value['value_ioblokk.Delutvalg']  = a7_value[0].str.extract('.*\'(.*)\'.*')
#endtime - starttime
df['Diff_ES_min'] = ((df['EndTime'] - df['StartTime']).astype('timedelta64[s]')/60).round(2)
#Appointment_StartDate
a8_startdate = a_8[0].str.extract('.*\>(.*)\<.*')
#Appointment_StartTime
a9_starttime = a_9[0].str.extract('.*\>(.*)\<.*')
df['value_towhom'] = a1_value['value_towhom']
df['value_io_nummer'] = a2_value['value_io_nummer']
df['value_innled'] = a3_value['value_innled']
df['value_intslutt'] = a4_value['value_intslutt']
df['value_intervjustatus'] = a5_value['value_intervjustatus']
df['value_ioblokk.PeriodeNr'] = a6_value['value_ioblokk.PeriodeNr']
df['value_ioblokk.Delutvalg'] = a7_value['value_ioblokk.Delutvalg']
df['app_startdate'] = a8_startdate[0]
df['app_starttime'] = a9_starttime[0]
df["Date"] = df["StartTime"].dt.date
dial_history = df[['Id','InstrumentId','PrimaryKeyValue','CallNumber','DialNumber',
'Date','StartTime','EndTime','Status','value_intervjustatus','io_nummer','LastDialTi-
me','DialResult','app_startdate','app_starttime','value_towhom','value_io_nummer','value_inn-
led','value_intslutt','value_intervjustatus','value_ioblokk.PeriodeNr',
'value_ioblokk.Delutvalg','Diff_ES_min']]
# Saving to csv
dial_history.to_csv(path_to_save + f'{datetime.now().date()}.csv', header=True, index=False)
```

> back to Table of Contents

Code 3. Parsing Paradata and Storing the Structured Paradata in a Parquet File

```
# Importing packages
import pandas as pd
import xml.etree.ElementTree as ET
import dapla as dp
import pyarrow as pa

def paradata_parser(raw_paradf):
    datalist = [] # Create an empty list that will contain all the dict
    for i, row in raw_paradf.iterrows(): # Iterate through each row
        dicInner = ET.fromstring(row['Content']).attrib # Transforms the [Conetent] columns by
creating a dict of all found attributes to a dict
        dicInner['event'] = ET.fromstring(row['Content']).tag # add event from the tag
        dicInner['KeyValue']= row['KeyValue']
        dicInner['TimeStamp']= pd.to_datetime(row['TimeStamp'])# add TimeStamp column
        dicInner['SessionId']= row['SessionId'] # add sessionID
        dicInner['InstrumentId']= row['InstrumentId']# add [KeyValue]
        datalist.append(dicInner) # append the row to a dataframe
    df = pd.DataFrame(datalist)  # create a dataframe of all rows
    return df

# Creating a schema where all variables the TimeStamp column is datetime timestamp and all
other variables are string variable
def create_schema(column_names, timestamp_column_name="TimeStamp"):
    fields = []
    for column_name in column_names:
        if column_name == timestamp_column_name:
            fields.append((column_name, pa.timestamp('ms')))
        else:
            fields.append((column_name, pa.string()))
    schema = pa.schema(fields)
    return schema

def main(file_path):
    prod_file_path = "gs://ssb-prod-datafangst-person-data-produkt/Inndata/paradata/"
    para_date = file_path[-22:-4]

    # Read the last file and creat a dataframe
    raw_paradf = dp.read_pandas(f"gs://{file_path}", file_format = "csv")

    # parsing the dataframe with raw paradata through the paradata parser.
    df = paradata_parser(raw_paradf)

    # create a schema
    column_names = df.columns.to_list()
    schema = create_schema(column_names)

    # Write the cleaned  with defined schema
    dp.write_pandas(df = df,
                    gcs_path = f"{prod_file_path}{para_date}.parquet",
                    file_format = "parquet",
                    schema=schema)
    return
```

Code 4. Manipulating Paradata for Analysis

```
def fill_para(table_df):
       # Sorting rows based on SessionId and timestamp.
        table_df = table_df.sort_values(['SessionId','TimeStamp'])

        # Reset the index
        table_df.reset_index(inplace = True, drop = True)

        # fill pageindex downwards
        cols = ['PageIndex']
        table_df.loc[:,cols] = table_df.loc[:,cols].ffill()

        # Grouping on page index and fills FieldName down and then up.
        table_df['FieldName'] = table_df.groupby('PageIndex')['FieldName'].transform(lambda v:
v.ffill())
        table_df['FieldName'] = table_df.groupby('PageIndex')['FieldName'].transform(lambda v:
v.bfill())

        # Create a new variable that contains the short question variable name
        table_df['VariableName'] = table_df['FieldName'].str.rsplit('.', n=1).str.get(-1)

        # Create a variable diff_time. Time in seconds between one observation and the next
within each session.
        table_df['diff_time'] = table_df.groupby(['SessionId'])['TimeStamp'].diff()
        table_df['diff_time'] = table_df.diff_time.dt.total_seconds()

        # Capitalise KeyValue because Blaise is not case-sensitive and Python is.
        table_df['KeyValue'] = table_df['KeyValue'].str.upper()
        return table_df
```

> back to Table of Contents

Code 5. Module to Query Data from the Production Bucket

```
def file_concat(InstrumentId, dager=None, start_dato=None, slutt_dato=None):
    # Importerer nødvendige pakker
    import dapla
    import sys
    from dapla import FileClient
    import pandas as pd
    import os
    import pyarrow.parquet as pq
    import warnings
    warnings.filterwarnings('ignore')
    # Får tilgang til bøtte strukturen
    fs = FileClient.get_gcs_file_system()
    alle_filer_i_bøtta = fs.glob('gs://ssb-prod-datafangst-person-data-produkt/Inndata/ringedata'+
"/*.parquet")
    # LAger tester foir ulike parametre
    alle_dager = ((dager is None) &(start_dato is None) &(slutt_dato is None))
    antall_dager = ((dager is not None)&(start_dato is None)&(slutt_dato is None))
    fra_dato = ((dager is None)&(start_dato is not None)&(slutt_dato is None))
    til_dato = ((dager is None)&(start_dato is None)&(slutt_dato is not None))
    intervall = ((dager is None)&(start_dato is not None)&(slutt_dato is not None))
    alle_filer_i_bøtta = FileClient().ls('ssb-prod-datafangst-person-data-produkt/Inndata/ringedata')
    # Sjekker indexen for ønsket datoer i alle filers liste
    start = ''
    if start_dato is not None:
        for index, element in enumerate(alle_filer_i_bøtta):
            if start_dato in element:
                start = index
        if start == '':
            sys.exit(0)

    elif start_dato is None:
        pass
    else:
        raise UnboundLocalError(f'{start_dato} finnes ikke i bøtta')
    slutt = ''
    if slutt_dato is not None:
        for index, element in enumerate(alle_filer_i_bøtta):
            if slutt_dato in element:
                slutt = index
        if slutt == '':
            sys.exit(0)
    elif slutt_dato is None:
        pass
    else:
        raise UnboundLocalError('Denne datoene finnes ikke i bøtta')

     #Velger filer i som ønsket antall dager
    if alle_dager:
        files = alle_filer_i_bøtta[1:]
    elif antall_dager:
        files = alle_filer_i_bøtta[-dager:]
    elif fra_dato and start != '':
        files = alle_filer_i_bøtta[start:]
    elif til_dato:
        files = alle_filer_i_bøtta[1:slutt+1]
    elif intervall and start!='':
        files = alle_filer_i_bøtta[start:slutt+1]
    # Lager en beholder for ønskede fielr
    out = []
    # Looper gjennom alle ønskede filer
    for file in files:
        file = 'gs://' + file
        out.append(file)
    table_df = pq.ParquetDataset(out, filesystem=fs, filters=[("InstrumentId", "==",
InstrumentId)]).read().to_pandas()
    return table_df
```

> back to Table of Contents

Integrating Blaise 5 and DDI Lifecycle 3.3

Dan Smith, Jeremy Iverson, Colectica

1. Abstract

This paper is pleased to introduce a new Blaise 5 to Data Documentation Initiative (DDI) Lifecycle conversion tool. DDI Lifecycle is an open metadata standard that is used for describing survey specifications and the resulting datasets, along with study, process, and linage information. Colectica Designer has supported a Blaise 4 and Blaise 5 to DDI Lifecycle converter for many years, utilizing a custom-built grammar for the Blaise language. While this has worked well for converting the major structures of a Blaise survey into DDI Lifecycle, not all aspects of the Blaise language were supported. External language definitions within Blaise .bitt files were not processed, as only the Blaise source code was processed and the translations are added later in the .bmix build process. Another issue was keeping the grammar updated with the new language features that are being added into each new Blaise 5 release.

To improve the story for Blaise and DDI Lifecycle integration, Statistics Netherlands and Colectica partnered to jointly develop several DDI tools for the Blaise ecosystem. The first tool developed was the Blaise Colectica Questionnaires tool, which is used to specify a survey specification in DDI Lifecycle and generate corresponding Blaise code. This paper introduces a second tool, a Blaise 5 to DDI Lifecycle 3.3 converter. The Blaise Colectica DDI Connector takes the approach of using the Blaise API directly to process compiled Blaise .bmix survey specifications and .bdix data definitions to create DDI Lifecycle 3.3.

The DDI items are uniquely identified and versioned, and the converter additionally computes hashes that can be used to help locate identical question or answer choices previously documented within question banks. The resultant DDI can then be exchanged or imported into any other tools that support the DDI standard.

2. Introduction

The DDI Lifecycle metadata standard is a comprehensive metadata ontology, primarily designed for documenting and managing survey specifications and statistical datasets. It provides a structured framework for describing the entire data lifecycle, from data collection and processing to preservation and dissemination. The standardization provides a common language and structure for describing data, making it easier for researchers, data producers, and data archives to communicate and understand the content and context of survey data.

2.1 History of DDI

The DDI initiative began in the late 1990s as a collaborative effort among data archives, libraries, and research organizations to develop a standardized way of documenting and managing social science and survey data. The original version of the standard was called DDI Codebook (DDI 2) and focused on single data files. DDI Codebook made no distinction between a variable description and a question description. This version of DDI also lacked unique identifiers for the various metadata being documented, which hampered efforts to reuse, link, and share metadata information across multiple datasets, studies, and organizations.

> back to Table of Contents

Colectica and Statistics Netherlands both had representatives participate in the drafting of a new version of DDI, which became DDI Lifecycle (DDI 3). This newer version had several new capabilities including a particular emphasis on questionnaire specification and metadata reuse:

- **Questionnaire Design and Documentation**: DDI Lifecycle allows researchers and survey designers to document survey questionnaires comprehensively. This includes specifying question wording, response options, skip patterns and routing instructions, multiple languages, and track question and block reuse. This detailed documentation helps maintain the integrity and consistency of surveys.
- **Data Collection**: Information can be tracked using DDI that describes fielding periods, populations of the respondents, and specific versions or revisions of the fielded survey instrument.
- **Data Analysis**: Researchers can use DDI to understand the structure and content of the dataset produced by a survey instrument. This information is crucial for accurately analyzing the data, including identifying the meaning of variables, handling missing data, and understanding the survey's design.
- **Data Sharing and Data Discovery**: DDI-compliant metadata enhance the discoverability of survey data. Researchers and data archives can use DDI metadata to search for relevant datasets and evaluate their fitness for research purposes. Questions, response options, and survey blocks can be reused and linked in documentation.
- **Long-Term Data Preservation**: DDI helps in preserving the context and documentation of survey data over time. This is essential for ensuring the long-term usability and integrity of social science datasets.

2.2 DDI and Blaise Terminologies

The representatives participated in several years of design discussions to ensure that DDI Lifecycle could document the general structure of Blaise surveys. The Blaise rules, blocks, fields, groups, rosters, and computations all correspond to elements of the DDI Lifecycle ontology.

- **Rules**: Rules in the Blaise survey system are akin to DDI's Question Flow and Logic elements. They specify conditional actions or skip patterns that control the flow of questions or the survey path, based on respondent inputs. These correspond to DDI's control constructs, which are nested components that describe when questions are asked and how the survey progresses.
- **Blocks**: In Blaise, blocks are groups of related questions and rules within a survey. These correspond to DDI's sequence control constructs. Sequences captures the structure and organization of questions and content within a survey specification, which is similar to how blocks function in Blaise.
- **Fields**: Fields in the Blaise survey system are equivalent to DDI's Question and Measurement items. They represent individual data elements or questions in the survey. DDI Questions is concerned with documenting variables in terms of their type, labels, question text, response options, and other metadata, which maps to how fields, their types, and role texts are defined and documented in Blaise.
- **Groups**: Blaise groups allow you to organize related questions or fields within a survey into special composite displays. This corresponds to DDI's concept of a Question Grid or Question Block.
- **Rosters**: Rosters in Blaise are used to create dynamic repeating sets of questions, such as household members or survey responses, to a list of items. DDI's Question Grids contain roster dimensions, and the DDI Looping concepts of Loop, RepeatWhile, and RepeatUntil are similar in

> back to Table of Contents

that they capture how certain questions or data elements are repeated for multiple items or cases within a survey.

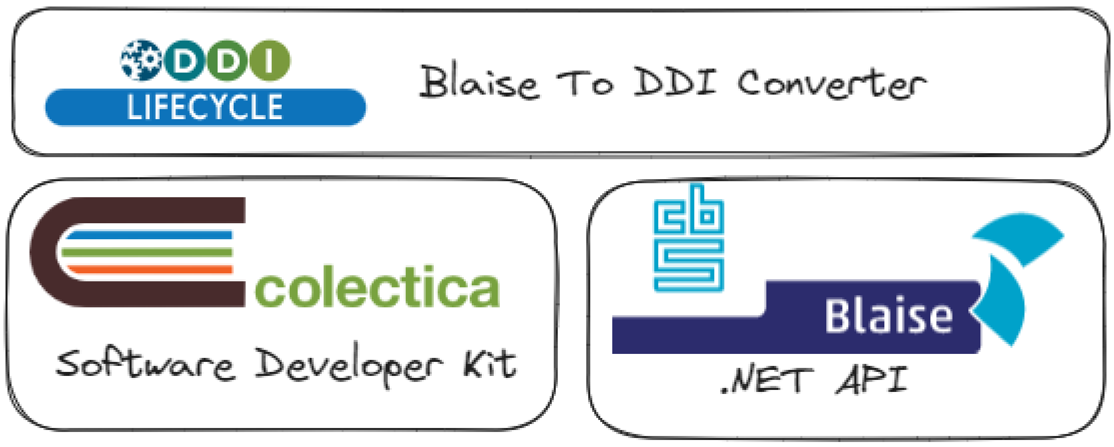
- **Computations**: Blaise allows you to define in the rules section computations or derived variables based on responses to other questions or data and store this information into fields. DDI includes a concept of computations that documents how new data values are computed from existing ones during a survey. This corresponds to how computations are used in Blaise to generate new data based on respondent inputs.

A complete detailed mapping of the Blaise metadata fields to DDI items and properties can be found within the tool distribution.

3. Blaise to DDI Converter

3.1 Components

The Blaise to DDI Converter is a tool that reads compiled Blaise .bmix and .bdix files to generate DDI Lifecycle descriptions of the survey specification and resulting raw dataset. The Blaise .NET API and the Colectica SDK are used together to convert the compiled Blaise metadata structures into DDI Lifecycle metadata.



- **Blaise .NET API**: The Blaise .NET API is a set of libraries distributed as a NuGet package to allow developers to interact with Blaise survey projects programmatically. It provides methods for reading and manipulating Blaise survey instruments and their associated metadata.
- **Colectica SDK**: The Colectica SDK is a software development kit for working with the DDI Lifecycle model. It includes libraries and tools for creating, reading, and exporting metadata in various DDI formats, as well as creating integration Addins for other Colectica tools.

Users of the Blaise to DDI Converter can use either the command line interface to create batch processes or use the GUI user interface. A .bmix file will be chosen to load, and the resulting DDI Lifecycle metadata can be written to a .xml file or saved directly to a Colectica repository. All described metadata content that is generated, including questions and type definitions, will be given globally unique identifiers to allow storage in question banks or type libraries.

> back to Table of Contents

3.2 Reuse with Question Banks and Type Libraries

It is useful to find all instances of identical Blaise fields and types, DDI questions, and codelists that appear across many different Blaise survey instruments during the conversion process. This allows reusing the unique identifiers assigned to metadata items and producing documentation and web portals showing question reuse.

During each conversion of a .bmix to a DDI Lifecycle description of a survey, new identifiers are generated for each metadata item. To allow finding reused questions and types, the Blaise to DDI Converter creates unique hashes of the content of the items. Creating a unique hash of the contents of question text and response options is a useful technique for identifying and finding duplicates that may already be stored in a repository. These hashes serve as fingerprints for the field or type and can be used to compare and match questions efficiently.

DDI Lifecycle and Blaise are multilingual; the normalization process for each piece of text involves sorting the metadata by language tag and creating a string in the format of {language-tag}:{text}:{languagetag}:{text}: and so forth. A similar normalization is done for type codes and multilingual value labels. Text fields of the questions and codelists are concatenated to create a normalized concatenated string. A SHA-256 hash is created from the normalized string and stored as an additional identifier within each metadata item. This hash can now be used as a reference for finding and identifying duplicate questions efficiently.

During the conversion process, the Blaise to DDI Converter can optionally cross-reference the computed SHA-256 hashes in a Colectica repository to see if the question or codelist with identical content has already been created. The converter employs the computed hash and does a search of the repository. If a match is found, the converter can utilize the pre-existing metadata item and its corresponding identifier instead of generating a new duplicate and redundant metadata item. This allows for deduplication during the conversion process instead of as a post-processing step. Using this approach, you can quickly identify duplicate questions and response types without the need for a more resource-intensive textual or semantic comparison.

While this process of using hashes of metadata fields finds exact matches, DDI is also capable of relating similar questions. A question can be associated with a concept. Similar questions can be related by linking to the same concept. Conceptual linking of questions can be addressed as a step after the Blaise to DDI conversion, done by this tool.

4. Summary/Reflections

The introduction of the new Blaise to DDI Converter represents a significant advancement in terms of sustainability and efficiency compared to the previous Colectica Designer source code parser. The new converter leverages the official Blaise API, marking a fundamental shift from the previous approach. This integration with the official API allows the tool to directly access and interact with Blaise .bmix survey projects and their metadata without having to parse Blaise source files. This is a pivotal enhancement because it means the converter is now closely aligned with the Blaise survey platform itself, allowing the utility to easily keep pace with any changes introduced to the Blaise language.

One of the primary advantages of utilizing the official Blaise API is the new converter's ability to adapt to changes in the Blaise language. As Blaise evolves and introduces new features or modifications, the converter can readily accommodate these changes. This adaptability ensures that the tool remains

> back to Table of Contents

compatible with the latest versions of Blaise 5, minimizing the risk of compatibility issues or data parsing errors that may have occurred with the previous Colectica source code parser.

The development of the new converter is a collaborative initiative between CBS (Central Bureau of Statistics) and Colectica. This partnership not only demonstrates a commitment to the ongoing improvement of the tool, but also enhances its stability. The combined expertise and resources of both organizations contribute to the tool's robustness and reliability, and will lead to continuous updates, maintenance, and support for the converter.

DDI Lifecycle places a strong emphasis on reusable metadata descriptions, change tracking, documenting lineage, and unique identification. Whether comparing different draft versions of a survey during the developmental stage or linking different questions across a multitude of surveys and projects, DDI offers a production-ready framework for storing and establishing relationships among this information. When applying this to Blaise fields and response types, many new opportunities for reporting and visualizations can be imagined. While starting with exact matching for linking questions across surveys is a solid foundation, it will be interesting to see what other types of question comparison the community could find useful to be included within the tool during the DDI conversion process.

In summary, the new Blaise to DDI Converter is a sustainable solution that benefits from its integration with the official Blaise API, adaptability to changes in the Blaise language, and the collaborative effort between CBS and Colectica. These factors combine to ensure that the converter will be a dependable and up-to-date tool for converting Blaise survey projects into DDI Lifecycle-compliant metadata.

> back to Table of Contents

Automation Testing Experience with Blaise

Kay Brenner, Gayu Subramanian and Rittu Jittu, Westat

Presenter: Mangal Subramanian

1. Abstract

Automation testing provides many benefits to projects that include time and cost savings, efficiency with regression testing, and consistency across test runs. It also provides better scalability and analysis reports on test coverage. We review our experiences using various automation solutions with Blaise 4.8 and 5, show several demonstrations of these products, and discuss the advantages and disadvantages of the different approaches investigated.

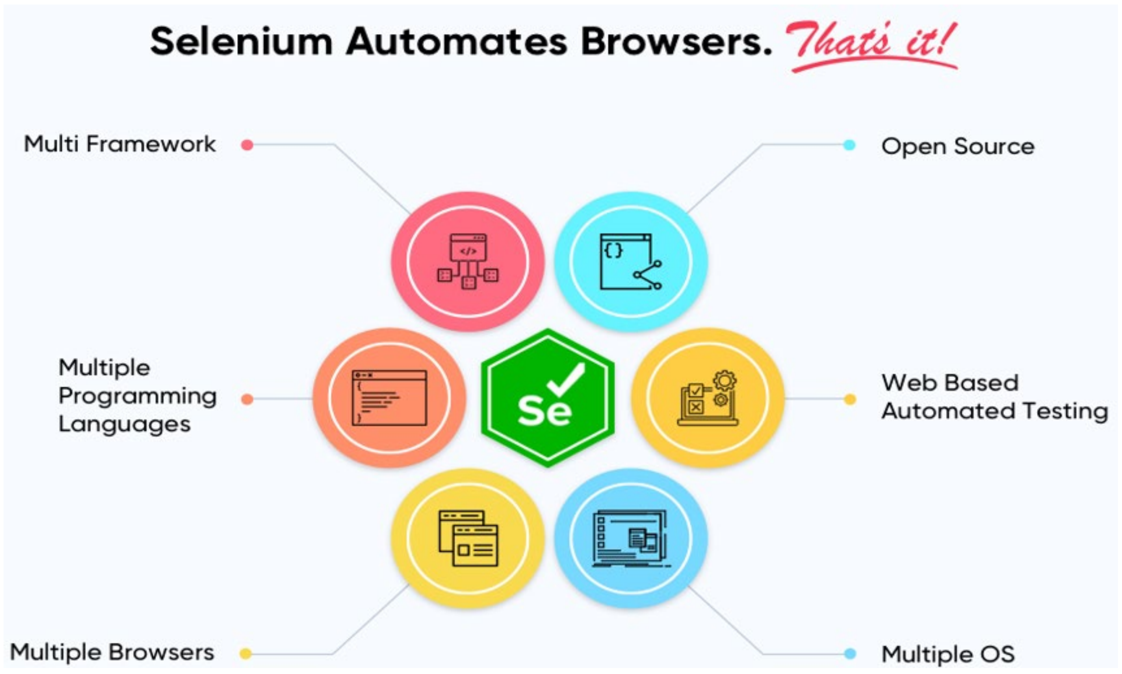
2. Introduction

This past year, we reviewed various automation solutions with Blaise 4.8 and 5, including Selenium, Winium, and Ranorex, and looked at the innovations coming out of Statistics Netherlands this year. Each of the products is discussed below.

2.1 Selenium

It is clear from Selenium's tagline that it is a testing tool for automating web application testing. When it comes to web automation testing tools, Selenium is one of the best. It is an outstanding open-source automation testing tool that can be executed in multiple browsers and operating systems, supporting a considerable amount of programming languages. It is the base for most of the other software testing tools. We are using Selenium WebDriver for automating websites and web surveys. Our Selenium script checks text validations (comparing actual text with expected text), image validations, buttons enabled, web elements displayed, hyperlinks, colors, and bolding of text.

Figure 1. Selenium Automates Browsers


Selenium Automates Browsers. That's it!
Multi Framework
Open Source
Multiple Programming Languages
Se
Web Based Automated Testing
Multiple Browsers
Multiple OS

> back to Table of Contents

2.2 Winium

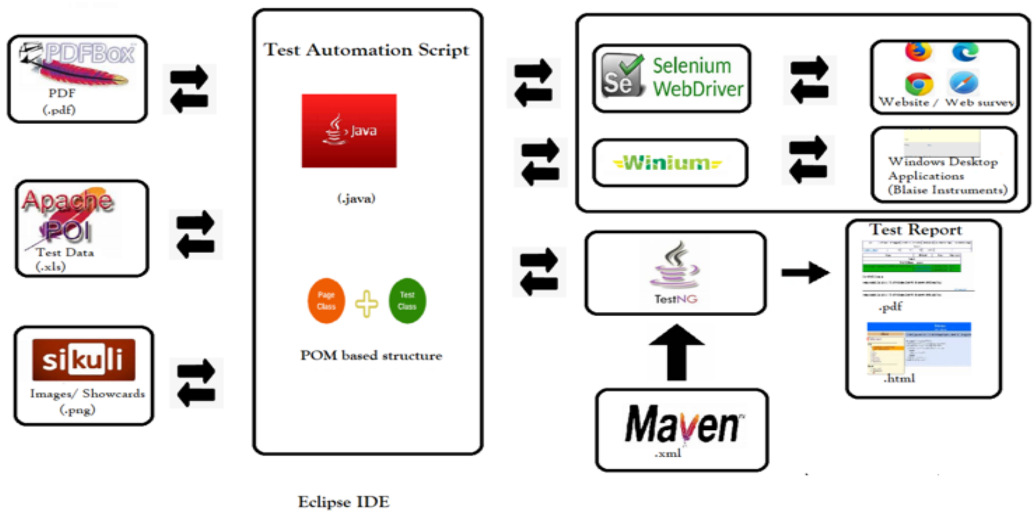
When it comes to testing and automating desktop applications on Windows, Winium is the ideal option. Winium—built on Selenium—is an open-source automation framework used for interacting with Windows applications. This framework can automate any desktop application developed on Windows Presentation Foundation or on Winforms.

Figure 2. Winium

Winium works very well for automating both CAPI and ACASI Blaise surveys. We have created automation scripts for surveys in Blaise 5.13. These scripts run in under five minutes, testing the same task that manually could take 1–2 days to test. Our script checks text validations, image validations, routing, skips, and e-signature entry.

Figure 3 shows the test automation framework.

Figure 3. Test Automation Framework



Java is the programming language used to write the test scripts. We use the TestNG Framework for managing test cases and generating detailed test reports, and we store the project in GitLab for version control. We also use a number of integration tools, for example:

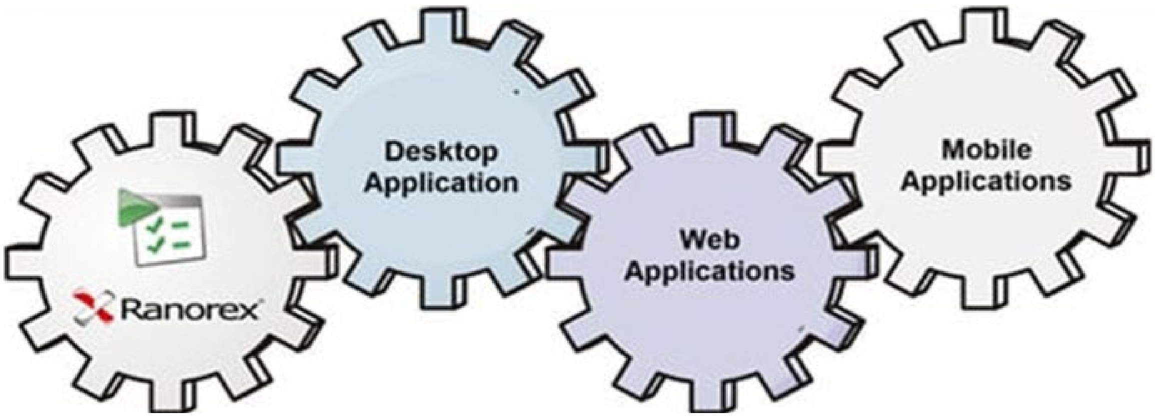
> back to Table of Contents

- Sikuli: validates images, such as show cards
- Apache POI: reads data from Excel files, like preloads values
- Opencsv: reads data from CSV files (data files)
- Apache PDFBox: validates PDF text, such as consent forms

2.3 Ranorex

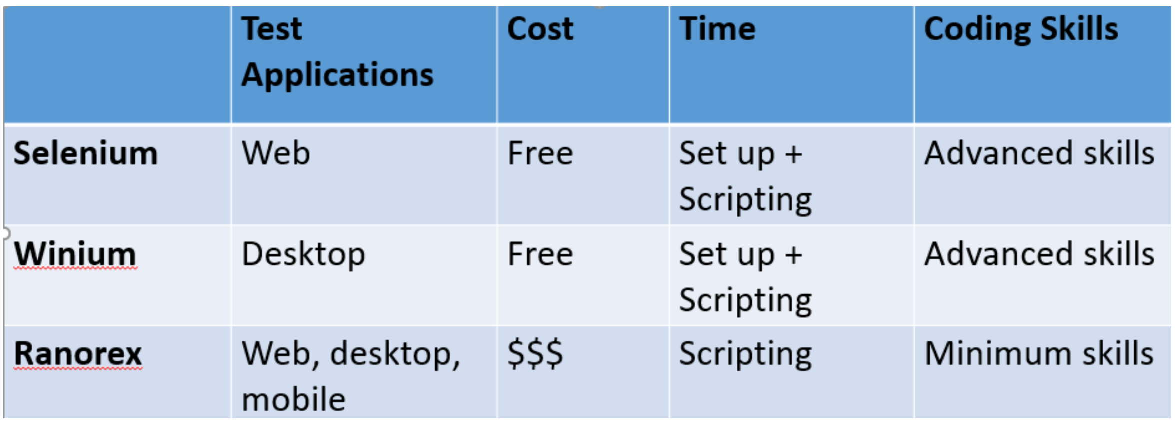
Ranorex Studio is a very powerful tool to automate tests for web, desktop, and mobile applications. It is less programming intensive, with a record/playback structure, so even noncoders can begin to create tests. It supports all technologies (.Net, Java, Flex, HTML) and works in different browsers (IE, Chrome, Firefox) and mobile applications (Android, iOS).

Figure 4. Ranorex



We use Ranorex to test the same validations as Selenium and Winium (text, image, routing validations, etc.) but in ACASI.

Figure 5. Pros and Cons of Different Blaise Automation Solutions



	Test Applications	Cost	Time	Coding Skills
Selenium	Web	Free	Set up + Scripting	Advanced skills
Winium	Desktop	Free	Set up + Scripting	Advanced skills
Ranorex	Web, desktop, mobile	$$$	Scripting	Minimum skills

Comparing our three automation solutions, Selenium gave us the least resistance to identifying Blaise objects when dedicating a Java-experienced test resource, but it’s a web-only tool, so let’s look at other considerations.

> back to Table of Contents

2.3.1 Cost

The open-source tools make it hard to justify the cost of a purchased testing automation tool and measure return on investment if you have short-term goals.

2.3.2 Time

Automated testing is essentially writing code to test other code. Added to the cost is the time spent setting up and maintaining the framework. Sometimes it will result in having to spend more time writing code than actually testing. Having a stable system is imperative; otherwise, resources constantly spend time maintaining test scripts instead of testing. The time payback comes with running automated scripts on a long-term regression system when tests can run in minutes instead of days. Now multiply that by the ability to run tests overnight while your manual testers are sleeping, and you can really reap the benefits of automation.

2.3.3 Skill

A good automation tester needs either basic programming skills and experience or time and aptitude for online learning. Even "record and playback" tools require script adjustments, and scripts need maintenance if something changes in the system under test.

2.4 Future Generation of Blaise Automation

Statistics Netherlands programmers are working on an automation solution that could replace, or at least enhance, our current automation tool(s). The Test Records Generation (TRG) provides a record of test cases that will account for navigating each questionnaire statement at least once. The TRG test record values can then be used to get the minimum number of test cases that exercise all statements without having to run every, or even duplicate, routing tests. A test log file will contain errors found while implementing the test records in a Blaise engine, including missing expected values and other test record values. Blaise 5.14 will introduce a separate test tool that offers TRG, and other solutions are upcoming.

2.5 Summary

We found a combination of tools suited our needs and settled on Selenium for websites and web surveys and Ranorex for Blaise 4.8 and 5.13 CAPI and ACASI surveys. The biggest payback on automation resources comes when you have scripts that can run overnight on a stable system. We use a premium Ranorex license for creating and maintaining test scripts and runtime licenses for running the test scripts.

When working on a stable system, testers and management will see benefits from automation testing over manual testing in these expected ways:

2.5.1 Fast and Efficient Testing

Automation testing runs tests much faster and more efficiently than manual testing. A large number of tests running in a shorter amount of time allows for faster detection of issues.

2.5.2 Consistent and Reliable

Automation testing eliminates human error and executes the same tests every time, improving the accuracy and consistency of the test results.

> back to Table of Contents

2.5.3 Cost-Effective

Although there is an initial investment in developing automated test scripts, if you have a stable system, it is more cost-effective in the long run than manual testing. The repetition of running automated tests without additional costs ensures the cost per test decreases over time.

2.5.4 Better Regression Testing

Automation testing can quickly identify regression issues and detect whether new changes have affected existing functionalities, thus reducing the risk of introducing new bugs or issues.

2.5.5 Increased Test Coverage

Manual testing puts limits on how many tests you can verify. Automation allows you to spend time writing new tests and adding them to your automated test suite. This increases the test coverage and reduces the risk of defects.

2.5.6 Scalable

Project requirements can determine the number of users, modules under test, executions, and test cases, making it an ideal solution for projects with stable systems.

> back to Table of Contents

Experiences in Accessibility and 508-Compliance Testing at RTI

Al-Nisa Berry, Emily Caron, Melissa Page, and Rhymney Weidner, RTI International

1. Abstract

As surveys become increasingly web-based, and the federal government—as well as other clients—frequently require accessibility standards to be met, it grows ever more important for developers to have flexibility and relative ease in applying accessibility-related attributes to survey controls. Blaise 5 includes many new features to help foster the development of an accessible survey; however, there remains room for improvement. This paper will cover some of the discrepancies between RTI's testing with ANDI (Accessible Name & Description Inspector) and what passed for the NVDA (NonVisual Desktop Access) screen reader. It will also describe the overall challenges we faced, along with some solutions applied while developing a 508-compliant web survey in Blaise 5.

2. Introduction

While we have used Blaise 5 successfully on many projects, in fall 2022, we endeavored to use it on our first project with the goal of being completely 508 compliant. Blaise 5.12.8 was the latest production version of Blaise 5 available when we began serious development, and so it was the version selected for use on this project. The survey in question was to be completed in CAWI and CATI mode. CAWI mode was designed with two sets of layout templates, one "large" for browsers with a width greater than 600 px and one "small" for browsers with a width less than 600 px. CATI mode would also be completed in a web browser, with slight adjustments to the layout and text compared to what was used for the CAWI "large" version. 508-compliance testing was conducted only on the CAWI "large" layout set and evaluated using the Chrome browser. 508 guidelines require the survey to follow all WCAG 2.0 rules to be compliant.

The 508-compliance testing team at RTI International, which is comprised of RTI QA staff, utilized the accessibility testing tool created by the Accessible Solutions Branch of the Social Security Administration called "ANDI." This tool is a Javascript-based tool that is easily installed in a variety of browsers, including Chrome, Edge, Firefox, Safari, and Internet Explorer.

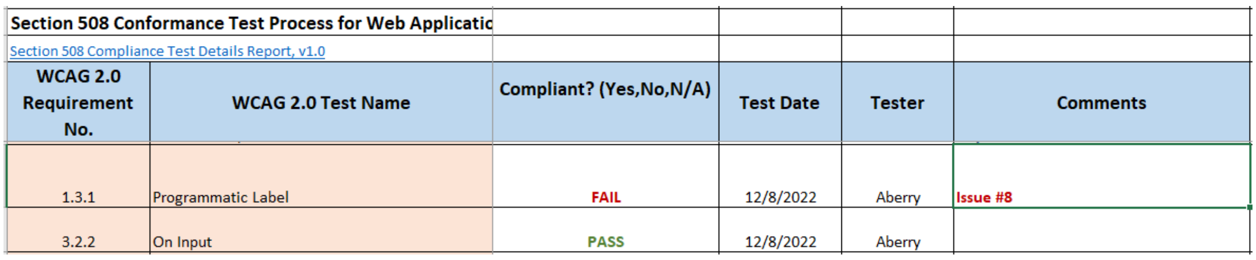
While we encountered many issues during our journey to 508 compliance, this paper will cover representative issues from each of these areas: compliance issues that could not be solved, issues where we found workarounds to achieve compliance, and issues where we communicated with the Blaise 5 development team for potential solutions.

3. Testing Results

At every stage, the 508-compliance testing team provided testing results in easy-to-read Excel and Word documents. The Excel document listed each WCAG 2.0 requirement and the results for that item in the tested survey. Any issues discovered were listed in more detail in a separate Word document that provided a detailed description of the issue, along with screenshots. The screenshots provided specific screens for testing potential fixes, as well as making it clear what features in ANDI were used to identify the problem.

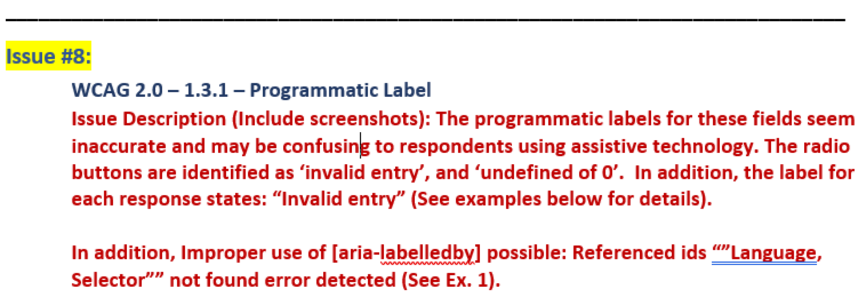
> back to Table of Contents

Figure 3a. An Example of Results Provided in Excel after Completing 508-Compliance Testing



Section 508 Conformance Test Process for Web Applicatio					
Section 508 Compliance Test Details Report, v1.0					
WCAG 2.0 Requirement No.	**WCAG 2.0 Test Name**	**Compliant? (Yes,No,N/A)**	**Test Date**	**Tester**	**Comments**
1.3.1	Programmatic Label	FAIL	12/8/2022	Aberry	Issue #8
3.2.2	On Input	PASS	12/8/2022	Aberry	

Figure 3b. An Example of a More Detailed Issue Description, Which Was Provided in a Separate Word Document; The Word Document Detailed Results Also Provide Screenshots to Help Isolate Fields/Screens Where the Problem Was Located



Issue #8:

WCAG 2.0 – 1.3.1 – Programmatic Label

Issue Description (Include screenshots): The programmatic labels for these fields seem inaccurate and may be confusing to respondents using assistive technology. The radio buttons are identified as 'invalid entry', and 'undefined of 0'. In addition, the label for each response states: "Invalid entry" (See examples below for details).

In addition, Improper use of [aria-labelledby] possible: Referenced ids ""Language, Selector"" not found error detected (See Ex. 1).

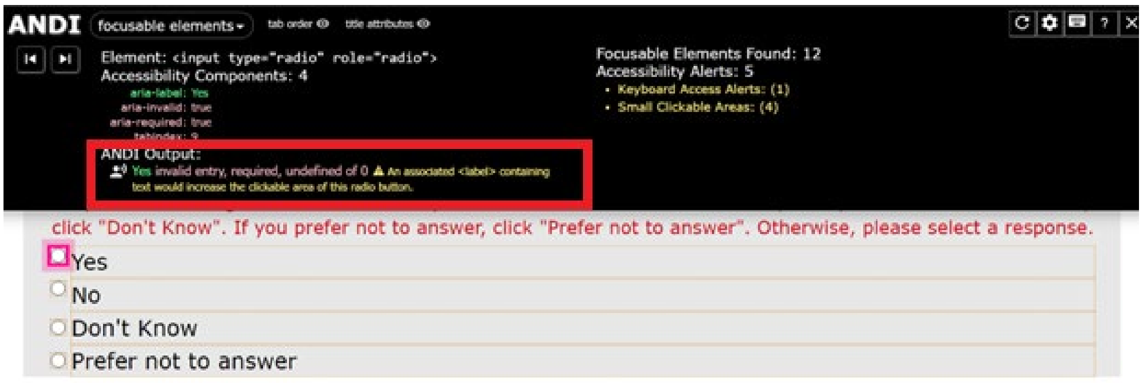
4. Initial Layout Testing

Before development began on the full instrument, we developed a small test instrument in Blaise 5.12.7 that contained one of each question type expected in the real survey. Early testing revealed issues in our test project. Some were issues that could be easily corrected. For example, error messages must be descriptive and specifically state what needs to be entered to correct the issue. This could be solved by implementing a role specifically for error text and making sure it was utilized where necessary. One caveat with this correction, however, is that additional role texts will require translation for surveys conducted in more than one language.

A few issues were quickly determined to be bugs and were corrected by Team Blaise in the next build of Blaise 5 (Blaise 5.12.8). Other issues were not so easy to solve, and some we were unable to resolve. Examples of these issues are demonstrated in the figures below.

> back to Table of Contents

Figure 4a. Programmatic Labels Were Not Accurate; For Example, Radio Buttons Were Labeled as “Undefined of 0” Instead of a Name That Described the Option That Would Be Selected by That Radio Button

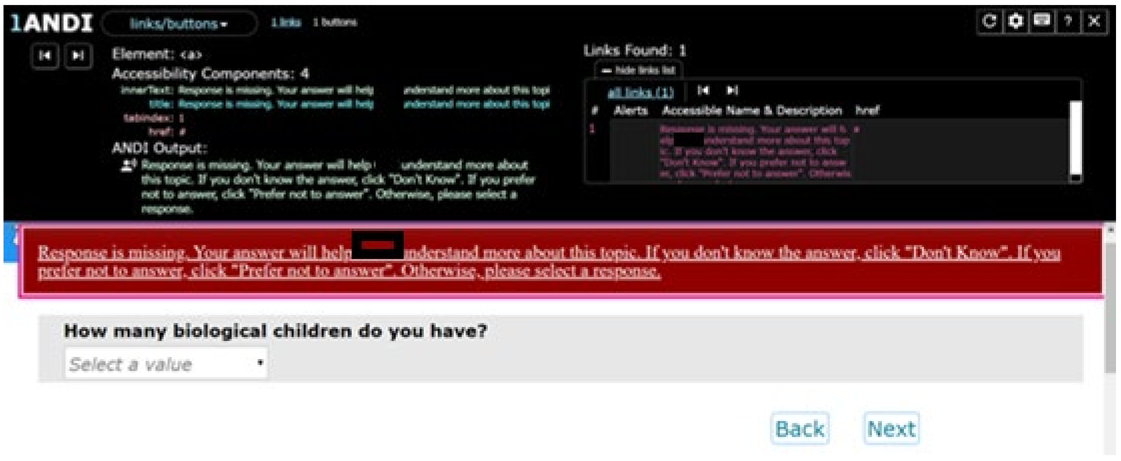


```
Element: <input type="radio" role="radio">
Accessibility Components: 4
      aria-label: Yes
    aria-invalid: true
   aria-required: true
        tabindex: 9
ANDI Output:
  Yes invalid entry, required, undefined of 0 ⚠ An associated <label> containing
  text would increase the clickable area of this radio button.

Focusable Elements Found: 12
Accessibility Alerts: 5
 • Keyboard Access Alerts: (1)
 • Small Clickable Areas: (4)
```

click "Don't Know". If you prefer not to answer, click "Prefer not to answer". Otherwise, please select a response.

- Yes
- No
- Don't Know
- Prefer not to answer

Figure 4b. Text on the Page Is Denoted as a Link, but It Is Not Visible; The Element Is a Hidden Error Message



```
Element: <a>
Accessibility Components: 4
  innerText: Response is missing. Your answer will help      understand more about this topi
      title: Response is missing. Your answer will help      understand more about this topi
   tabindex: 1
       href: #
ANDI Output:
  Response is missing. Your answer will help      understand more about
  this topic. If you don't know the answer, click "Don't Know". If you prefer
  not to answer, click "Prefer not to answer". Otherwise, please select a
  response.

Links Found: 1
#  Alerts  Accessible Name & Description  href
1          Response is missing. Your answer will h  #
           elp [redacted] understand more about this top
           ic. If you don't know the answer, click
           "Don't Know". If you prefer not to answ
           er, click "Prefer not to answer". Otherwis
```

Response is missing. Your answer will help [redacted] understand more about this topic. If you don't know the answer, click "Don't Know". If you prefer not to answer, click "Prefer not to answer". Otherwise, please select a response.

How many biological children do you have?

Select a value

Unfortunately, we were unable to find a way to fix item #1 (Figure 4a) in Blaise 5.12.8.

Oddly enough, we were able to fix item #2 (Figure 4b) by *turning off* some of the built-in Blaise 5 accessibility settings. By unchecking the “Use Skip Links” option under “Accessibility Options” on the “Data Entry” tab under “Settings,” this 508-compliance failing was corrected.

5. Full Instrument Testing

Similar documents were produced for 508-compliance testing on the full instrument. Due to time constraints, the 508-compliance testing occurred as soon as the error messages and screen layouts were finished.

Testing on the full instrument revealed issues similar to those discovered during testing of the smaller instrument, as well as additional problems. While developing the test project, we had not known our final instrument would contain some large tables. These large tables presented multiple 508-compliance issues, but it was determined that the benefits of using the table outweighed the disadvantages of not being able to achieve full 508 compliance on these screens. The decision was made to keep the tables.

> back to Table of Contents

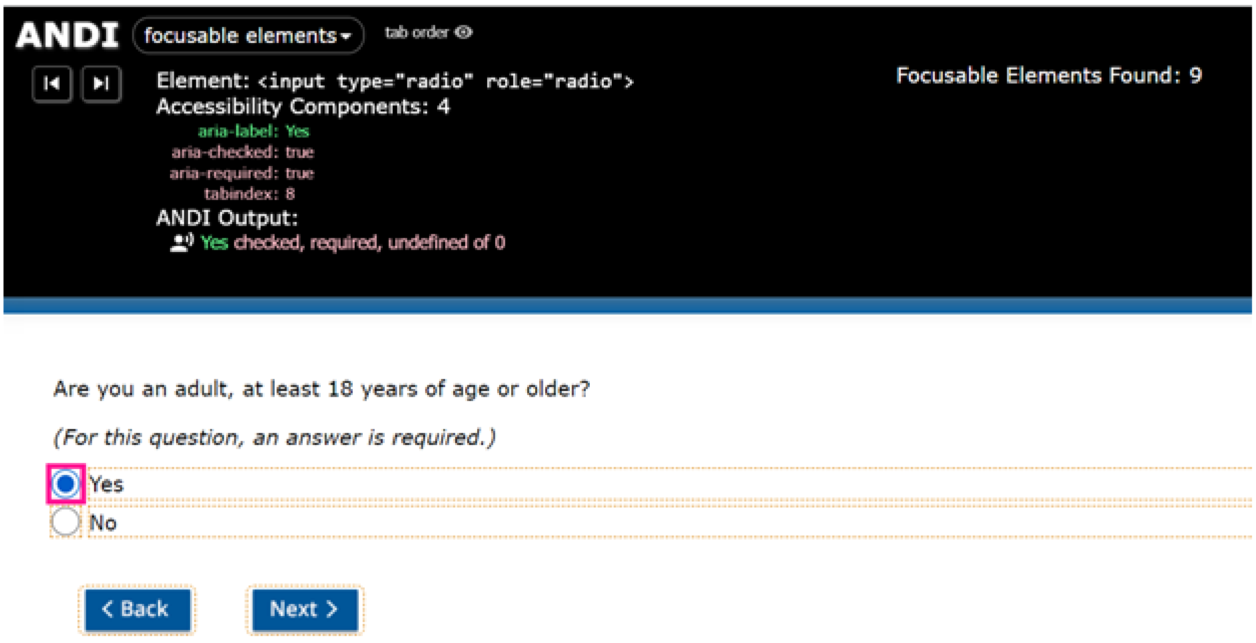
Issues that were already discovered during initial testing continued to cause problems in the full instrument. While design changes were considered to remedy these issues, in the end, it was determined that remediation was the better option. Rather than risk compromising the interview experience for the majority, we decided the option to complete the survey via CATI mode provided a suitable alternative for any users experiencing accessibility issues.

6. Compliance Issues That Could Not Be Solved

6.1 Enumerated Fields—Checked/Unchecked Responses

One of the first issues encountered involved all enumerated fields, which were quite numerous in the survey. WCAG 2.0 Success Criterion (SC) 1.3.1 requires programmatic labels that are meaningful and accurate. Unfortunately, for Blaise 5, we were unable to find a way to mark this information in a way that ANDI would recognize. In this case, the NVDA screen reader would read the correct responses, regardless of this ANDI-detected issue.

Figure 6a. Blaise 5 Correctly Indicates "Checked" for the Selected Response but Does *Not* Indicate "Unchecked" (aria-checked: false) for the Unselected Response and Does Not Include the Correct Number of Response Options



```
Element: <input type="radio" role="radio">
Accessibility Components: 4
     aria-label: Yes
   aria-checked: true
  aria-required: true
       tabindex: 8
ANDI Output:
  Yes checked, required, undefined of 0
```

Focusable Elements Found: 9

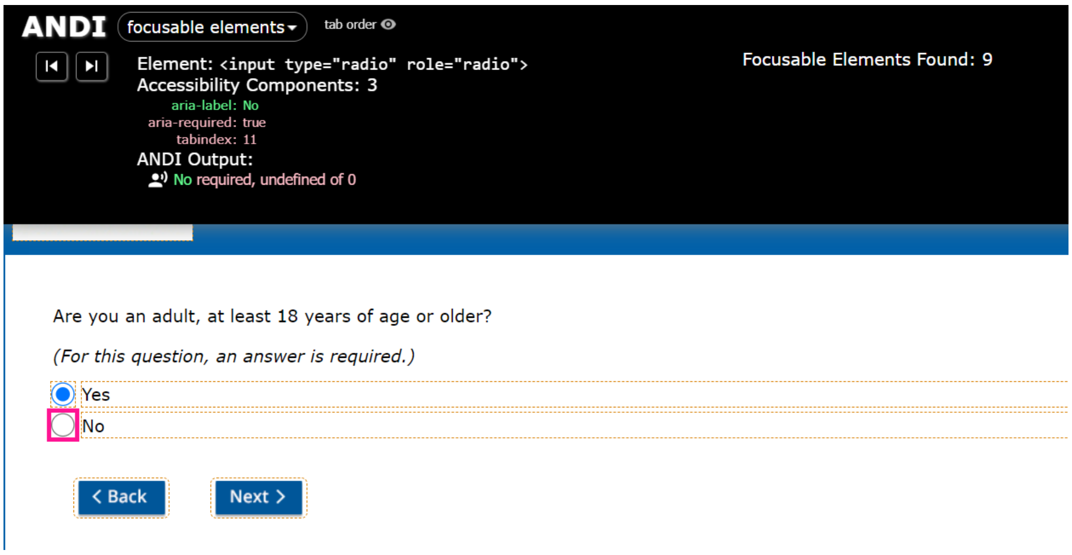
Are you an adult, at least 18 years of age or older?

(For this question, an answer is required.)

[✓ Yes]
No

> back to Table of Contents

Figure 6b. Blaise 5 Does Not Indicate "Unchecked" for Responses That Are *Not* Selected



```
Element: <input type="radio" role="radio">
Accessibility Components: 3
     aria-label: No
  aria-required: true
      tabindex: 11
ANDI Output:
  No required, undefined of 0
```

Focusable Elements Found: 9

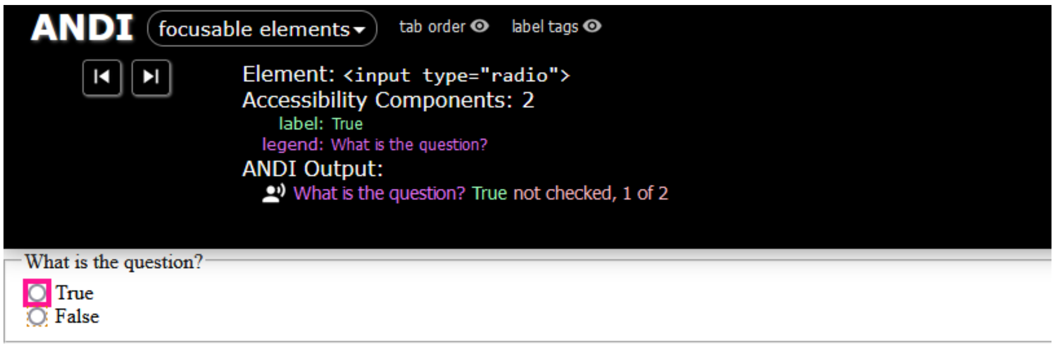
Are you an adult, at least 18 years of age or older?

(For this question, an answer is required.)

● Yes
○ No

< Back   Next >

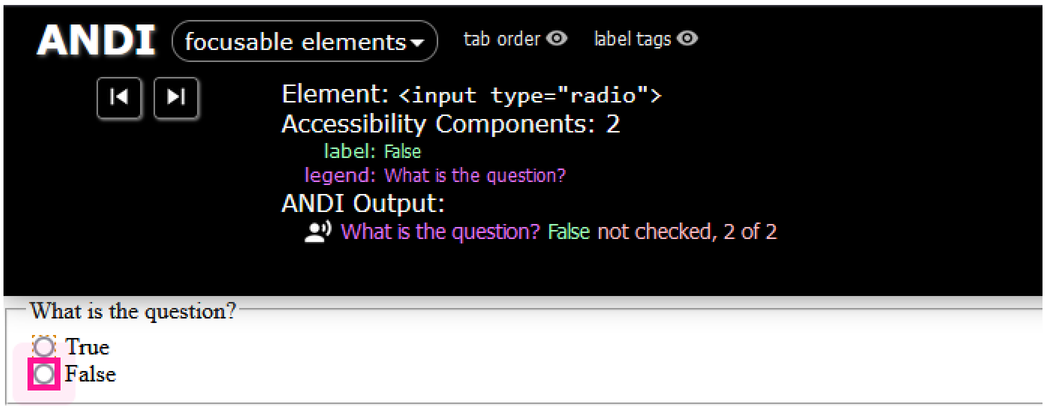
Figure 6c. An Example of a Radio Button Set That Correctly Indicates Checked/Unchecked and the Number of Response Options Available for a Selected Response



```
Element: <input type="radio">
Accessibility Components: 2
   label: True
 legend: What is the question?
ANDI Output:
  What is the question? True not checked, 1 of 2
```

What is the question?
○ True
○ False

Figure 6d. An Example of a Radio Button Set That Correctly Indicates Checked/Unchecked and the Number of Response Options Available for an Unselected Response



```
Element: <input type="radio">
Accessibility Components: 2
   label: False
 legend: What is the question?
ANDI Output:
  What is the question? False not checked, 2 of 2
```

What is the question?
○ True
○ False

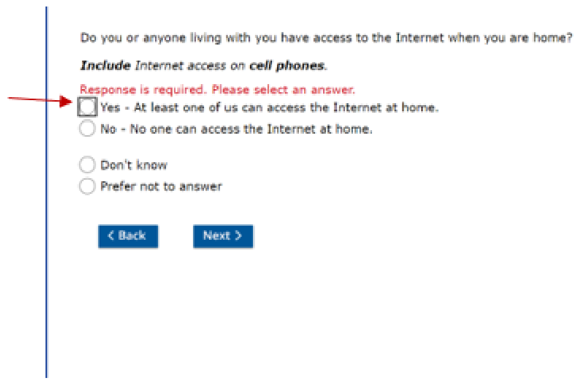
> back to Table of Contents

6.2 Focus Order Reveal—Cursor Does Not Move to Special Answer Response Options

SC 2.0–2.4.3 of WCAG requires the focus to automatically move to additional content that is revealed. In this particular survey, "Don't Know" and "Prefer Not to Answer" (DK/RF) special answer options are not initially visible. These options appear at the bottom of the valid options list only if the respondent attempts to leave the page without providing a valid response. Unfortunately, Blaise 5 does not allow for adjustments to the focus order and refreshes the page when special answers are displayed using the "Hide Special Answers the first time a respondent enters a page" option, which then sends the focus back to the first response option on the page (not the newly displayed DK/RF).

As a potential workaround to this issue, we considered relocating the DK/RF options to the top of the page so that the focus would automatically shift to these options when they were displayed. However, it was determined by our survey methodologist that this positioning may be potentially detrimental to survey results, and that—along with client preference—informed the decision to keep the DK/RF options at the bottom of the page.

Figure 6e. Displaying the DK/RF Options after the Other Response Options Caused Issues with the Focus Order Reveal Requirement, but Blaise 5 Did Not Provide Options for Adjusting the Focus Order


Do you or anyone living with you have access to the Internet when you are home?

***Include** Internet access on **cell phones**.*

Response is required. Please select an answer.

- Yes - At least one of us can access the Internet at home.
- No - No one can access the Internet at home.

- Don't know
- Prefer not to answer

Notice: The system remains on the same questions and displays two additional responses, i.e., Don't Know and Prefer not to answer. The cursor focus moves to the 'Yes' response, instead of 'Don't Know' response as specified by the WCAG 2.0 – 2.4.3 Focus Order Reveal requirement.

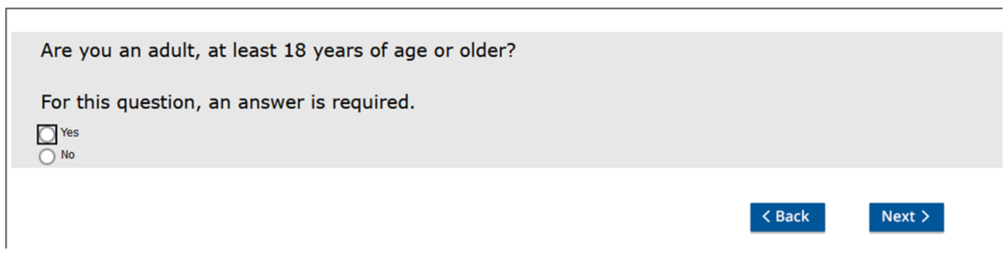
6.3 Reading Order of the Content

Dropdown fields also had their own set of problems. SC 1.3.1 of WCAG 2.0 requires that the reading order of the content (in context) is correct. The meaning of the content should be preserved without CSS positioning. On dropdown fields, selected responses were lost when the "Linearize Page" option in ANDI was used. This option removes CSS positioning from the elements on the page.

While Blaise 5 *does* have an option to "Optimize for readability without stylesheets" that may have helped with this issue, we were unable to utilize it. Whenever we turned this option on, it caused text formatting issues for the response options (see Figure 6f below for an example), changing the text on the response options to 12pt font when they really should have matched the question text at 20pt font.

> back to Table of Contents

Figure 6f. Turning on the “Optimize for Readability without Stylesheets” Caused the Response Option Text to Become Tiny (the Response Options Text Size Should Match the Size of the Question Text)


Are you an adult, at least 18 years of age or older?

For this question, an answer is required.

Figure 6g. The Selected Response Option Is Visible Prior to Utilizing the “Linearize Page” Option in ANDI

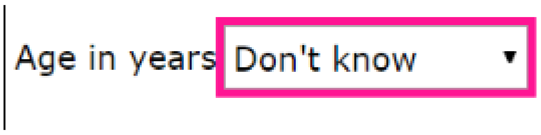
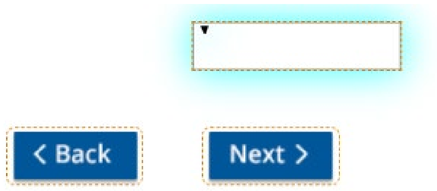


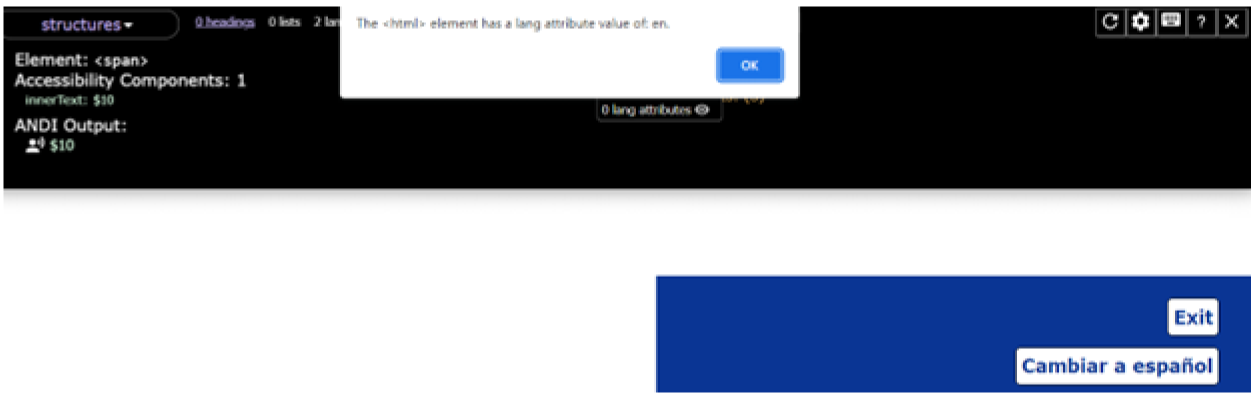
Figure 6h. When the “Linearize Page” Option Is Turned on, the Selected Response and Associated Label Disappear from the Screen



6.4 Part Language Defined

The overall page language is defined by Blaise 5, but we were unable to set the language on individual parts of a page. For example, the language selector was displayed in the opposite language than the one used on the page and, as such, should have been defined separately from the rest of the page. This definition is a requirement in SC 3.1.2 of WCAG 2.0.

Figure 6i. The Page Is Defined in English (en) and There Are No Additional Language Tags Defined (Shown Here as “0 lang attributes”), but There *Should* Be One Defined for the “Cambiar a español” Button


The <html> element has a lang attribute value of: en.

Element: <span>
Accessibility Components: 1
innerText: $10
ANDI Output:
$10

7

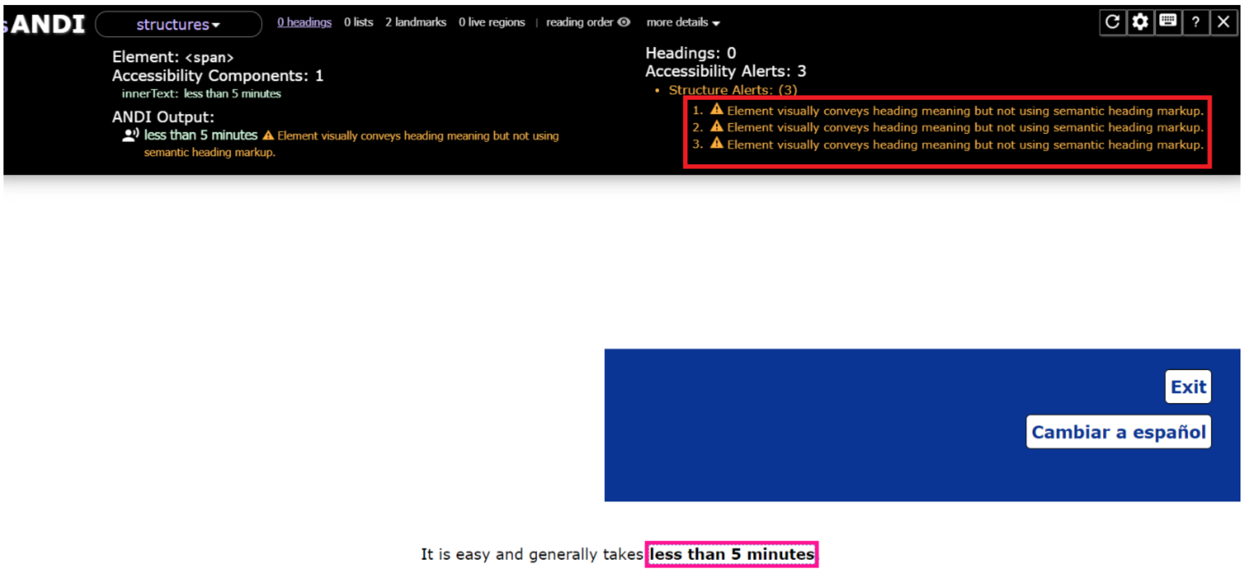
> back to Table of Contents

The Blaise 5 Help suggested naming individual parts on a page by setting the language tag for rich text elements. For example, placing text between these starting and ending tags, <lang value=”fr”></lang>, would specify that the enclosed text is in French. We attempted to implement these tags for the select language option, but Blaise 5’s <lang> tags are not recognized by ANDI. Regardless of the lack of recognition by ANDI, we left the <lang> tags in the final survey in case they are recognized by some accessibility software.

6.5 Font Enhancements and HTML Links

In traditional HTML, <strong> and <emphasis> tags should be used to bold or italicize text in a way that promotes accessibility. Blaise 5, on the other hand, utilizes its own set of tags to denote bold or italicized text. Unfortunately, the Blaise 5 method of bolding text is recognized as a “heading” by ANDI, which causes issues. WCAG 2.0 SC 1.3.1 specifies that every heading that can be programmatically determined is a visual heading, and that every visual heading can be programmatically determined. We tried utilizing <strong>/<emphasis> tags, but they were not recognized by Blaise 5 as marking text that should be bolded/italicized.

Figure 6j. Text Bolded Using Blaise 5 Tags Is Recognized as a Heading by ANDI



Element: <span>
Accessibility Components: 1
innerText: less than 5 minutes
ANDI Output:
less than 5 minutes ⚠ Element visually conveys heading meaning but not using semantic heading markup.

Headings: 0
Accessibility Alerts: 3
• Structure Alerts: (3)
1. ⚠ Element visually conveys heading meaning but not using semantic heading markup.
2. ⚠ Element visually conveys heading meaning but not using semantic heading markup.
3. ⚠ Element visually conveys heading meaning but not using semantic heading markup.

It is easy and generally takes **less than 5 minutes**

A similar issue occurred with HTML links. In standard HTML, links are designated using <a href> tags, but Blaise 5 utilizes its own <hyperlink> tag. This hyperlink tag is not recognized by ANDI as a link or focusable element. This lack of recognition causes issues with WCAG 2.0 SC 2.4.4, which requires the purpose of each link to be able to be determined from the link text alone or by both the link text and programmatically determined link context.

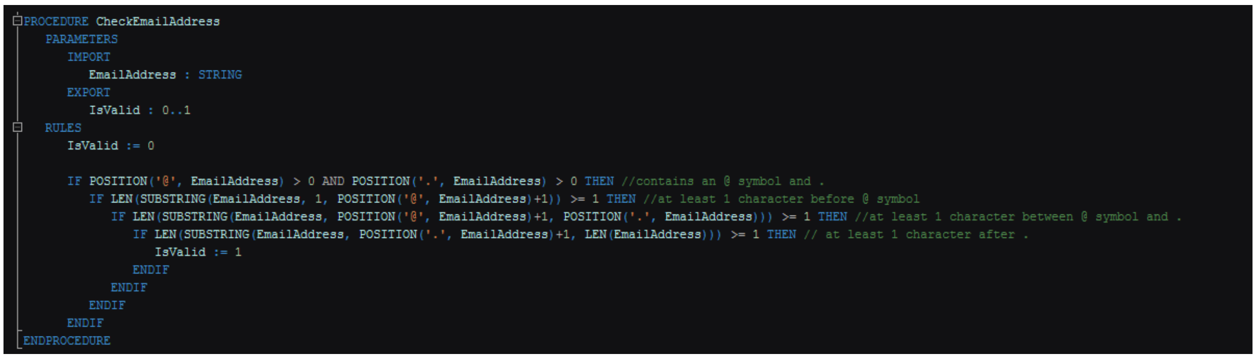
> back to Table of Contents

7. Workarounds to Correct Compliance Issues

7.1 Constraint Errors

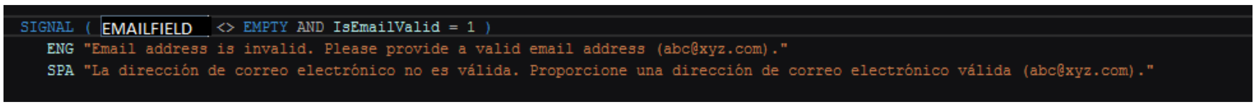
While there were many issues that we could not find a way to circumvent, we did find workarounds for some of the accessibility issues. One of the issues discovered by our accessibility testers was that a respondent could not move backwards after receiving a Blaise 5 constraint error. We were using constraint errors to prevent respondents from entering invalid email addresses (as suggested by the Blaise 5 Help). To get around this issue, we developed a procedure to check email addresses. This procedure exports an "IsValid" value that evaluates to '1' when the email is valid. By utilizing a signal and this procedure, we were able to allow the respondent to move backwards in the survey, even if they had entered an invalid email address (they still could not move forward).

Figure 7a. The CheckEmailProcedure Evaluates the Entered Email Address to Check Its Validity and Will Return a 1 (valid) or 0 (invalid)



```
PROCEDURE CheckEmailAddress
   PARAMETERS
      IMPORT
         EmailAddress : STRING
      EXPORT
         IsValid : 0..1
   RULES
      IsValid := 0

      IF POSITION('@', EmailAddress) > 0 AND POSITION('.', EmailAddress) > 0 THEN //contains an @ symbol and .
         IF LEN(SUBSTRING(EmailAddress, 1, POSITION('@', EmailAddress)+1)) >= 1 THEN //at least 1 character before @ symbol
            IF LEN(SUBSTRING(EmailAddress, POSITION('@', EmailAddress)+1, POSITION('.', EmailAddress))) >= 1 THEN //at least 1 character between @ symbol and .
               IF LEN(SUBSTRING(EmailAddress, POSITION('.', EmailAddress)+1, LEN(EmailAddress))) >= 1 THEN // at least 1 character after .
                  IsValid := 1
               ENDIF
            ENDIF
         ENDIF
      ENDIF
ENDPROCEDURE
```

Figure 7b. This Signal Will Prevent the Respondent from Moving Forward If the Email Address Is Invalid, but Still Allows the Respondent to Go Backwards in the Survey



```
SIGNAL ( EMAILFIELD  <> EMPTY AND IsEmailValid = 1 )
   ENG "Email address is invalid. Please provide a valid email address (abc@xyz.com)."
   SPA "La dirección de correo electrónico no es válida. Proporcione una dirección de correo electrónico válida (abc@xyz.com)."
```

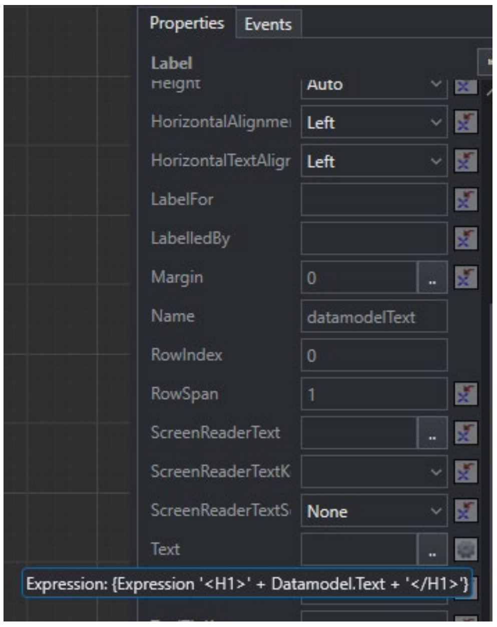
7.2 Page Title Is Not Programmatically Identified as a Heading

Another issue identified was that the page title, which was the data model name, was not identified as a heading. This lack of identification conflicted with the requirements of WCAG 2.0 SC 1.3.1, "Info and Relationships" that requires information and relationships to be able to be programmatically determined.

Adding the <H1> tag to the data model name in the Blaise 5 source code would have fixed this issue, but then it also would have been applied in both layout sets, large and small. Adding the <H1> tag overrides any other font settings and modifying it in the Blaise source would make the data model name H1-sized text in the small layout set, as well. To avoid making the data model name this large in the small layout set, we adjusted the page header template part used only in the large layout set. The small layout set, which we did *not* evaluate for 508 compliance, retained the small text determined to look best in the small layout set.

> back to Table of Contents

Figure 7c. Adding <H1> Tags on Text for the Label to Display the Data Model Name Corrected This Issue While Allowing the Small Layout Set to Retain the Smaller Font Size; This Change Was Made on the Template Part Used for the Header in the Large Layout Set



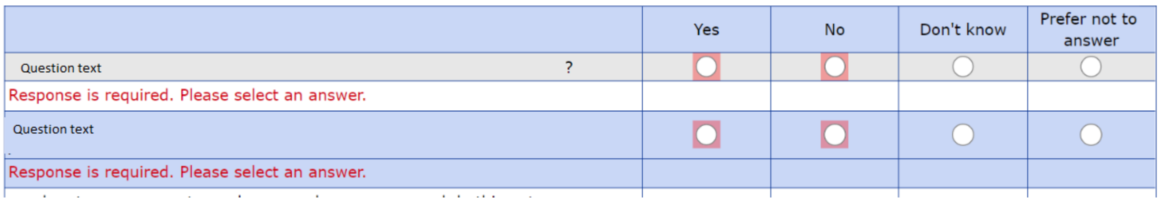
Height	Auto
HorizontalAlignme	Left
HorizontalTextAligr	Left
LabelFor	
LabelledBy	
Margin	0
Name	datamodelText
RowIndex	0
RowSpan	1
ScreenReaderText	
ScreenReaderTextK	
ScreenReaderTextS	None
Text	

Expression: {Expression '<H1>' + Datamodel.Text + '</H1>'}

7.3 Tables

Errors in tables presented another accessibility challenge. WCAG 2.0 SC 3.3.1 requires errors to be clearly identified. Part of that requirement is that the error message reference the associated field. Blaise 5 tables initially did not fulfill this requirement, but we were able to modify the templates to display information that would make these tables 508-compliant. This solution was not implemented in the final production survey, however, because it was decided displaying field names (as required for accessibility compliance) might prove confusing for most respondents.

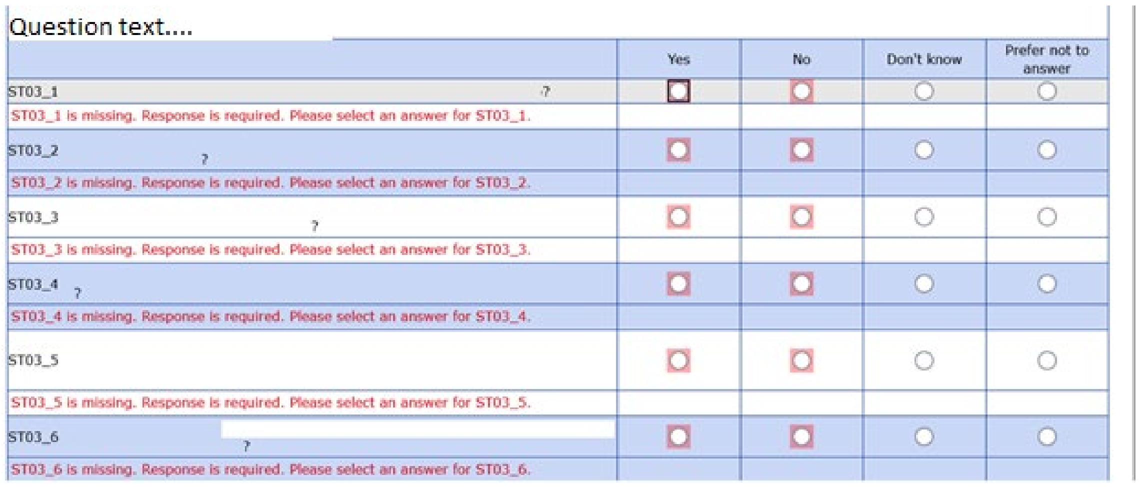
Figure 7d. Default Templates Do Not Make a Reference to the Field Name, Either Where the Question Is Displayed or in the Error Message When It Displays



	Yes	No	Don't know	Prefer not to answer
Question text ?	○	○	○	○
Response is required. Please select an answer.				
Question text	○	○	○	○
Response is required. Please select an answer.				

> back to Table of Contents

Figure 7e. Adjusted Templates Display the Field Name with the Question Text and in the Error Message When It Is Displayed so the Respondent Can Easily Tell Which Error Applies to Which Row in the Table



Question text....

	Yes	No	Don't know	Prefer not to answer
ST03_1 ?	○	○	○	○
ST03_1 is missing. Response is required. Please select an answer for ST03_1.				
ST03_2 ?	○	○	○	○
ST03_2 is missing. Response is required. Please select an answer for ST03_2.				
ST03_3 ?	○	○	○	○
ST03_3 is missing. Response is required. Please select an answer for ST03_3.				
ST03_4 ?	○	○	○	○
ST03_4 is missing. Response is required. Please select an answer for ST03_4.				
ST03_5	○	○	○	○
ST03_5 is missing. Response is required. Please select an answer for ST03_5.				
ST03_6 ?	○	○	○	○
ST03_6 is missing. Response is required. Please select an answer for ST03_6.				

7.4 Magnification to 200%

When size (pixels, in this case) is used to determine the layout set, Blaise 5 reevaluates the layout set used when the respondent magnifies the browser window. This feature caused us issues with WCAG 2.0 SC 1.4.4, which requires content to be able to scale up to 200% without changes to the content on screen. Our testing team determined that in the process of scaling the content up to 200%, the page would update from using the large layout set to the small layout set. To work around this issue, we adjusted the condition used to determine when to switch from the small layout set to the large one. Instead of using 800 px as the cutoff for the switch from the large layout set to the small one, we used 600 px. This cutoff point kept the layout set used from updating when scaling up from 100% to 200%.

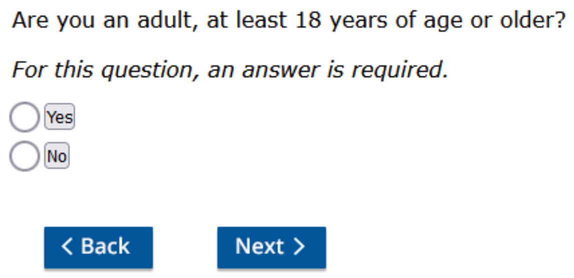
8. Communication with the Blaise Development Team

As always, the Blaise development team was very responsive to our accessibility questions and did their best to help us find ways to circumvent the issues identified. Some of the issues we reported were identified as bugs and corrected in a subsequent build, but due to time constraints for testing, we determined it would not be worth the risk to switch major versions so close to the launch date for the instrument. Other issues were corrected quickly enough that we could utilize the newer build prior to production. Initially, we experienced a problem where the tab order would jump to an incorrect spot after selecting a response from a dropdown. The Blaise development team determined this issue was a bug and it was promptly fixed in the subsequent build, 5.12.8, which we used in production for this instrument.

A few of the issues we reported were “fixable” by turning on some of the Blaise 5 accessibility options in the BLAX “Settings” tab, but these options caused other issues. For an issue where the error message was not receiving focus correctly, it was suggested to turn on the “Use CSS grids instead of the size tree” option on the settings tab of the BLAX to correct it. However, turning on this option caused the response options to display as buttons.

> back to Table of Contents

Figure 8a. Turning on the "Use CSS Grids Instead of the Size Tree" Option Caused Response Options to Appear as Buttons


Are you an adult, at least 18 years of age or older?

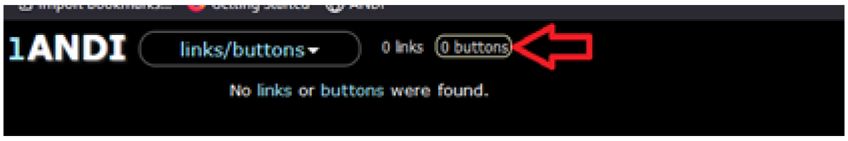
For this question, an answer is required.

Other issues were identified by ANDI as problems but worked correctly in NVDA. The issue with radio buttons identifying as "0 of undefined" is an example of a problem that showed in ANDI but worked correctly in NVDA. The Blaise development team said that NVDA read out (correctly) "4 of 4" for a field with four radio buttons, while ANDI still saw it as "0 of undefined."

9. Challenges in 508-Compliance Testing

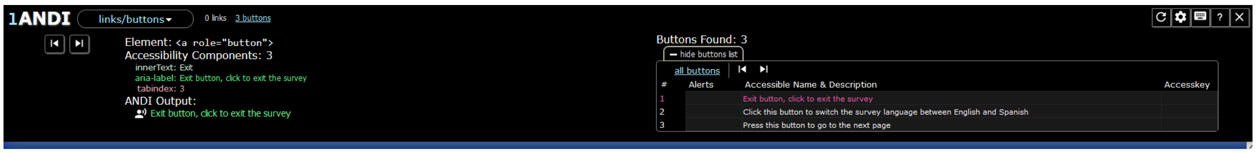
In addition to trying to navigate the sometimes complex 508-compliance requirements, we also had to contend with bugs found in the ANDI software while testing. One of the compliance issues discovered by our compliance testing team was "WCAG 2.0–2.4.4 Link Purpose: The purpose of each link or button can be determined from any combination of the link/button text." We discovered that ANDI had a bug: it wouldn't detect the Blaise 5 buttons when the screen was initially loaded. Instead, it required clicking on the "0 buttons" link on the page.

Figure 9a. Upon Initially Loading the Page Where the 508-Compliance Error Was Identified, ANDI Showed "0 buttons"; This Link Is Identified in the Image by the Large Red Arrow



No links or buttons were found.

Figure 9b. After Clicking on the Previously Mentioned "0 buttons," ANDI Correctly Identified All Blaise 5 Buttons on the Page as Buttons



Element: <a role="button">
Accessibility Components: 3
innerText: Exit
aria-label: Exit button, click to exit the survey
tabindex: 3
ANDI Output:
Exit button, click to exit the survey

Buttons Found: 3

#	Alerts	Accessible Name & Description	Accesskey
1		Exit button, click to exit the survey	
2		Click this button to switch the survey language between English and Spanish	
3		Press this button to go to the next page	

As mentioned previously, sometimes we were able to achieve better compliance by turning *off* certain Blaise 5 "accessibility" features. While it seems odd that *not* using features specifically meant to facilitate accessibility improved our accessibility testing "score," we believe this apparent conflict is the result of different methods of 508-compliance testing. In our communications with the Blaise development team (discussed in more detail in Section 8), it was discovered that they primarily relied on testing using the NVDA screen reader. This feature that caused issues for ANDI testing likely fulfilled some requirement

> back to Table of Contents

of the NVDA tool. We highlight this difference in method here to highlight the difficulty in achieving total 508 compliance when different tools or methods are used to determine compliance.

To further emphasize this point, when the client conducted their own 508-compliance testing on the finished survey, they found an issue not previously identified during ANDI testing. Not all WCAG 2.0 requirements can be measured using the ANDI tool, so individual tester evaluation and analysis can add yet another layer of variability to compliance testing. 508-compliance testing results will vary as much as the tools, methods, and testers used to conduct it.

10. Final Results and Future Plans

We were unable to achieve full 508 compliance using Blaise 5.12.8. Fortunately, we were able to resolve all compliance issues identified as “severe” by the client and only had a small number of more minor issues left unresolved. Bugs identified during testing for this instrument should ideally be corrected in future versions of Blaise 5, where possible. The testing process also taught us several valuable lessons that helped us get a few steps closer to 508 compliance, and we hope that future versions of Blaise 5 will allow us to get even closer.

11. References

ANDI (Accessibility Testing Tool), Accessible Solutions Branch of the Social Security Administration
https://www.ssa.gov/accessibility/andi/help/install.html

Blaise 5 Help Reference Manual, Statistics Netherlands
http://help.blaise.com/

WCAG 2.1 Understanding Docs, W3C Web Accessibility Initiative (WAI)
https://www.w3.org/WAI/WCAG21/Understanding/

> back to Table of Contents