IBUC 2020 LIMASSOL

Conference Proceedings

19th International Blaise User Conference

14–17 September 2020
Virtual Conference

Preface

This document contains the papers presented at the 19th International Blaise Users Conference. Due to the COVID-19 pandemic, the conference could not be held in Limassol, Cyprus. Instead, a virtual IBUC was held from September 14 – 17, 2020. The online conference included four days of presentations on the use of Blaise and related topics as well as Blaise training sessions.

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

- Jane Shepherd (Westat, USA)
- Leif Bochis Madsen (Statistics Denmark)
- Mark M. Pierzchala (MMP Survey Services, LLC, USA)
- Mike Hart (Office for National Statistics, UK)
- Tim Carati (Statistics Netherlands)
- Rob Wallace - US Census Bureau

The Statistical Service of Cyprus (CYSTAT) was the original host organization for the IBUC 2020. The online conference was hosted by Statistics Netherlands (CBS). Both organizations worked together to create this proceedings book for the benefit of the conference participants and others.

Table of Contents

The Co-existence of Blaise 4 and Blaise 5 in CYSTAT

Charalambos Charalambous, Charoula Charalambous, Niki Chrysostomou, Costas Diamantides and Konstantinos Mina, Statistical Service of Cyprus

1. Abstract

CYSTAT has a long history in the use of Blaise. The first electronic questionnaires were developed in 1995 and since then, Blaise has become the main tool for data capture. During the first years of the use of Blaise in CYSTAT, several employees from different production units participated in the trainings offered by Statistics Netherlands. In this way, there were no centralized procedures for development and consequently the sharing of knowledge was lagging. In 2011 it was decided to establish the Blaise team of CYSTAT with the task for the development of all the applications with Blaise. Nowadays, the Blaise team of CYSTAT is part of the Methodology Section and comprises of one developer.

Despite the limited human resources several applications are being developed and supported by the Blaise team. Blaise 5 was recently introduced in CYSTAT and it is now used in two surveys, the ICT Usage in Enterprises and e-commerce (ICT-ENT) and in Employment and Job Vacancies. The data collection in both surveys is multi-mode and the data collection tools are developed with Blaise 4 and Blaise 5. Another example of a major survey incorporating multi-mode data collection is the Income and Living Conditions (EU-SILC). In this case, however, all the data collection tools are developed in Blaise 4 and considering all its advantages the aim is to convert to Blaise 5. The EU-SILC is the only survey that is not being developed by the Blaise team.

The aim of the paper is to present the experiences of CYSTAT in multi-mode data collection by using both versions 4 and 5 and to share the future plans in using Blaise 5.

2. Background

This paper examines the use of Blaise in three surveys, the ICT-ENT, the Employment and Job Vacancies and the EU-SILC. This section provides all the background information for all three surveys.

2.1 ICT Usage in Enterprises and e-commerce (ICT-ENT)

The ICT-ENT is being carried out annually since 2004 by applying the method of Computer Assisted Personal Interviewing (CAPI) implemented in Blaise. The aim of the survey is to collect data about the Information and Communication Technologies (ICT) used by the Enterprises. The major topics covered are: use of computers, access and use of the Internet, employment of ICT specialists and their skills, ICT security and E-commerce. The data collected through the survey are necessary for the implementation of policy programmes at both the European and National level.

The responsibility for carrying out all the steps of the survey – from the design to the dissemination of the results - lies on the ICT Section of CYSTAT with the support of the Blaise team. The methodological manual, the model questionnaire as well as the validation and transmission tools are all provided by Eurostat.

The data collection takes place during January - May and covers approximately 2.000 enterprises with 10 or more employees in almost all categories of economic activities. The method used for the sample selection is stratified random sampling. Two variables are used for the stratification, the economic activity (16 groups) and size (3 size groups: Small enterprises (10-49 employees), Medium enterprises (50-249

employees) and Large enterprises (250+ employees)). The survey covers all the government controlled areas of the Republic of Cyprus and the reference period of the survey is the current year unless otherwise stated.

Since 2017, the data collection is being carried out either through a web questionnaire (CAWI - Blaise 5) or personal interviewing with the IT manager of the enterprise (CAPI - Blaise 4.8). The web questionnaire is available during the first phase of the survey period (first 4 - 5 weeks). If an enterprise fails to fill in the web questionnaire in the first phase then it is obliged to provide the data through the personal interview in the second phase. The duration of the second phase is up to 2 months. It is important to note that during the second phase the web questionnaire remains to be available online.

As soon as the data collection is completed, all the datasets (from both CAWI and CAPI) are joined by using Manipula and exported in ASCII format. Then the complete dataset is imported in SPSS for editing. As soon as the file with the clean data is ready all the necessary output is produced.

2.2 Employment and Job Vacancies

The Employment and Job Vacancies survey is carried out on a quarterly basis. All the enterprises with 20 and more employees are covered and for those with 1-19 employees a representative sample is used. The survey covers the vast majority of economic activity categories.

The sampling frame used for the sample selection is drawn from the Business Register which contains all enterprises as well as their local units that carry out any economic activity irrespective of their size. A stratified sampling technique is used as sampling method. The strata are defined by the cross-classification of economic activity groups with size classes (enterprise with no more than 1 employee, enterprises with 1,5 - 19,5 employees and enterprises with more than 19,5 employees). All enterprises employing more than 19,5 employees are selected for the survey whereas for the other size categories, the sample within each stratum is selected using simple random sample. The sample size in categories with no more than 19,5 employees is based on the number of enterprises in each category in the population. For the case of public sector and the publicly owned enterprises, those are fully covered. The same sample is used every quarter and the sample size is about 3300 enterprises.

The survey collects the monthly employment levels for each cell of a matrix determined by: 1) full-time/ part-time employment and 2) status of employment (working proprietors, unpaid family workers, employees). It is noted that working proprietors are included also in cases where they are self-employed and the only labour force of their enterprise. Part-time employment is defined as those persons working less than 30 hours in the reference week. In addition, the questionnaire collects the number of hours worked by part-time workers as well as an indication of the normal hours worked by a person in full-time employment. It also collects information on Job Vacancies like the number and the description of the position.

The data collection lasts for 3 weeks and it is carried out either by Computer Assisted Telephone Interviewing (CATI Module – Blaise 4) or through a web questionnaire (CAWI – Blaise 5). Both options run concurrently and there is a mechanism put in place which ensures the unique collection of data from each enterprise. As soon as the data collection is completed, all the datasets (from both CATI and CAWI) are joined by using Manipula script and exported in ASCII format. Then the complete dataset is imported in SPSS for editing. As soon as the file with the clean data is ready all the necessary output is produced.

2.3 Income and Living Conditions (EU-SILC)

The Income and Living Conditions (EU-SILC) is an annual cross-sectional and longitudinal household sample survey, based on European Commission Regulation and is the main source for data and indicators on income, poverty, social exclusion and living conditions in the European Union. Cross-sectional data pertain to fixed time periods, with variables on income, poverty, social exclusion and living conditions, while longitudinal data pertain to changes over time, usually observed over four years. The survey was first launched by CYSTAT in 2005.

The questionnaire of the survey is designed every year according to Eurostat's guidelines. The guidelines concern the methodology including the definitions of all the variables (both primary and secondary).

The sample is based on a rotational design of 4 replications with a rotation of one replication per year. This design is appropriate to serve the cross-sectional and longitudinal aspects of the survey. Each year the sub-sample that is in the survey for 4 years is dropped and a new sub-sample is selected by one-stage stratified simple random sampling. The sample selection is conducted by the Methodological Unit of CYSTAT. The total sample size (all 4 sub-samples) is around 5.000 households and complies with the effective sample size requirements for the cross-sectional and longitudinal components set by the Commission regulation. The measurement units are the households and their members.

The main mode of data collection is CAPI using BLAISE 4. From 2013 onwards, CYSTAT combines both CAPI and telephone interviewing. Telephone interviewing is used only for a sub-sample of the survey, that is Nicosia's households which are in the survey for 3 or 4 years (near the 20% of total sample). The data capture during the telephone interviewing is carried out on the same electronic questionnaires as in CAPI and thus, the CATI module is not being used.

An addition to the data collection process, from 2013 onwards, was the access to administrative income data from various Government departments such as the Grants and Benefits Service, the Department of Social Insurance Services, as well as the Treasury of the Republic. The use of administrative records has proven to be very effective. It serves two purposes: accuracy and reduction of the burden on the respondent. By using administrative records, it was possible to check the quality of the data collected by the interviewers and also to record income data in the cases where this was not provided by the respondents.

As soon as the data collection is completed, all the datasets (from both CAPI and telephone interviewing) are joined by using Manipula script and exported in ASCII format. Then, the complete dataset is imported in SAS for editing. As soon as the file with the clean data is ready, all the necessary output is produced.

3. Use of Blaise in CYSTAT

This section contains descriptions of the processes that are applied in the use of Blaise in the three surveys with the aim first to present how Blaise 4 and 5 coexist for the purposes of the same survey and second to identify the problems encountered.

3.1 CAPI - CAWI

As described in 2.1 the ICT-ENT survey has been carried out annually by CYSTAT since 2004. The data collection had always been carried out by applying the method of CAPI implemented in Blaise. In 2015, a web questionnaire was developed and pilot tested for the first time in Blaise 5.

Although a member of our staff participated in a Blaise 5 training course organised by the Blaise Team of Statistics Netherlands the knowledge gained in Blaise 5 was insufficient for the implementation of a web questionnaire. In the framework of the long standing cooperation of CYSTAT and Statistics Netherlands the Blaise Team provided assistance to implement the first web questionnaire according to CYSTAT's requirements. Moreover the Blaise Team provided guidelines on how to make all the installations on the web server.

As a result from the transfer of knowledge, from 2016 onwards the web questionnaires are developed by CYSTAT. In 2016, a second but in a larger scale pilot was carried out. Since 2017, the option for completing the web questionnaire is being offered to all enterprises which are included in the sample. The web questionnaire option is available to all enterprises during the first phase of the data collection survey. At the second stage, for those enterprises that do not fill in the online questionnaire a personal interview is carried out and the questionnaires developed in Blaise are filled in on netbooks/ laptops. When the second phase of data collection begins the respondents still have the option to fill in the web questionnaire. So at this phase both CAWI (Blaise 5) and CAPI (Blaise 4.8) are running.

In the 2019 survey, the web questionnaire was developed in Blaise 5.6.1. No problems were encountered in the transition from previous versions. There were some difficulties regarding the compatibility of the web layout after the upgrade that have been resolved. The CAPI version was developed in Blaise 4.8.

Until the full implementation of the survey in Blaise 5 there are procedures put in place for the co-existence of Blaise 4 and 5. In particular, the responsible Statistics Officer of the survey has the daily task to export all the questionnaires that were submitted (or started) through the web. Then these data are converted by running a Manipula script and imported in the Blaise database built in Blaise 4.8 (the same is used in CAPI). In order to avoid double counting the Statistics Officer removes the questionnaires completed online from the workload of the enumerators. The editing is carried out in all the questionnaires included in the database which is built on Blaise 4.8. The editing stage includes the import of data for those surveys that were also included in the samples from previous years. Figure 1**: Processes in CAWI** displays all the major processes displays all the major processes concerning the web questionnaire and Figure 2 the processes to combine CAWI and CAPI.

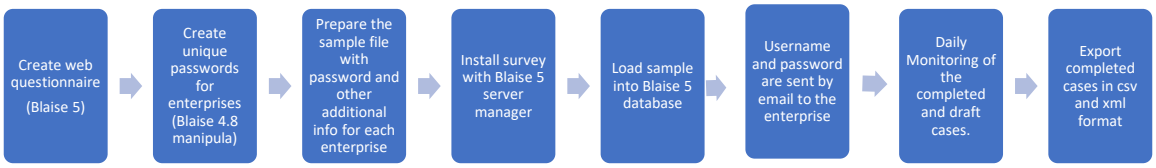


Figure 1: Processes in CAWI

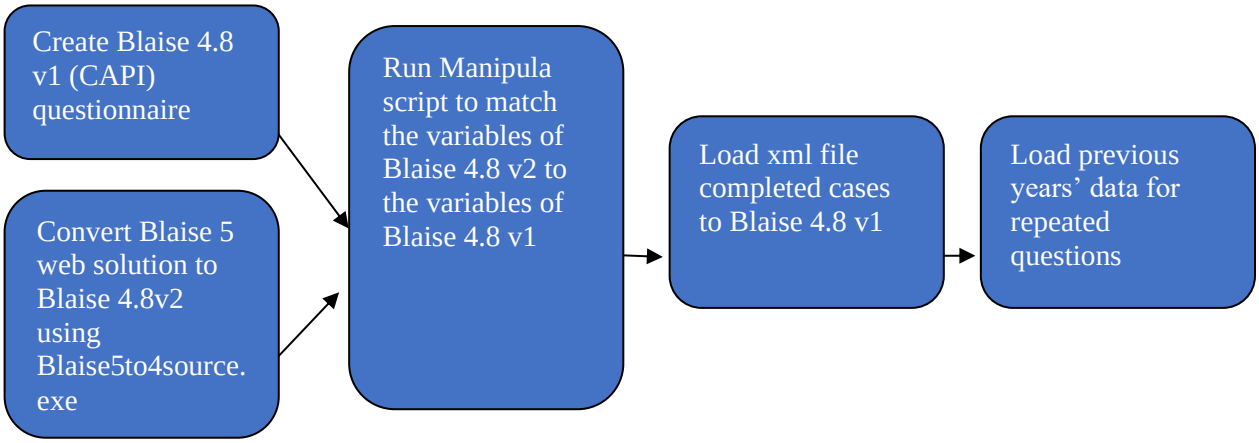


Figure 2: Processes to combine CAWI and CAPI

The process described above for converting from Blaise 5 to 4 is time consuming and prone to errors. This period is considered, however, as transitional to the full implementation of Blaise 5 and the planned steps are described in Chapter 4.

3.2 CATI - CAWI

Employment and Job Vacancies is a quarterly survey in which, traditionally, the data has been collected through CATI. Since the third quarter of 2016 the Blaise 4.8 CATI module has been utilised for data collection and management of the survey.

In the framework of the implementation of CYSTAT's strategy to offer respondents alternative means to provide the required information, the CAWI option built on Blaise 5.2.5 was offered for the first time in the second quarter of 2018. This web questionnaire was developed by CYSTAT based on the knowledge gained from the ICT_ENT implementation.

The Blaise CATI module offers the opportunity to the operator to arrange an appointment with the enterprise in order to fill in the questionnaire at a later stage, to complete the questionnaire for any other reason (e.g. closed enterprise, merged with other enterprise etc.). The questionnaires that are completed and there is no need for any further treatment are considered as "Completed" and are not active in the day batch. Touched questionnaires that are not completed are considered as "Pending" and appear in the day batch file as they are routed. All completed cases collected through web are imported every morning before the creation of day batch file in CATI Blaise program and are routed back at the end of the day to the interviewers to check the questionnaire and proceed with coding**.**

Figure 3 describe the processes that are related to the CATI implementation. It is noted that the processes to join CATI and CAWI are the same as described in Figure 2 with the exception that instead of CAPI in this case we have CATI. Furthermore, there is no need for the last step (load previous years' data for repeated questions) in CATI since this step is already done at the phase of the creation of the Blaise 4.8 CATI questionnaire.

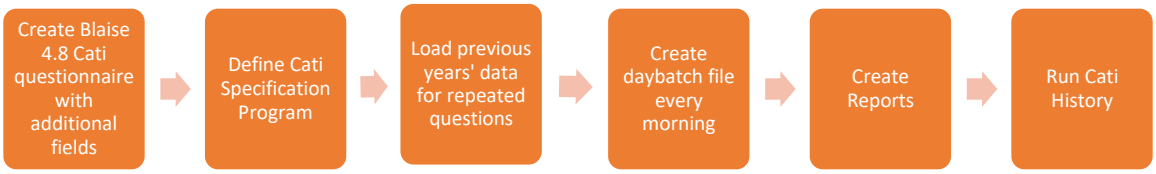


Figure 3: Processes in CAWI

Similar to the case of CAPI and CAWI it is envisaged to have the CATI module in Blaise 5 in order to be able to work in a homogeneous environment and thus, avoiding having the time consuming conversion and updating procedures.

3.3 CAPI

The main mode of data collection in the EU-SILC survey is CAPI using BLAISE 4. From 2013 onwards, CYSTAT combines both CAPI and telephone interviews. The telephone interviews are not implemented with the Blaise CATI module. Instead the operator type in the responses during the telephone interview to the questionnaire developed in Blaise 4. Telephone interview are used only for a sub-sample of the survey, that is Nicosia's households which are in the survey for 3 or 4 years (near the 20% of total sample).

The75% of the sample are HH that are in the survey for the 2nd, 3rd or 4th time; hence a lot of information from the previous year's survey is preloaded. Some of this information is visible during the interview and some is accessible only during the coding and editing by the supervisors. For that reason, 2 ASCII files are exported from the previous year's survey and they are modified and merged with the new sub-sample location information in order to import them to the new year's electronic questionnaire.

Implementing BLAISE provides the researcher with a powerful tool for applying validation and consistency checks on the field and it also helps enormously in the correct routing of the long and complicated questionnaire of the survey. Additionally, by collecting data with BLAISE, the process of after the data collection data entry is abandoned, making the procedures more cost effective.

Every week during fieldwork, at supervisor – interviewer meeting, the completed questionnaires are transferred to the supervisor for coding and editing. During this phase, the supervisors are calling back to the households for clarifications and any inconsistencies found. Moreover, some reports are generated using Blaise to monitor the data collection, i.e. the number of completed questionnaires, the number of not located households and the response rate. The IDs needed for extracting the information from the registers are also exported.

At the end of the editing, all the supervisors' databases are merged and form the complete database for the year. Then, 11 ASCII files are created and imported to MS-Excel for cleaning by the Statistics Officers. Finally, SAS is used for the final data checks, the calculations of the weights and the creation of the 4 data files that are transmitted to EUROSTAT.

The EU-SILC processes in CYSTAT are summarized in Figures 4a and 4b.

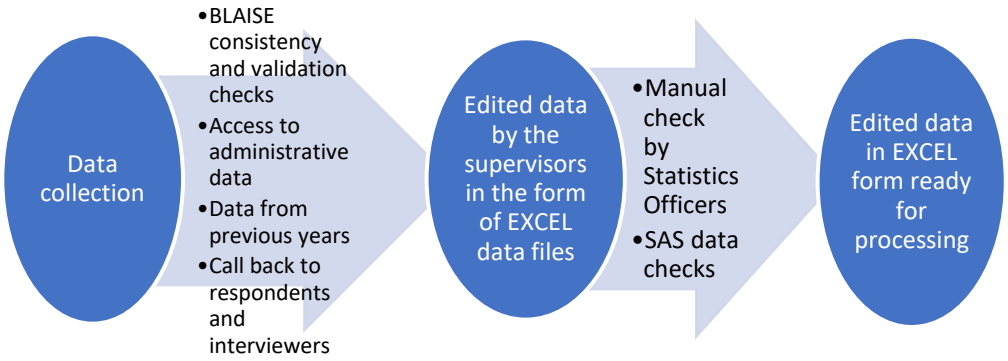


Figure 4a: Processes in EU-SILC

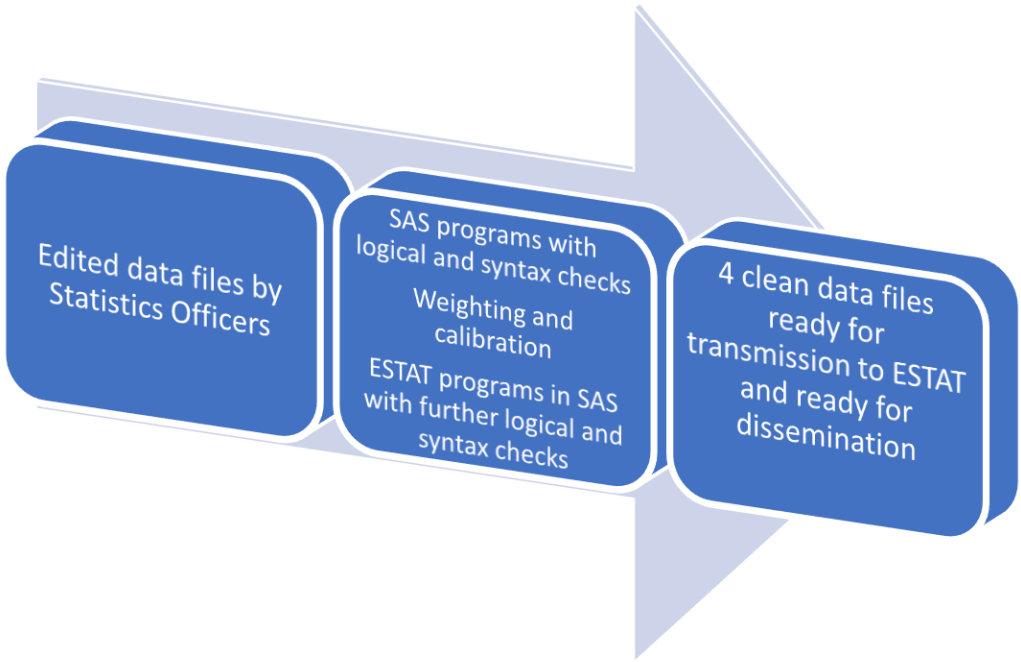


Figure 4b: Processes in EU-SILC

4. Future Plans

As already mentioned above, CYSTAT is a small office with limited human resources and in particular IT resources. The progress achieved during the years in the use of Blaise is mainly due to the significant support provided by the Blaise Team of Statistics Netherlands. It is envisaged that this excellent cooperation will continue and in the forthcoming years. Having as asset this excellent cooperation, CYSTAT plans to gradually transit to Blaise 5. Due to limited human recourses, however, there is no specific timeframe for this transition. It is imperative that the size of CYSTAT's Blaise Team needs to increase in order to tackle all the challenges that are included in the transition.

One of CYSTAT's goals is to implement the ICT-ENT survey as multimode in CAPI and CAWI with Blaise 5. In this respect, in the 2020 survey one tablet has been given to an enumerator and he is given the task to carry out the CAPI by filling in the questionnaire developed in Blaise 5.6.4 Android App. The completed questionnaires are uploaded through the internet to the server. This process at the moment of writing this paper is still running and therefore, no feedback can be provided at this stage. However, from the experience gained so far there were some problems with the design of the layout, but those were tackled with the assistance of the Blaise Team of Statistics Netherlands.

From the relative short term experience in CAWI it can be concluded that most of the respondents prefer to complete a web interview using mobile device. However, this cannot be done due to the small size of text and buttons which cannot be resized. Question text, navigation buttons, and code frames are sometimes clipped if there is too much information on the screen. In such situations navigation is impossible because the buttons are unavailable. Moreover, the scrolls up and down buttons do not work. In this respect, the web screens should be optimized for the mobile too.

As regards the survey on Employment and Job Vacancies, the next step is to successfully convert Blaise 4 CATI instrumentation into Blaise 5. The Blaise 5 instrument should be executed in CATI, CAWI modes. In this respect, as soon as this conversion is achieved CYSTAT envisage to use Blaise 5 CATI in other surveys as well, for instance the EU-SILC. The long term goal is to set up a telephone centre in CYSTAT based on Blaise 5 for all the surveys that are carried out by telephone.

As far as the EU-SILC survey is concerned, in the 2021 the new IESS regulation will be implemented. The most difficult aspect of this regulation, for most of the countries, is the timeliness of the data transmission. The data must be transmitted at least 9 months earlier than the existing regulation. In order to comply with the IESS, a lot of the procedures must be reviewed and modernized. First, for the data collection it is necessary to upgrade from Blaise 4 to Blaise 5, especially as it supports multimode data collection. Additionally the field work needs to be shortened. It is essential to examine how to improve and further develop existing procedures in order to use administrative data more effectively. For example, questions regarding benefits, the salaries of the government employees and pensions from the Social Insurance Services and the Government should not be asked during interviewing when the Civil ID of the person is provided. This information should be uploaded automatically from the respective registers. Moreover, completed questionnaires from the interviewers should be transferred to a server and not locally, as it now, for better monitoring of both the data collection and the editing from the supervisors. This would help to decrease the length of the editing phase. In addition to that, the reviewing and modernization of the microdata editing and processing in Excel and SAS is necessary.

In 2021, the census of population will take place. Following the success of the previous census in 2011 where the data collection was carried out with netbooks and the information system was built on Blaise 4.8, it has been decided to carry out the census with tablets and Blaise 5. We expect that the experience to be gained in the whole process will be very beneficial in the implementation of the strategy to use Blaise in all surveys carried out by CYSTAT.

According to CYSTAT's strategy Blaise is the standard for data capture, CAPI, CATI, CAWI and CADI. Moreover, the data collection in several surveys is multi-mode. Considering that Blaise 5 offers the infrastructure to build the necessary tools for multi-mode data collection and coupling this with CYSTAT strategic aim, the full transition to Blaise 5 has no return.

5. Conclusions

The transition from Blaise 4 to Blaise 5 is a long process which is further hampered by the lack of human resources. Considering, however, the benefits of Blaise 5 that mainly concern the support for multi-mode data collection, and relying on the support of the Blaise Team of Statistics Netherlands there is optimism that the strategic aim for the full conversion to Blaise 5 will be achieved.

The use of CAWI is expected to offer significant savings in the cost for carrying out the surveys. Indicatively, from the recent experience in applying CAWI in the ICT-ENT survey, 50% of the questionnaires were filled in online. The direct data entry by enterprises minimizes interviewers' contribution and the working hours required by the responsible officer.

Another advantage is automation and real-time access. Respondents import their own data and these are automatically stored electronically on the server. At any time it is easy to have access to the data. Daily reports are also available.

Finally, with the use of Blaise 5 it is envisaged improving the layout of the survey due to the design flexibility and respondents can provide the information at their convenience.

Converting Social Survey Blaise 4 questionnaires to Blaise 5; The Challenges of Multi-lingual Questionnaires and Challenging Collection Environments

Andy Watson and Steve Maurice, Office for National Statistics United Kingdom

1. Abstract

The Office for National Statistics ONS) Social Surveys Division (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. While collecting data via Blaise 4 ONS have used case management software 'Casebook', developed and maintained in-house. Now that we're moving to Blaise 5, we're looking to retire Casebook and rely more heavily on "out of the box" features of Blaise 5.

In this paper we report on our experiences moving the National Survey for Wales (NSW) and the International Passenger Survey (IPS) from Blaise 4 to Blaise 5.

1.1. The National Survey for Wales

The NSW survey is a study run by Office for National Statistics on behalf of the Welsh Government. The study gathers information on many topics including health, schools, sports, arts and culture. The statistics are used by Welsh Government, Sports Wales, Arts Council Wales, Natural Resources Wales, Welsh local authorities, charities and academics. They are used to inform progress against key indicators of the Future Generations Act, to inform progress on existing or the shaping of new policies within Wales.

There is now a follow up survey, the Welsh Language Usage Survey (WLUS) in the form of a paper questionnaire distributed by interviewers to eligible respondents at the end of NSW interview. This questionnaire is about the Welsh language. It is to establish if a respondent, who has indicated they may be able to speak Welsh in the main questionnaire, can speak Welsh, and if so, how often, where and when they use it.

1.2. The IPS

The IPS is a continuous survey that has been conducted by ONS since 1961. Between 700,000 and 800,000 interviews are conducted every year at major airports, seaports and tunnel routes covering both departures and arrivals terminals. This is the main source of information used by the UK governing bodies for planning, monitoring and informing decisions on tourism and immigration policies.
Having successfully moved IPS data collection from paper to tablet[1] using Blaise 4 as the collection tool[2], the Office for National Statistics (ONS) are now ready to move collection from Blaise 4 to Blaise 5 so collection can continue as Blaise 4 is phased out.

2. Behind the surveys

2.1. NSW

The NSW was started in 2016 and has covered a variety of subjects. The questionnaire object is currently written in Blaise 4.8 and collected face-to-face. The Welsh Government stipulated that the questionnaire be available in both English and Welsh for all respondents. A specification was provided by Welsh government, in English with all questions and routing specified. The questionnaire utilised some of the standardised 'Core blocks' from other Social surveys already created. When this completed single language version of the questionnaire was finalised, a separate specification was then provided by Welsh Government with all questions and routing translated into Welsh.

Table 1: NSW Selected Language of interview

Language	Frequency	Percent
English	11,756	*98.6*
Welsh	159	*1.3*
Other	7	*0.1*
Total	11,922	*100.0*

2.1.1. NSW Methodology

The annual sample size for the NSW is 12,000 face-to-face interviews with a required response rate of over 56%. This sample is split into 12 field periods over the year. Each year Welsh government request different subject modules for the questionnaire, depending on their requirements for informing decisions to be made on new policies. Due to the number of modules created each year, the length of the survey can differ greatly. To keep a level of consistency, some modules are 'Sub-Sampled' meaning that certain modules only get asked to a randomly selected percentage of the main sample.

2.1.2. NSW Collection

Data collection for the NSW is done face-to-face by field force interviewers in the homes of the respondents. Advanced letters are sent to all sampled respondents and 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.

2.2. IPS

2.2.1. IPS Methodology

The IPS is conducted at various Airports, Seaports, and at the Eurostar/Eurotunnel. The survey is not pre-sampled as it's impossible to know who'll pass through the ports in advance. Each location has a different sampling interval for each direction of travel, e.g. At Gatwick it might be that every 20th person to pass is selected and every 40th for departures. A new form is created for each attempt at an interview.

The questionnaire takes two different routes according to subsampling. One route being the full questionnaire with all questions asked (full form), the other being a much shorter route with the potential to turn into a full questionnaire if certain criteria are met (filter form). As with the sample intervals, each port has a different sub sampling for both arrivals and departures e.g. At Gatwick arrival shifts might be 2 out of 5 presented with the full form, while the other 3 are presented with the filter form. In a shift with both full and filter forms, research were keen for the order these forms are presented to be random, e.g. for a 2 out 5 shift, the first form of every group of 5 forms has an equal chance to be full or filter.

All sample intervals and sub samples are decided ahead of collection by the research team and stored in a Blaise database to be used by the questionnaire as a lookup.

2.2.2. IPS Collection

As IPS interviewers are standing in busy ports and need to be mobile to approach those sampled, it was decided that interviewers would collect using tablets rather than laptops. Tablets are far lighter and more suited to longer shifts so the questionnaire had to be designed to work well on a Tablet[1][2]

IPS data are collected offline due to a lack of available connectivity at the ports our interviewers operate in. In Blaise 4 currently the questionnaire is downloaded, at home before the shift is due to start, via Casebook and is launched offline via Casebook. The data for each interviewer on each shift is packaged into a randomly named compressed file and transmitted back to the office where it is smashed into a master file. Reports are created from this file, flagging cases that interviewers have indicated may need additional editing and a team of coders and editors work through each item resolving issues.

3. NSW Questionnaire

The questionnaire is written in in Blaise 4.8 in a Computer Aided Personal Interviewing (CAPI) format, with the interviewer leading the respondent through the questionnaire. In the NSW there is a section of more personal question blocks about religion and sexual orientation, where greater discretion maybe required. These blocks have been designed in such a way that they are available in a Computer Aided Self interviewing (CASI) format, as well as the standard CAPI format should the respondent feel they are unable to complete the section unaided.

3.1. Current Multilingual elements on the NSW

In the current Blaise 4 version of the NSW questionnaire, the multiple languages are handled by the Languages setting in the data model. The NSW questionnaire has three languages; English (ENG), Welsh (CYM) and the Help language (HLP). The language can be set at the beginning of the questionnaire, asking the respondent which language they would like to complete the interview in. The interviewer can use the next form language in the menu to toggle between the languages in case reference to the other language is required. This does not change the language permanently however, when the questionnaire is moved on to the next question, it reverts to the ‘Set’ language.

3.1.1. Character Encoding

Most of the characters in the Welsh alphabet are ‘standard’, apart from all the vowels can have a circumflex (to bach) as well as the letters ‘w’ and ‘y’ which can also have a circumflex.
Examples of the special characters - â Â, ê Ê, î Î, ô Ô, û Û, ŵ and ŷ

To enable this within Blaise 4.8, UTF-8 encoding must be applied to the language in the projects datamodel properties.

The development environment, the Blaise 4 control centre is not capable of receiving these special characters directly from a copy and paste. A third-party text editor has to be used to open the block requiring the special characters, the text inserted in the correct location, saved and then re-opened with the Blaise control centre.

3.1.2. The Helpfile

The NSW questionnaire has an accompanying helpfile, defined as language in the datamodel and the accompanying Q-by-Q help file specified in the Mode Library file, this also must be available in both languages. There are 2 separate pages for each help page, one for each of the available main languages. To get to the correct page, once a language has been selected, a text fill has been utilised that can be changed in the rules for each question that has a Q-by-Q help page. This text fill is then the address for the page within the Q-by-Q helpfile, according to the language required.

3.2. Converting those Multilingual elements into Blaise 5

The same approach can be taken in Blaise 5 as in Blaise 4, by creating languages within in the datamodel and then having multiple entries for each question, description or message. When using the conversion tools within Blaise 5 converting from 4 to 5 these language options are converted directly across and no additional work is required to enable them. This solution is just as limited as the Blaise 4 however, in that the multiple languages all must be defined for each questionnaire object at the datamodel level. There is no re-usability to the additional settings and the encoding

```
IF (QLanguage.LangInt = CYM) THEN
   EduHelp := 'EducationWelsh'
ELSEIF (QLanguage.LangInt = ENG)
   EduHelp := 'Education'
ENDIF
```

3.3. The future Blaise 5 solutions

With the onset of the recent upgrade from Blaise 4 to Blaise 5, ONS has been developing its own standardised Resource Database. Creating ONS themed templates and styles that can then be used across different resource sets. The ability to create a suit of layouts for the standard questions within ONS should make vast improvements to the flexibility, transferability, adaptability, consistency and quality of our questionnaires.

Creating a multilingual questionnaire in Blaise 5 there appears to be a great many more tools to aid the user in doing so. The languages can be added to the datamodel in the same way as in 4, but now there is a Datamodel text editor, and Language selector control and you can define a culture for each language. For the definition of each language you are allowed to define language specific Font Definitions, Styles, Media, and Texts. These can then be mapped from the resource Database to whichever datamodel it is applied. The culture can be defined by a language or the country. For example, English-United States, French-France, French-Belgium. The culture is used to represent the following attributes aligned to a specific culture or language such as numbers, currency, dates and times. The defined culture can even be used to alter the direction in which texts are presented on the screen either right-to-left or left-to-right. NB: The Q-by-Q helpfile is a challenge yet to be resolved!

4. IPS Questionnaire

As the IPS has already gone through a long transformation project moving from paper to tablet our goal is to move from Blaise 4 to 5 changing as little as possible, to keep interviewer training to a minimum.

As with the work on NSW, converting the Blaise 4 IPS questionnaire into Blaise 5 was relatively straight forward. The source files and existing lookup files were converted to Blaise 5 using the 'Source Converter 4 -> 5' and 'Data Converter 4 -> 5' applications that are provided with Blaise 5.

There were some issues that needed to be resolved before the questionnaire could be compiled and run, and a few things that interviewers had requested during development of the Blaise 4 instrument which had to be considered while developing in Blaise 5. These main challenges are detailed in this section.

4.1. Current IPS complexities in Blaise 4

4.1.1. Subsampling

In the Blaise 4 questionnaire, subsampling is carried out by reading the previous form. This is done using a technique we call 'double compiling' because it requires that the questionnaire be compiled twice, first without referencing the questionnaire to create an initial metadata file (BMI), followed by a second time once reference to that BMI has been added. As Blaise 4 doesn't check to see if a database exists when compiling, we can reference the database file (BDB) that's created when the questionnaire is run, allowing us to search the BDB with the Primary Key minus 1 to read the previous forms data. Each form holds fields which act as a current total of full/filter forms, as well as group sizes and target amounts for full/filter forms, so reading the previous forms data allows us to determine which type the current form should be.

4.1.2. Creating a new form without losing shift information

Running the questionnaire through Casebook allowed us to first ask interviewers to complete a shorter "shift details" questionnaire to fill in key shift information that can be read in by each form of the main questionnaire. This saves the interviewer from having to re-type the same information multiple times but creates extra databases containing data duplicated in the main questionnaire that in turn are stored on our servers by case management processes. It's been a goal of the office to avoid having a separate shift information questionnaire when moving to Blaise 5 as it leads to lots of un-necessary files in several locations.

4.1.3. Duplicate case identifiers.

In the Blaise 4 questionnaire, shift information is keyed in manually by the Interviewer. The key variables are Shift Number, Form Number and Interviewer number which make up the primary key, and Port Route which describes which port the interview took place in. As Casebook gives each set of data a random name when packaging it for transmission back to the office, if any of these details are incorrect it doesn't matter as the data are set aside and not smashed until the issues are resolved by the coding and editing team. This prevents data with an incorrect primary key overwriting other legitimate data.

4.1.4. The navigation panel

One of the features of the Blaise 4 instrument that Interviewers gave a lot of feedback on was the navigation panel which was designed in the Blaise Menu File (.BMF). During development we reduced the number of panels, added the option for a panel on the other side of the tablet for left-handed interviewers and removed the flags as some were offended by having to select certain flags for language that didn't match their country. This panel was the most contentious part of Tablet development and it was difficult getting a consensus amongst interviewers how best to implement it.

4.2. How these complexities were handled in Blaise 5.

4.2.1. Subsampling

We have been unsuccessful replicating the 'double compile' process using Blaise 5 so struggled with this while working with early versions of Blaise 5.

However, once the 'Actions Setup' feature was added in a later version, allowing a Manipula procedure to be called during the running of a questionnaire, the subsampling problem was easily overcome.

On the first form once the shift information has been filled in a Manipula procedure is called that writes out all the necessary data a separate file. That file is then searched and read in when the next form is running so that code in the rules can dictate whether it should be a Full or Filter form.

4.2.2. Creating a new form without losing shift information

As with the subsampling issue, when the 'Actions Setup' feature was added, this became a very easy problem to solve and we were able to get rid of the separate shift details questionnaire completely.

First we established how many of the 167 fields in the shift details questionnaire were essential to the main questionnaire.

There were only 8 which were essential, so when the form number is 1 these data are collected and then written out to the same file that the subsampling information is written out to (as described in 4.1.1). These values are read in at the key page of the next form to save the interviewer some typing (increasing form number by 1) and throughout the questionnaire as needed.

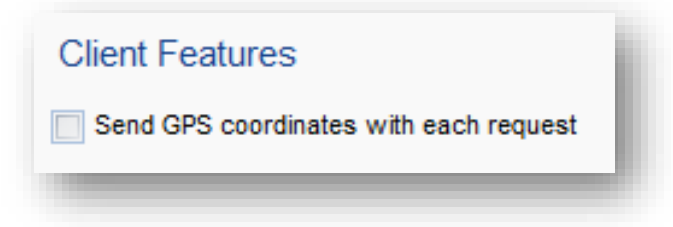
4.2.3. Duplicate case identifiers

As it's vitally important that the key shift information fields are accurate in Blaise 5 to prevent data being overwritten, we decided quite early on in Blaise 5 development that we'd use lookups for everything. The research team already had two separate files containing…

a) Shift numbers, Port routes and Shift dates
b) Interviewer numbers, names and the port they work at.

These files were converted into lookups in Blaise 5 and are used at the Primary Key pages to fill in the information. The first Primary Key page uses lookup 'a', to set the Shift number and port route, and Shift date. The second Primary Key page displays the information that's been selected on the first page, highlighting the shift date in red if it doesn't match the system date asks the interviewer to confirm that these details are correct by selecting their own name in lookup 'b'.
Unfortunately, due to unforeseen circumstances, shifts are sometimes moved around to different dates during the month, so we have to allow for shift dates that don't match the system date.

In the future we'll consider using the GPS features of Blaise 5 to set the Port details and further reduce error:



This will be dependent on hardware as we're aware that a new Tablet device is currently being considered. We will likely enable GPS-recording at first when a survey month is live to see what range of co-ordinates we get back for each port before automating the process.

To ease the transition from Blaise 4 to 5, we've subsequently implemented the Interviewer Name lookup in Blaise 4.

4.2.4. The navigation panel

The navigation panel was quite tricky to try and resolve. In one of the earliest versions of Blaise 5 we were able to recreate the 'Keyboard action' of the Blaise 4 menu file by adding a button to the page template in Blaise 5 that, for instance when pressing the '2' button, assigned the active field a value of: 'Page.ActiveField.ValueAsText + '2'' followed by a save.

This unfortunately was dependent on a bug that was fixed in a later version of Blaise 5. Since that fix we've been unable to recreate the Keyboard Actions of Blaise 4, but there's really no need. Our interviewers are already using the On-Screen Keyboard (OSK) that is part of Windows 10 for text input, and it is very straight forward to change a setting and make this appear with a Number-Pad. Showing the interviewers how to turn on the Number-Pad allows us to free up valuable screen space which would otherwise have been taken up by the clunky and un-necessary navigation panels.

5. Summary/reflections

Overall, moving the IPS from Blaise 4 to Blaise 5 has been very straightforward. The Data and Source convertors provided with Blaise 5 are excellent and save what could have amounted to many hours of work.

Moving the NSW from Blaise 4 to Blaise 5 has been similarly straightforward and the research team, field interviewers and Welsh government have all been positive in their feedback when consulted. The 2020 NSW questionnaire was produced in parallel in Blaise 4 and 5 to provide a comparative test for all concerned, as well as a development and testing plan for actual go-live in 2021.

The transition has been easier for IPS and NSW than some other surveys due in no small part to the hard work David Kinnear (ONS) has put in to make a very good Resource Database for our team to use. The resource database is full of applicability conditions that have meant all fancy layout features and field panes for all types of questions are applied automatically. That enabled us to focus purely on issues described in this report, and other "nice to have" improvements.

Being in a current state of transitioning from Blaise4 to Blaise5 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, 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. Links and References

[1]. Lan Benedkit. Developing the UK Passenger Survey in Blaise 5 on Tablet Computer, Office for National Statistics, UK, Statistics. IBUC 2015, 16th International Blaise Users Conference

[2]. Mike Hart. Converting Paper Collection on the International Passenger Survey to Blaise 4 on a Tablet Computer, Office for National Statistics, UK, Statistics. IBUC 2016, 17th International Blaise Users Conference

https://gov.wales/national-survey-wales

Continued Adventures Transitioning from Blaise 4 to 5

Vito Wagner, Charles Less and Peter Kilpatrick, United States Department of Agriculture – NASS

1. Abstract

Although much of the individual survey code translates easily with the included conversion tools, many pitfalls await your transition. Other papers have documented numerous coding modifications needed to ones datamodel code and this issue will continue to plague the developer. Developing a house style layout gives your organization many possibilities, but with great potential come great responsibilities. Additionally, not all layout features in Blaise 4 are easily paralleled in Blaise 5. Changing versions within Blaise 5 also poses unique challenges. Major releases and minor releases sometimes change the columns required resulting in incompatibility of prior projects within the same table structure of a new project. These errors manifest themselves in multiple messages which can be frustrating to the user and difficult to describe when seeking assistance. Internal organizational process may also require changes with some being very subtle and easily overlooked such as changes in the order that the Blaise system executes some request. We look to lead the reader through some of these struggles and share in the victories that will come.

2. Introduction

The National Agricultural Statistics Service (NASS) has been using Blaise for many years. Ten years ago we conducted about 25 major surveys per year using individual Blaise datasets with the data distributed to individual field office LANs. Our last major upgrade in our Blaise infrastructure was to move from a distributed system to a centralized structure and storage. Today, we collect and edit most data using Blaise 4.8.4 utilizing a single generic in-depth data structure in a MySQL database. Our 20 major projects have grown to 150 separate instruments conducted with approximately 1,200 separate survey instances conducted per year. Our data collection takes place across five telephone data collection centers and data is edited predominately in 12 regional offices.

3. Jira

We cannot thank staff at CBS enough for the assistance they have offered through Jira. The additional tracking has been valuable to our organization. It is also helpful that one of us can submit a request and others can easily see it and add on useful information as needed. The searchable interface also nicely separates support, feature request, and bug reports to the Blaise team from our normal internal agency email interactions. Even with our time zone difference support is fast. Large example files can more easily be attached than is typically available via email, so entire solutions and test datasets can be transferred to whomever is providing you support. This feature has been immensely helpful when trying to troubleshoot an issue.

Our developer group has submitted well over 100 separate requests via Jira. New software and unique implementations will lead to some rare issues. Our agency uses the Blaise software heavily in CATI data collection, manual data editing, and requires the use of one database for all of our data collection for ease and timeliness of downstream processes. Multiple times we have asked ourselves: Have we done something wrong or are we superior software testers? Either way, help is definitely available and should not be ignored.

4. Layout – Our Paradox of Choice

One of the great features that is most easily seen by our end users is the ability to better customize the layout. A common layout issue is that we do not know how to properly describe what we are attempting to accomplish.

4.1 Edit Mask or Thousand Separator

Creating better displays for fields, such as the phone number, which has a consistent format is easily accomplished with an edit mask. The EditMask property works exactly how you would expect for something like a telephone number in the US format. Our test dataset uses a phone number that would be formatted as follows: (202) 555-1234.

Modern users are used to seeing large values with separators to make reading easier. The number one million is more easily read when displayed as 1,000,000 than 1000000. When we began our journey to Blaise 5 the thousand separator was not available. If you attempt to use an EditMask to add a thousand separator, the EditMask would be setup in a format such as 999,999,999. When entering the value of ten thousand, the user would see: 100,00. A European user would interpret such a value as 100, but the American user would simply be perplexed that your system did not know how to handle this number at all. The ToString function can be used to format the display of fields after they are collected; however, this timing is too late to be the most useful to the user entering data. Fortunately, the ThousandSeparator and DecimalSeparator are available as properties to the NumberTextBox control.

4.2 Screen Design

Having hundreds of users, we have thousands of opinions of how screens should be laid out. Building a house style is difficult from the ground up not only making design choices but learning the skills to use the Blaise Resource Database. Hiring out our major layout design was useful to get us started as beginning from scratch was very difficult. We have found making small adjustments after end users experience the new look and feel to be simpler.

Our five telephone data collection centers and our public affairs office were both involved in early design choices since they should be more knowledgeable than programmers. We also sought buy in from these user groups since they would be with us through the early growing pains. In cases such as picking color palates, few had strong opinions during the design phase but then would have concerns later. (Real feedback from a user “This red background makes me mad. I’m seeing red!”) Other seemingly simple choices such as where the contact information should go is not agreed upon.

Blaise 5 does give you the options. You could make a toggle button or hot key to put the respondents name and telephone number at the top of the screen or bottom of the screen. Would this design choice be smart? Would the survey methodologist find it wise?

4.2.1 Keyboard vs Mouse

More options are easily made available via mouse clicks in Blaise 5 than we experienced in Blaise 4. The desire to allow for more types of devices is lost on the user who sits at their desk and believes they can most quickly act via the keyboard. Most of our on screen buttons are also assigned to a hot key that matches similar functionality that existed in our Blaise 4 setup; however, the buttons are more prominent and the anti-mouse users do not like their old ways to be infringed upon.

4.3 With Great Power Comes Great Responsibility

The solution Settings can have dramatic consequences for performance. Based on values in the Rules tab, a Server Contact can be required after leaving a changed question or at a page switch. There are values in both options and only you can decided what is best in your situation. Most users probably want a hybrid where request are only sent when they are going to matter.

Audit Trail Level can range from None (fast) to Keyboard (slower or maybe slow). Somewhere in the middle exists your ideal setting. Save and Delete Session functions change what the user sees if they exit a form and reenter but also how the data makes it back to your desired dataset. It is possible to setup so you delete the session data and never save the data to your database. This setup would probably be unwise but may be desired to implement and option to Undo All Edits.

4.4 Fear of the Unknown

Change is hard. Many developers see a change and think it is hard and they would rather leave things as they are currently. Many of our projects are repeated for years with little change from year to year. This situation is nice and comfortable.

Changes to the Blaise language itself appear fairly minor. Most code easily translates with the Blaise 4 to Blaise 5 converter. Changes required such as implementing the former TABLE as a BLOCK but handling the display in the Layout or Resource Database can be ignored with few ill effects beyond the less desirable display. More dangerous is the Rules Behavior found in Settings that can be Strict (Fields with DK, RF, or EMPTY are not equal to the Default Value) or Blaise 4 compatible (Fields with DK, RF, or EMPTY are equal to the Default Value). A seemingly simple comparison of IF MyField = 0 has opposite meetings in these two choices. Conveniently, Blaise warns you when building the project, but you are require to read the message and act upon it. In Blaise 4, developers largely only had to update their individual projects code. With the extra capabilities introduced in Blaise 5, there are more areas where coding and settings must be updated including source code, settings, and layout.

5. New Software Versions

Version updates bring hope that bugs you have found have been squashed and features you have requested have been implemented. In practice, you are ready for a wild ride of compatibility testing and adventure.

5.1 Installation

Departmental regulations have taken ways many of our rights and abilities to do our own installation. We download the latest version of the software and have the latest build installed by our IT service desk on our testing server and on a developer's individual virtual machine. By internal departmental policy, these installations are done by two different people with varying turnaround times.

In early versions of the software, it was difficult to tell which version number was used to build your instrument files and which version was installed on a given server park. In version 5.6, you could see which version an individual survey package that is installed on the server park was created in, but the only way we currently can see the server's software version is to remote out to the server. (Note: we are certain that the CATI Dashboard will someday tell you what software version is running on your server park.) Access to viewing this remote information is locked down to system administrators, so they must be consulted to verify versions.

5.2 When an Upgrade is not an Upgrade

Database compatibility was an early struggle as new columns would be added to the schema. With installations of 5.5.x and earlier multiple attempts to upgrade Blaise versions to deal with an editing issue would make it impossible to create a CATI daybatch.

In our typical CATI problem situation, installation of the survey project would appear to be successful. Then an error such as the figure below would be encountered when attempting to make a daybatch.

In this particular case, OperationTime was added in 5.6.1 requiting a new column to be added to one's database tables. The easy solution available was to drop and read the CATI related databases; however, this solution also lost the history of records, essentially resetting progress to zero and losing telephone appointments and other such paradata. We also do not have the rights (nor should I be trusted) to drop and add tables in our database environment, so coordination with a database administrator would be required.

Additionally after experiencing a few upgrade issues related to data incompatibility, all issues looked like compatibility issues. Errors shown in the CATI Dashboard after installing new projects and upgrading versions often show the same results when you have issues with newly added columns to the schema Improvements to such upgrades have been parts of many releases with vast improvements to compatibility coming with 5.6.7. Warnings in the version changelog are especially welcome in this context.

5.3 The Version Juggling Act

Blaise 5 installs are more complicated than we were used to in Blaise 4. In our Blaise 4 setup, installs were done once and then the resulting directories were copied to a shared area where all developers could access the Control Centre. This setup has not been possible for us with Blaise 5, so each developer is require to have an install on their individual workstation. Additionally, that gives a developer access to only one version of the software making testing new releases more difficult.

We also can get lost in the different versions spread amongst our different servers and developer workstations. These differences can lead to problems when a new version used to prepare a package expects different columns than the version included on our server park.

A strategy was developed to deal with the juggling act of switching to the latest version, but it is clunky at best. We install a prospective new version on our test server, so there is a server park to install to and a CATI Dashboard to test daybatch compatibility. To build a package in the newest version, the tester can remote to the test server and build the solution using the installed Control Centre there, they can also install there using the Server Manager. Testing can then resume in the user's normal workstation and via our normal menu system.

5.4 Upgrading Your Solution

Especially when switching between major releases, the reader should note that your solution may be upgraded. The Control Centre politely and helpfully informs you of this fact. You may also upgrade your projects Resource Database. Without smart planning a version of your solution compatible with the most recent stable release may not be available nor is it easy to automatically roll back or downgrade your solution.

6. Bug or Feature

6.1 ValidationStatus

Blaise 4 contained the keyword FORMSTATUS that has been replaced by the similar keyword VALIDATIONSTATUS. The values returned by VALIDATIONSTATUS via manipula are not always what we expect. To get the status associated with the Data Entry Setting you desire (in our situation CheckedEditing akin to CADI), a CHECKRULES can be applied when opening each record. However, if your purpose was to write a report, you may encounter the situation where the record is really NOTCHECKED in the database, but your report will make it appear to be some other status due to the CHECKRULES.

One could solve this problem by actually performing and saving the data of the CHECKRULES but they would have to pay the performance cost of an update instead of a mere read. VALIDATIONSTATUS is available in the database and can be accessed directly via SQL at much faster speeds than manipula. If the option exists, we find a SQL query vastly outperforms manipula.

6.2 Harmless Changes

One of the most hotly anticipated changes for our developers has been increase in data file compatibility through Harmless Changes. Our agency sometimes produces and enters into production with projects on a very short timeline where shortcuts are sought and deep, thoughtful testing occurs in production. In a small minority of these situations, fields or edits need to be added, allowable values widened or narrowed, or additional enumerated types need to be added. Previously such changes required producing a nearly identical project that now added the expanded values.

Moving to the instrument with expanded values required both projects in place. Next a manipula would be used to copy the data from the old project to the new project. We creatively called this manipula an old2new and it was feared by all due to necessity to be performed when data collection was not happening (which meant after 9 pm but before 6 am while also avoiding scheduled server processes in the early morning) so data collection would be stopped, and it also risked mistakenly copying the new data to the old instrument which would effective erase all your data.

The addition of Harmless Changes made the prospect of this data move seem like it would be a thing of the past with the data transition easily being handled by the system. The mere mention of this feature brought developers joy, and they might even exchange the famous “Blaise 5”.

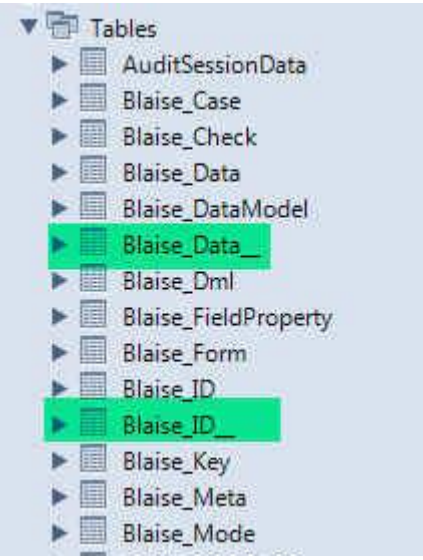
6.2.1 Harmful Changes – Our Data is Gone

Early surveys in our journey to Blaise 5 had Harmless Changes Support set to AutoUpdate. Just like the users we just lamented about, we never tested how this feature would work. One day we mistakenly installed a package that would meet the definition of a Harmless Change. Instead of the joy that was expected, browsing the bdix in the Data View made it appear that all of our data had gone missing.

Furthermore, querying the data via SQL also showed tables with the expected number of rows but no data in any columns including the primary key.

Assuming we were all about to be fired and would need to being sending resumes to someone very desperate in the Blaise community, we went to warn our telephone enumerators what disaster we had caused. Surprisingly, they did not seem aware that we had done anything wrong. Furthermore, they were still actively calling respondents which meant at least the name and address information had not been lost.

Looking at our SQL database, we notice some additional tables had appeared. We assumed that we had done something ill-advised as we could see what looked like our data in these two tables ending in __. It was not clear how to merge the Blaise_Data__ and the Blaise_ID__ tables together based on the key information we could see. Defeated, we decided that not even the most desperate shop would want to hire us.



After telling others of our failures, data began showing back up in the Blaise_Data and Blaise_ID tables as if by magic. When a telephone user would be delivered a form by the call scheduler, the data would be populated in the tables as we expected. Tempted to call this a Blaise miracle, we instead discovered that using the Start in DEP feature found in the Blaise Server Manager along with explicitly naming the primary key allowed us to return the data of all records. Fortunately, this early project only included about 100 records to be collected.

Some refinement appears needed to make Harmless Changes less nerve wracking such as more automatic copy of the data back to the desired tables. We are certain the Harmless Changes will be a useful feature in the future and would like to explore in a safe (not production) environment.

7. Introducing Users

7.1 CATI Users

A major feature of the Blaise software for us has been its ability to handle telephone data collection well. Telephone enumerators also find change frightening. Most of our CATI users are working a second job in the evening after their primary job during the day. Training on two different systems can be difficult for them. We attempted to keep many of the same functions and shortcuts as possible between Blaise 4 and

Blaise 5 projects. For example in Blaise 4, we used F5 as a hot key for Don't Know and F6 as a hot key for Refusal. Our Resource Database in Blaise 5 allows for these same hot keys. Even the minor difference where F6 is now displayed as Rather Not Answer has caused confusion for some users.

7.1.1 CATI Dashboard Issues

We have encountered a few important issues with the CATI Dashboard.
Supervisors tasked with reviewing the CATI specifications see the new layout, assume everything is completely different, and mistakenly let the time between busy dials and no answers be set at 5 minutes. When an appointment was missed, the scheduler undesirably rapidly redelivered the same few records to the point that enumerators recognized the respondent by name.

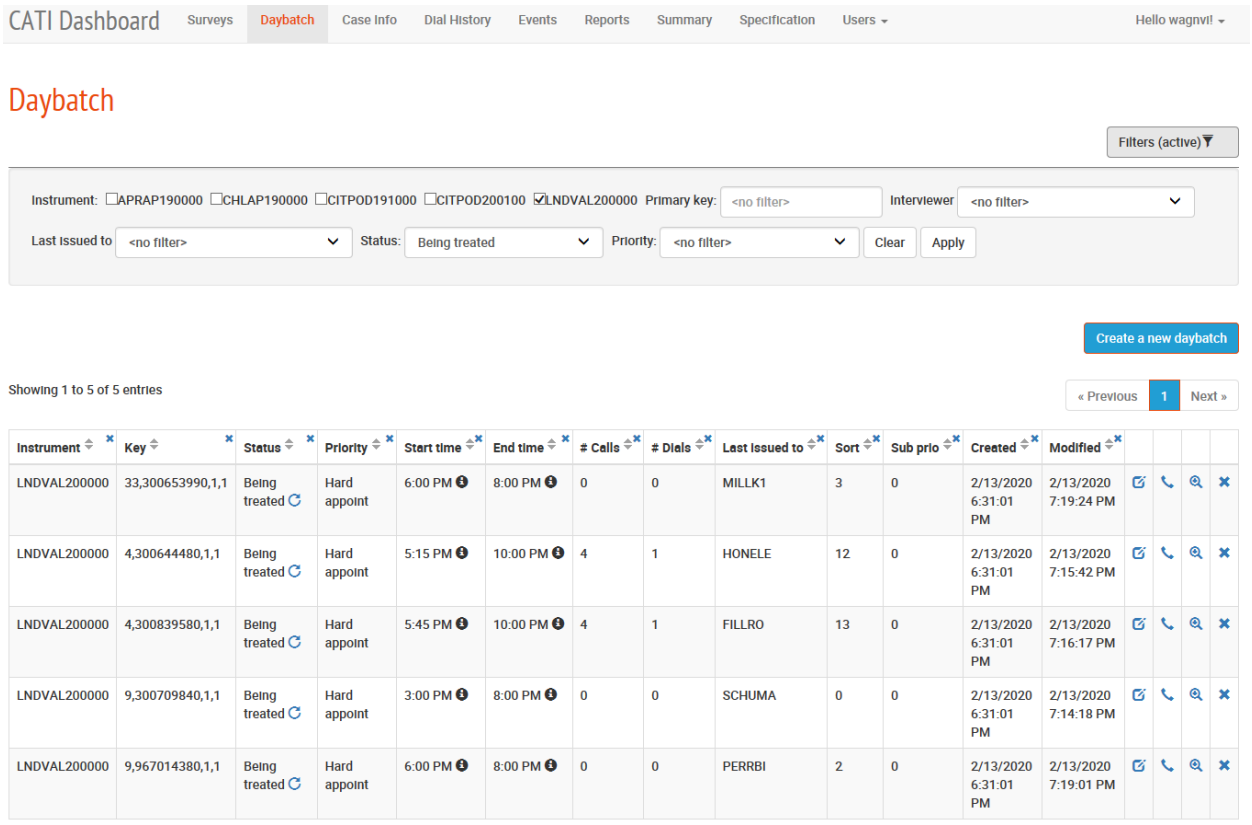
It may be desirable to see all information for all installed CATI projects on the same dashboard. Finding the particular sets of records you would like to explore can typically be done through the available filters that only differ slightly from tab to tab. However, if a poor development choice of the included field selections exists for one installed project in your server park, the Case Info becomes difficult to use on its own.

Due to some series of events not yet resolved at this writing, some records will not write information to the Dial History. You can see that these records were contacted when looking at the Events, but the scheduler does not seem to recognize all of the actions such as incrementing number of dials and increasing number of calls.

Time zones are also an important concept for us. Typically we call records across 6 different time zones and do not want to call before or after certain times in their local time. This feature continues to work as we expect. Unexpectedly with what appear to be correct local time settings for the server park and on the btrx file, daybatches expire at 6 pm local time which so happens to be midnight GMT (also known as tomorrow). We have developed a few work arounds for this situation. One workaround was to set our time as -12. Then the daybatch will expire at midnight local time instead of 6 pm. When this change was implemented more records appeared to be no longer writing to the Dial History (response data of our actual survey questions was confirmed to be stored).

7.1.2 CATI Dashboard – More Information Than You Require

The paradata available in the CATI Dashboard is expansive. Specifications are very similar to what was used in the past making user training easier. Case Info and Dial history can be seen easily. Particularly useful is the ease of seeing who the active users are and what cases they are treating.



Sorting of the daybatch has also proved useful. With little instruction, our enumerator supervisor can see how calling is progressing in real time and how many more records are to be called today.

7.2 CADI User - Editors

Most editing in NASS takes place in our interactive edit. This feature is the most lacking in our current Blaise 5 system. Manipula were used extensively in our prior system to control which records were to be edited and maniplus dialogues were used to guide users through edit choices.

With the large number of projects and some being conducted weekly, we have reused much of this edit system between surveys and relied on the manipulas being built and run via command line with options passed to specify the dataset desired. In Blaise 5, the instrument builder has a large overhead too large to build these manipulas on the fly as we did in the past. Many manipulas can be built in advance and the keyword (var) used in the declaration of the datamodel. This style of development is more difficult for us, but it is possible. We have begun using this style with only minor issues due to our own programming. The startup time to begin a manipula remains a concern, and these times are even larger when using manipulus, but we also have been finding alternatives such as direct SQL queries and displaying virtual tables in Visual Basic.

The views expressed in this paper are those of the authors and not necessarily those of USDA-NASS.

PAPI to Blaise 5 – Challenges and Solutions in Creating Complex Tables

Emily Caron, Lilia Filippenko and R. Suresh - RTI International

1. Abstract

Recent Blaise 5 development at RTI International included creating large tables to display preload data, with the option of entering or editing the data values in similar fashion to what was previously done on paper. This brought about challenges in coding and configuring the Resource Database. The process we followed and the solutions we developed to overcome the challenges will be discussed in the paper using a household roster example.

2. Background

RTI International has a long history with developing Blaise instruments for use in survey research, as part of its overall mission to improve the human condition. We continue to primarily use Blaise version 4.8 due to long-standing survey projects using it and its overall stability. Within the past few years we have also delved into Blaise 5 with software evaluation, integration into existing case management systems, and deploying to production. We currently have one web survey in production and previously conducted a CAPI pilot survey. Three other Blaise 5 projects are in development stage with plans to deploy to production this summer.

Images and functionality mentioned in this paper come from a demo survey in Blaise version 5.6.9.2082. This survey mimics the features and behavior we wanted to highlight after programming an instrument for a client and includes different question content than what was used for the client. For simplicity's sake, the client will be referred to as "ClientX" in this paper, and discussion below will be worded as if this was the exact instrument we created for them.

3. Our Task

Recently we were tasked with creating Blaise 5 surveys on tablets to collect information which was previously entered by ClientX onto paper forms (PAPI). The PAPI forms had been in use for quite some time and contained several large tables as well as previously collected data printed on the forms. Much of the data collected was specific to household members. If a table happened to fit within the regular paper form it would be located there; otherwise, supplemental paper forms were often used to store the previous data and/or collect new data in table format.

For the Blaise 5 surveys, ClientX requested that most previous data be visible on screen when present. It needed to be retained and readily available for interviewer access. Some of this data would be available for reference only as it was not expected to change, and other data should be verified with the ability to change the data if necessary.

Another requirement was that if preload data existed and then was changed or otherwise indicated as incorrect, comment fields would route to collect any pertinent details about the changes. Additionally, the interviewer remarks feature should be available on demand.

3.1 Task Challenges re: Large Tables with Preload Data

As previously mentioned, some of the tables we needed to display on screen were overly large, and this posed a problem considering screen width and visibility. Not only would the tables take up a lot of real estate for the data we needed to collect, but we also needed to show preload information in a manner that would be readable and readily accessible to the interviewer at the time they were collecting data.

Some different layouts were considered and tested.

Option 1: Show a read-only preload table over or under a table to update or collect data. Considering the size of some of the tables, this was not a good option because of vertical scrolling. Comparing original data to the new data row was too far to travel across the screen.

Option 2: Use odd-numbered rows to show preload data, with even-numbered rows to update or collect new data. This was a bit better on space, but a little more confusing to the eye. In households with a large number of members, this would make the table unnecessarily long and require scrolling.

Option 3: Use table tags within field text above the table to display any available preload data, while leaving the table itself to update or collect new data. Like option 1, this could cause vertical scrolling and was too far of a distance between original and new data rows.

3.1.1 Our Solution for Large Tables with Preload Data

Our eventual solution was to combine the preload data and new capture or potential changes into one table with the same cells used for both. We just needed to find a way to ensure that the interviewer could access preload data on demand if needed, no matter if they had made adjustments to what was initially loaded into the table. We were able to do this with the help of a row level PreloadIn flag and a LoadHH procedure.

A PreloadIn field was created at the row level to serve as a flag to allow or prevent preload calculations when needed. Initially in the Rules section we left this field empty. We also created a LoadHH procedure to handle setting all values in the table that we knew from preload data.

```
PROCEDURE LoadHH
  PARAMETERS
        piRow : INTEGER
        EXPORT pePreloadStr : STRING
  RULES
        iPerson         := Preloads.Person[piRow].iPerson
        First           := Preloads.Person[piRow].First
        Last            := Preloads.Person[piRow].Last
        DOB             := Preloads.Person[piRow].DOB
        iAge            := Age(Preloads.Person[piRow].DOB)
        Sex             := Preloads.Person[piRow].Sex
        HOHrelation     := Preloads.Person[piRow].HOHrelation

        …[some other calculations]

        PreloadIn       := 1
```

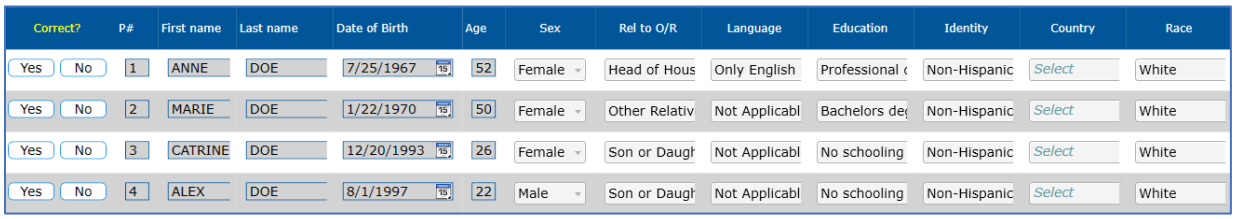
```
        pePreloadStr     := ''
ENDPROCEDURE
```

The procedure was kicked off early in the Rules after evaluating PreloadIn value for each row. Since PreloadIn was not 1 for any rows yet, this served to initialize all known data into the table. Near the end of the procedure, PreloadIn value was set to 1 to signify that the preload info was loaded.

```
PreloadIn.KEEP
IF PreloadIn <> 1 THEN
  //fill preload data
  LoadHH(piRow,PreloadStr)
ENDIF
```

Upon arriving at the household demographics table, the interviewer could review the preloaded information and verify it as being correct or mark it as incorrect. The first column required a response or they could not move to the next screen. All columns except "Correct?" started out as read-only using .SHOW statements.

Figure 2. Household Demographics Table, original view

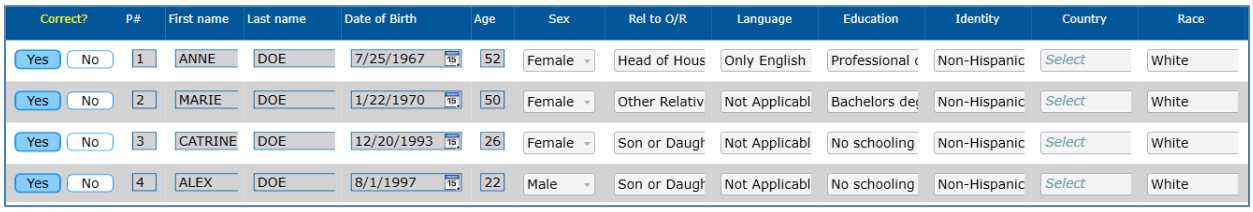


Correct?	P#	First name	Last name	Date of Birth	Age	Sex	Rel to O/R	Language	Education	Identity	Country	Race
Yes No	1	ANNE	DOE	7/25/1967	52	Female	Head of Hous	Only English	Professional (	Non-Hispanic	Select	White
Yes No	2	MARIE	DOE	1/22/1970	50	Female	Other Relativ	Not Applicabl	Bachelors de(	Non-Hispanic	Select	White
Yes No	3	CATRINE	DOE	12/20/1993	26	Female	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	White
Yes No	4	ALEX	DOE	8/1/1997	22	Male	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	White

(Note: The table was much larger than this, but some rows and columns were cut for visibility's sake.)

In the Rules, we now needed to evaluate the different values of the Correct field. If Correct = Yes, this continues to consider all other columns as read-only.

Figure 2. Household Demographics Table, all data verified as correct



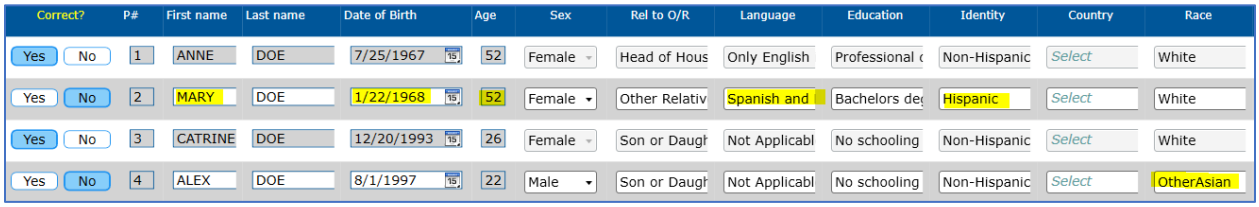
Correct?	P#	First name	Last name	Date of Birth	Age	Sex	Rel to O/R	Language	Education	Identity	Country	Race
Yes No	1	ANNE	DOE	7/25/1967	52	Female	Head of Hous	Only English	Professional (	Non-Hispanic	Select	White
Yes No	2	MARIE	DOE	1/22/1970	50	Female	Other Relativ	Not Applicabl	Bachelors de(	Non-Hispanic	Select	White
Yes No	3	CATRINE	DOE	12/20/1993	26	Female	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	White
Yes No	4	ALEX	DOE	8/1/1997	22	Male	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	White

Let's consider the scenario where Correct = No for one or more rows where data needs to be modified. In this situation we needed to be sure to prevent the continual overlaying of valid changes to the data by the preload data. This is where PreloadIn flag came in handy once again.

When handling Correct = No, first we reset PreloadIn = 1 to ensure the LoadHH procedure didn't get kicked off for any of these rows. Then we did .keep statements on all of the fields, followed by asking the fields where we would allow data changes:

```
IF Correct = No THEN
        PreloadIn := 1
        iPerson.keep
        First.keep
        Last.keep
        …[other .keep statements]
        iPerson.show     //not allowed for edit
        First
        Last
        DOB
        iAge := Age(DOB)
        iAge.show        //calculated, not allowed for edit
        Sex
        HOHrelation
        Language
        EducLevel
        Hisporig
        CountryOrigin
        Race
```

Figure 3. Household Demographics Table, some rows indicated as not correct and fields are updated



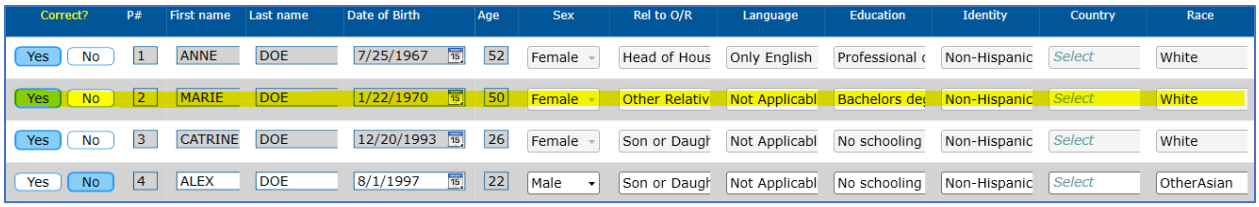
Correct?	P#	First name	Last name	Date of Birth	Age	Sex	Rel to O/R	Language	Education	Identity	Country	Race
Yes / No	1	ANNE	DOE	7/25/1967	52	Female	Head of Hous	Only English	Professional	Non-Hispanic	Select	White
Yes / No	2	MARY	DOE	1/22/1968	52	Female	Other Relativ	Spanish and	Bachelors deg	Hispanic	Select	White
Yes / No	3	CATRINE	DOE	12/20/1993	26	Female	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	White
Yes / No	4	ALEX	DOE	8/1/1997	22	Male	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	OtherAsian

So now we had some data modifications. But recall that one of the requirements from ClientX was for the interviewer to have easy access to any available preload data. What if they needed to get back to the original data values? Due to space and other reasons, we were now combined into one set of fields all in one table.

We achieved this by using the PreloadIn flag again, with the use of the LoadHH procedure. When Correct <> No, we could invoke the code below in order to return data to its original preload state if the interviewer switched from Correct = Yes to No.

```
Correct
…[some other code]
PreloadIn := 0
LoadHH(piRow,PreloadStr)
iPerson.show
First.show
Last.show
DOB.show
…etc
```

Figure 4. Household Demographics Table, one Correct=No row is set back to Yes to restore preload data



Correct?	P#	First name	Last name	Date of Birth	Age	Sex	Rel to O/R	Language	Education	Identity	Country	Race
Yes / No	1	ANNE	DOE	7/25/1967	52	Female	Head of Hous	Only English	Professional	Non-Hispanic	Select	White
Yes / No	2	MARIE	DOE	1/22/1970	50	Female	Other Relativ	Not Applicabl	Bachelors de	Non-Hispanic	Select	White
Yes / No	3	CATRINE	DOE	12/20/1993	26	Female	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	White
Yes / No	4	ALEX	DOE	8/1/1997	22	Male	Son or Daugh	Not Applicabl	No schooling	Non-Hispanic	Select	OtherAsian

The caveat is this does lose any newly changed data, but ClientX was fine with it because this situation would only occur during the interview when any necessary data modifications could be easily re-entered, and in general this was only expected to happen under rare circumstances.

3.2 Task Challenges re: Handling Comments for Changes to Preload Data and Remarks

A requirement from ClientX was to force comment capture when any preload data values were changed. In many cases, we were able to do this quite easily on an individual question basis. Other screens contained simple table layouts, such as shown in Figures 5 and 6, where comment fields were routed as a new row if a field value didn't match its counterpart in a preload block.

Figure 5. Simple Address Information Table, original view

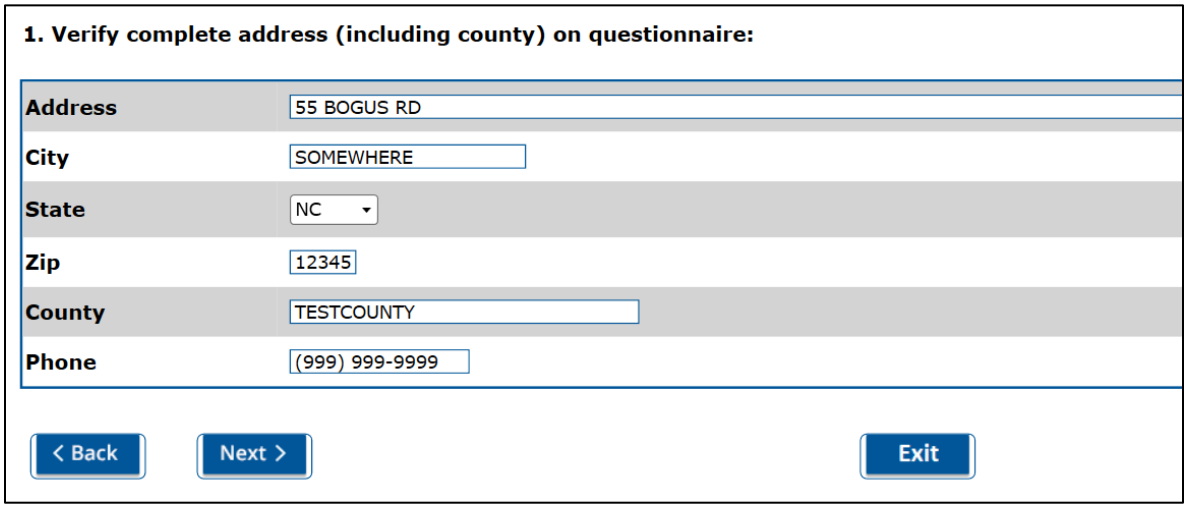
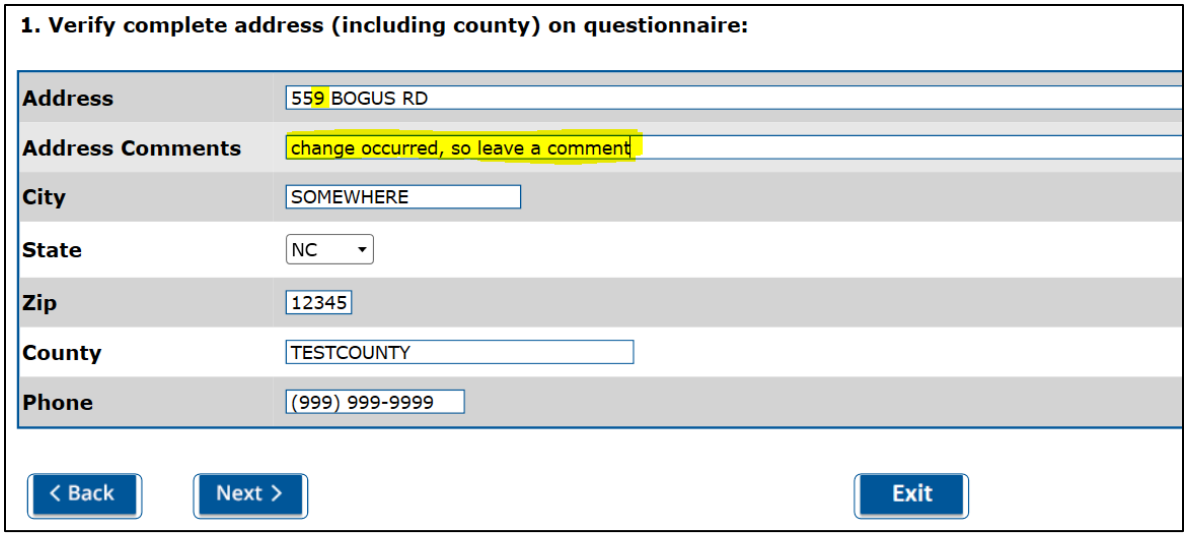


Figure 6. Simple Address Information Table, change to Address field put Address Comments on route



In other situations, dealing with comments became problematic because we needed to put them on route when preload data was modified in a large table. We started out by adding a comments column to the right-hand side of tables, which would force comments to be left on a row-by-row basis when changes occurred to fields within that row. This was not desirable to ClientX because either the string length of the comment fields had to be very short to completely fit on the screen, or the comment column width was shortened in layout so not much of the comment would be visible at one time (the cell would scroll).

Regarding interviewer remarks, this feature appeared to require updates in the Resource Database to get it to work similar to behavior in Blaise 4. Only when the paperclip image was visible did it seem to work correctly, but otherwise it was not right.

3.2.1 Our Solution for Handling Comments for Changes to Preload Data and Remarks

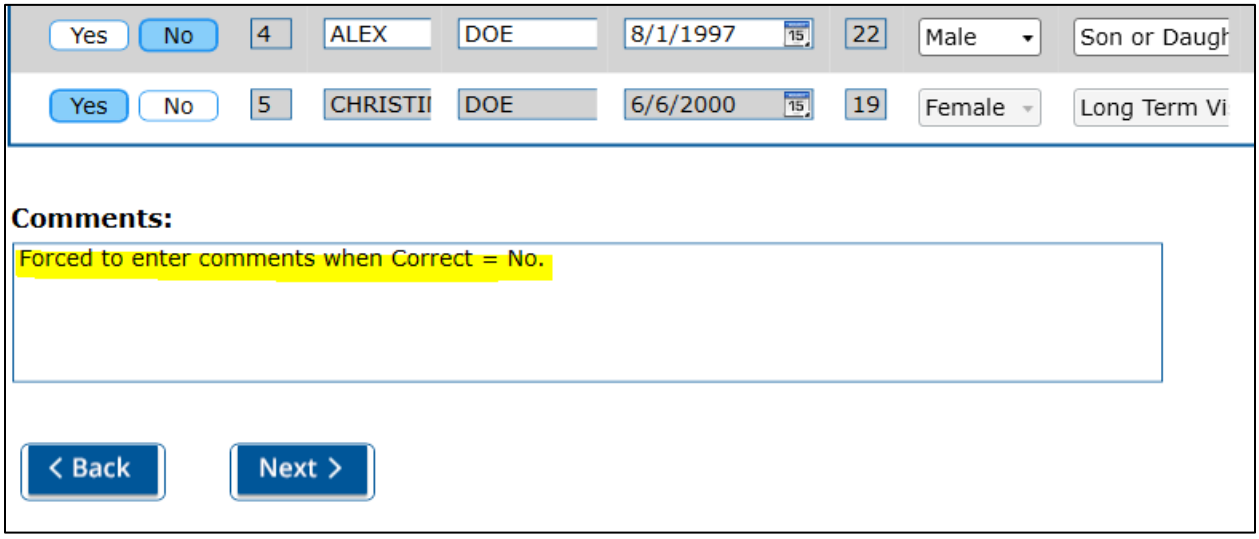
It was decided that for comments connected to large tables, one very large comment field routed below the table would suffice. The field was large enough to accommodate any and all information about data changes. The interviewers were previously trained to leave comments on paper about any changes they happened to pick up on using PAPI mode, so this was still an improvement.

The comments field would be required if any rows showed Correct = No, and this was done using a QxxChanges auxfield (xx corresponded to question number). Putting it all together:

```
QxxChanges := EMPTY
IF Preloads.HHTotal > 0 THEN
        Qxxintro.SHOW
        Qxxtable(Preloads.HHtotal)
        IF Qxxtable.HH[Preloads.HHTotal].Correct = RESPONSE THEN
          FOR k := 1 TO Preloads.HHTotal DO
            IF Qxxtable.HH[k].Correct = no THEN QxxChanges := yes ENDIF
          ENDDO
        ENDIF
ENDIF
IF QxxChanges = yes THEN
        QxxComments
```

ENDIF

Figure 7. One or more Correct = No responses results in required comments



For interviewer remarks, we updated the shortcut to Ctrl+C as this would be easier to access from the onscreen keyboard with just one hand. The Resource Database needed modification to use the newly created "OnCtrlC" event, and this included updates to Horizontal, Vertical, InfoPane, and Appointment field pane templates. The following logic was also added to the remarkButton/clip Visibility property:

```
IF LEN(Field.FieldProperties.GetProperty('Remark').StringValue) > 0 THEN
   'Visible'
ELSE
   'Hidden'
ENDIF
```

3.3 Task Challenges re: Collecting Data in Tables with Undefined Number of Columns

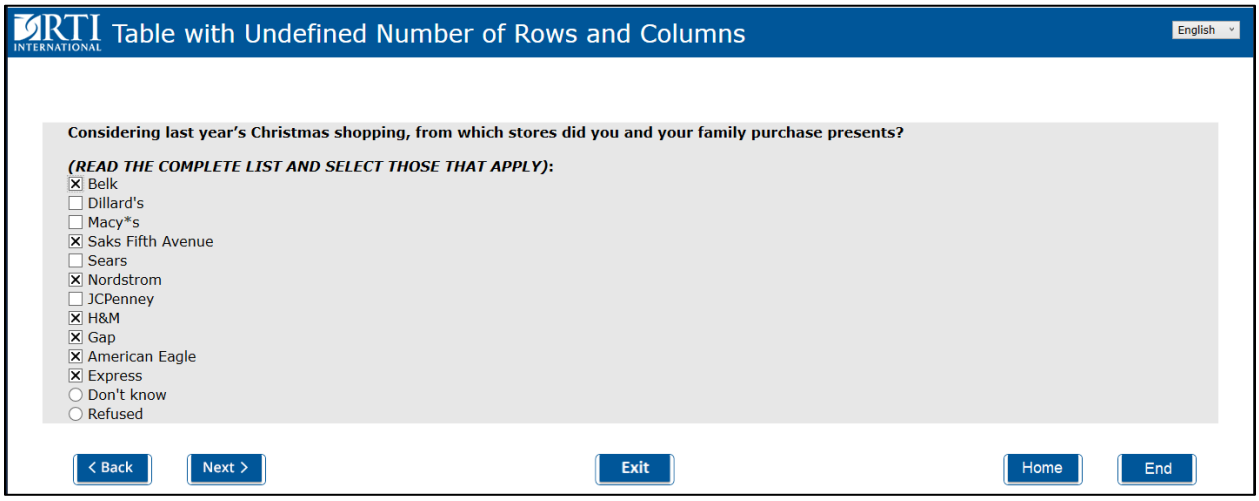
Another challenging section of one of ClientX's surveys involved collecting information for a list of household and closed family members after selecting different services used by the household. On that page of the PAPI instrument, the interviewer would write down the list of household names and then write the services selected for a given name. There could be any number of services selected, and any number of household members included from preload plus newly added members.

3.3.1 Our Solution for Collecting Data by using "OptionButtons" Table

For the corresponding service selection section in the CAPI instrument, we came up with a solution for interviewers to first collect the list of the services used by family members and then on the following screen use check boxes to specify a service for a given family member.

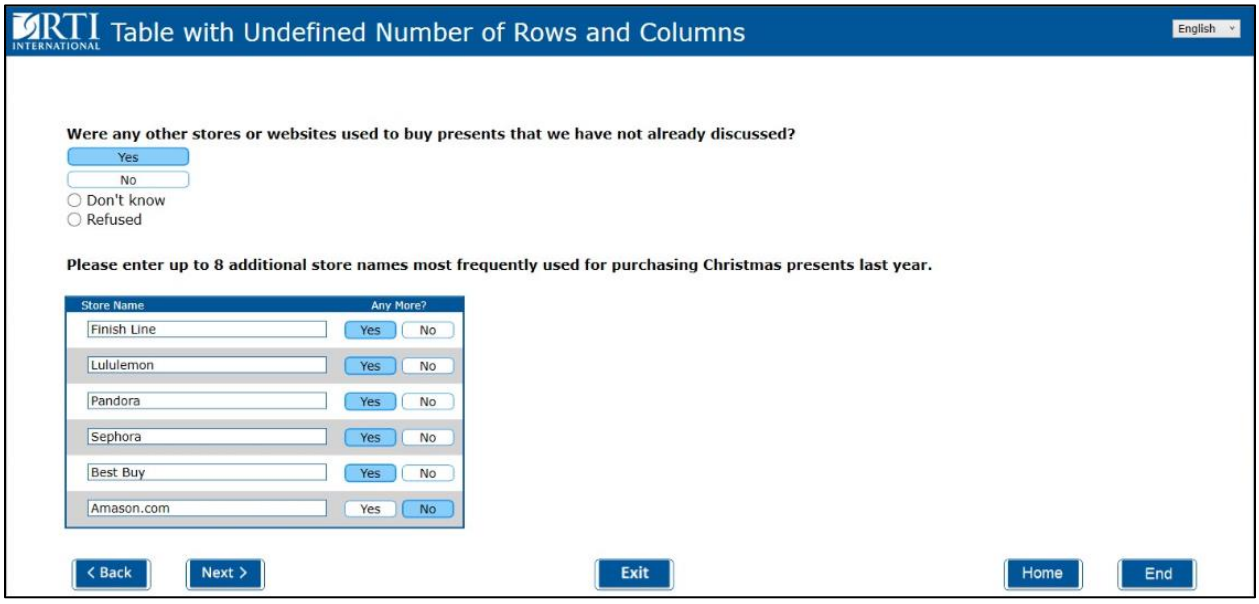
For our demo instrument, we use a predefined list of American department stores as an example of "services" with the ability to add more stores to the list.

Figure 8. Stores are selected from standard list presented to all respondents



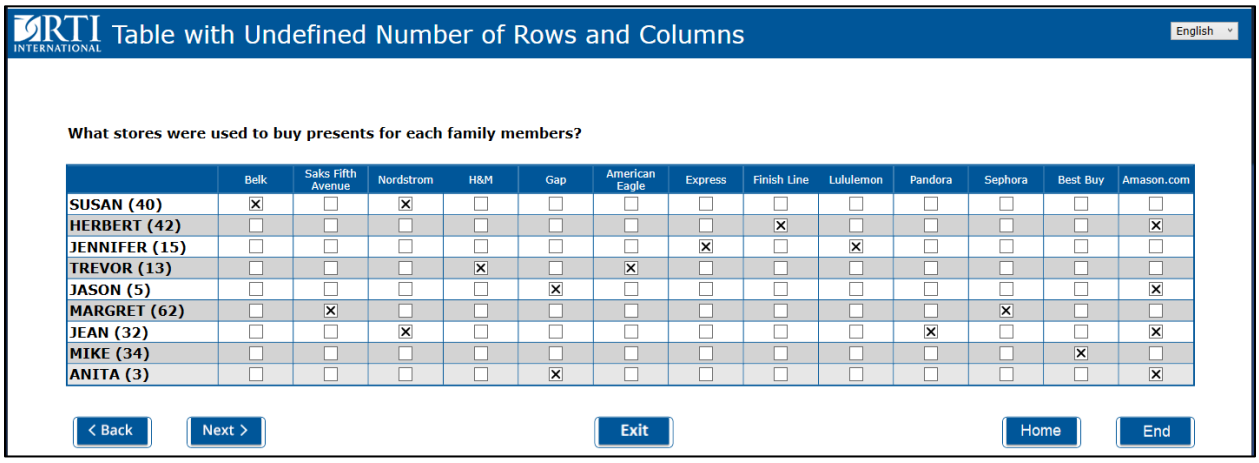
If other stores were shopped than what was initially listed, the respondent is asked to provide names of the additional stores. This CAPI interview would use a tablet, so as a rule we used push buttons for "Yes"/"No" options and present "Don't know" and "Refused" options as simple buttons.

Figure 9. Screen to capture other stores shopped



After stores are marked on the initial screen and any additional are collected after, this information is combined to generate columns for a new table where the interviewer can indicate which stores were shopped by each family member. The known family members (with age) make up the rows of the table.

Figure 10. Table to indicate stores shopped per family member



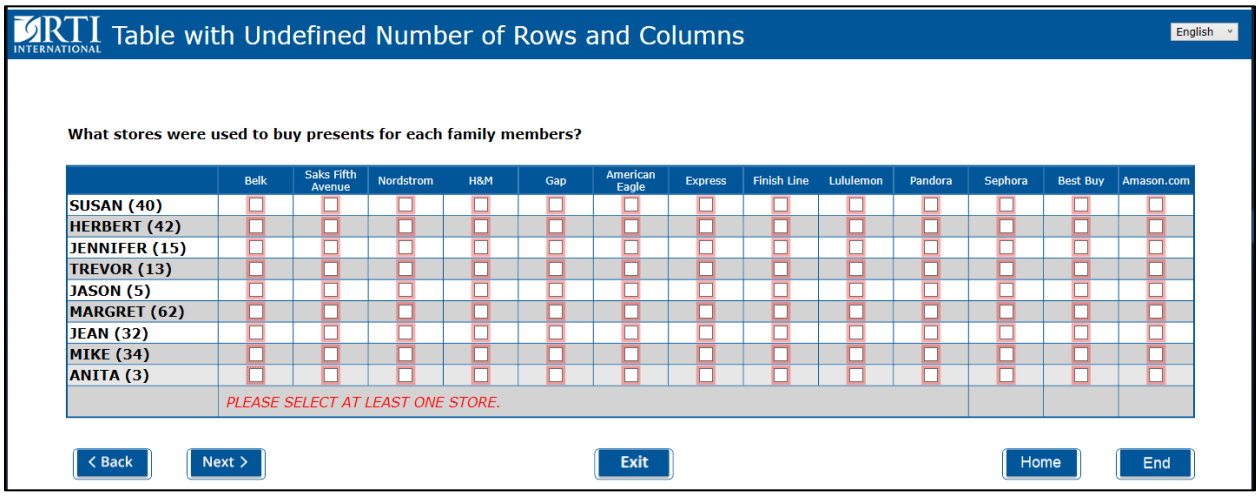
In the program, all variables for populating rows and columns in the table above are defined as arrays:

```
FillNameAge : ARRAY[1..20] OF STRING[40] {Family member Name and Age}
FillStore : ARRAY[1..20] OF STRING[30] {Name of selected or added Store}
```

Fields in the block to hold all selections are defined as below. This table will be populated only if at least one store was added to the combined list of stores, so attributes "Don't know" and "Refusal" are not allowed, but "Empty" is allowed. A check is in place to ensure at least one store is selected.

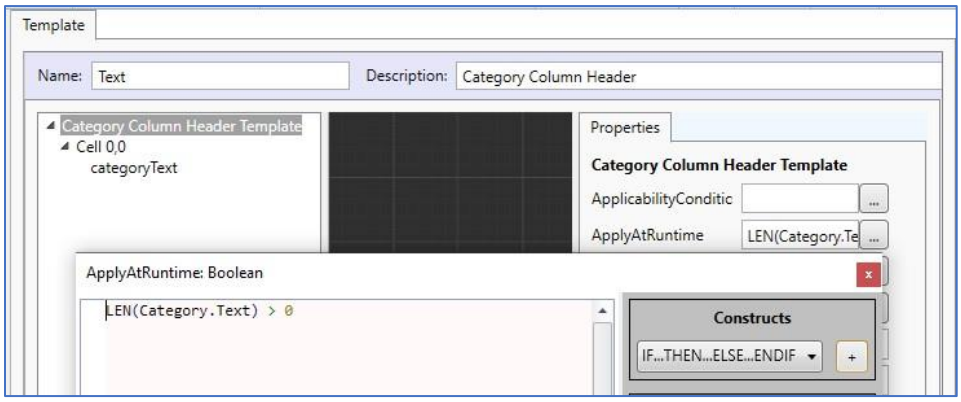
```
FIELDS //Family Members and Stores
  D1 "^{FillNameAge[1]}"/"^{FillNameAge[1]}" : SET OF tStores, NODK, NORF, EMPTY
  D2 "^{FillNameAge[2]}"/"^{FillNameAge[2]}" : SET OF tStores, NODK, NORF, EMPTY
  …etc.
```

Figure 11. At least one store must be selected



Only one change was made to the default Resource Database to "Text" of "Category Column Header Template" to show names for only needed stores to display in the column titles.

Figure 12. Category Column Header Template change was needed

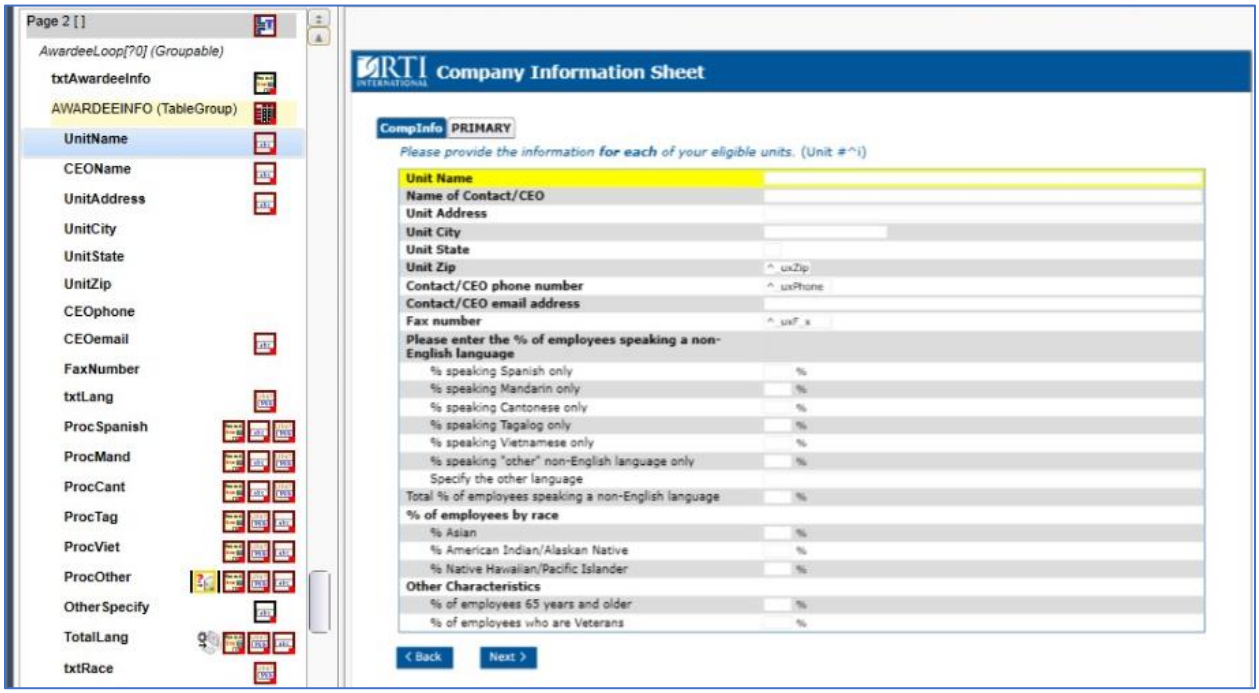


4. Use of Generic Pages with Tables

We thought it was worth mentioning that when Generic pages were introduced in Blaise 5, we attempted to use them with our demo instrument described in Section 3. In addition to collecting data in complicated tables, a few blocks were needed to collect information per household member on one page.

Unfortunately, each row in the previously defined tables was presented on a new page after setting "Generic Pages" for our demo instrument. But we did successfully use them on another project where the requirement was to collect data presented in an Excel spreadsheet for a CAWI instrument. Considering that up to 120 pages could be entered, that was great to use.

Figure 13. Use of Generic Pages with Table



5. Conclusions

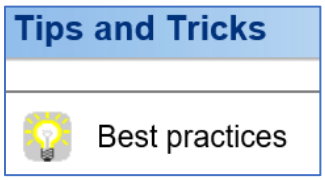
During our recent development of Blaise 5 instruments, we found that large tables can be used and generally work well, but they need extensive testing and tweaking to get them to function and appear just right. Most large tables using tablet or smaller screen size will be a challenge.

Once a layout is applied to fields within in a table, any change made to the number of fields in the table should be done carefully and without numerous changes made all at once. This tends to throw off the layout. A better way to do it is to make one change at a time, test, and then save after confirming the table layout looks good. Layout continues to be where we spend most of our time in fine-tuning instruments.

In our experience thus far, the default company Resource Database can be used quite nicely but just needs additional templates developed for specific instruments. Typically, this doesn't take much time or effort. Having a solid company Resource Database template to reuse across instruments is key.

In general, if something is not working as expected, checking online help under "Tips and Tricks" has been quite helpful. Often the solution is right there.

Figure 14. Find "Tips and Tricks" in the Blaise 5 online Help index



6. References

Blaise 5 Help Reference Manual, Statistics Netherlands
http://help.blaise.com/

Lilia Filippenko, Preethi Jayaram, Joe Nofziger, Brandon Peele, R. Suresh, "Blaise 5 with RTI's Integrated Field Management System on Field Interviewer Laptops", 2018, Proceedings of the 18th International Blaise Users Conference

R. Suresh, Keith Bajura, Matt Boyce, Emily Caron, Lilia Filippenko, Preethi Jayaram, Patty LeBaron, Joe Nofziger, Jean Robinson, Gil Rodriguez, Vorapranee Wickelgren, "Rolling Out Blaise 5 – An Interesting Journey", 2018, Proceedings of the 18th International Blaise Users Conference

Expressions in Blaise 5 Resource Editor

Nikki Brown and Yelena Beale, Westat

1. Abstract

This paper will examine ways in which the Expression Editor in Blaise 5 can be used to link specific layout objects to templates in order to create a layout instruction. Expressions created in Blaise can be very straightforward or incredibly complex, each providing powerful capabilities to the survey designer.

Expressions are created by utilizing the drop-down lists with functions and variables each editor provides in its current context. We describe our experience learning to use the Expression Editor to meet client requirements.

2. Overview of Expression Editor

"An expression is a content aware combination of explicit values, constants, variables, operators, and functions that are interpreted and evaluated at runtime to produce a value. With expressions you can change colors of active controls, set visibility of controls and many things more." - Blaise Help File

The Expression Editor in Blaise 5 is a powerful tool that enables a developer to accomplish many tasks that cannot be accomplished within the Blaise code. Expressions have been used for everything from toggling buttons off and on to security. There is information on the Expression Editor in the Blaise Help file and there are also many Expressions already written in the default templates that come with the default Resource Editor and in the Resource Editors of the samples. Reviewing the Expressions that have already been created can give the developer ideas as to what they themselves can do. Why does error text only show when there is an error? Take a look at the default Vertical template's Expression for the error grid's Visibility Property to find out. (Fig. 1)

Fig. 1. Visibility Expression from Error Grid

Visibility: Visibility

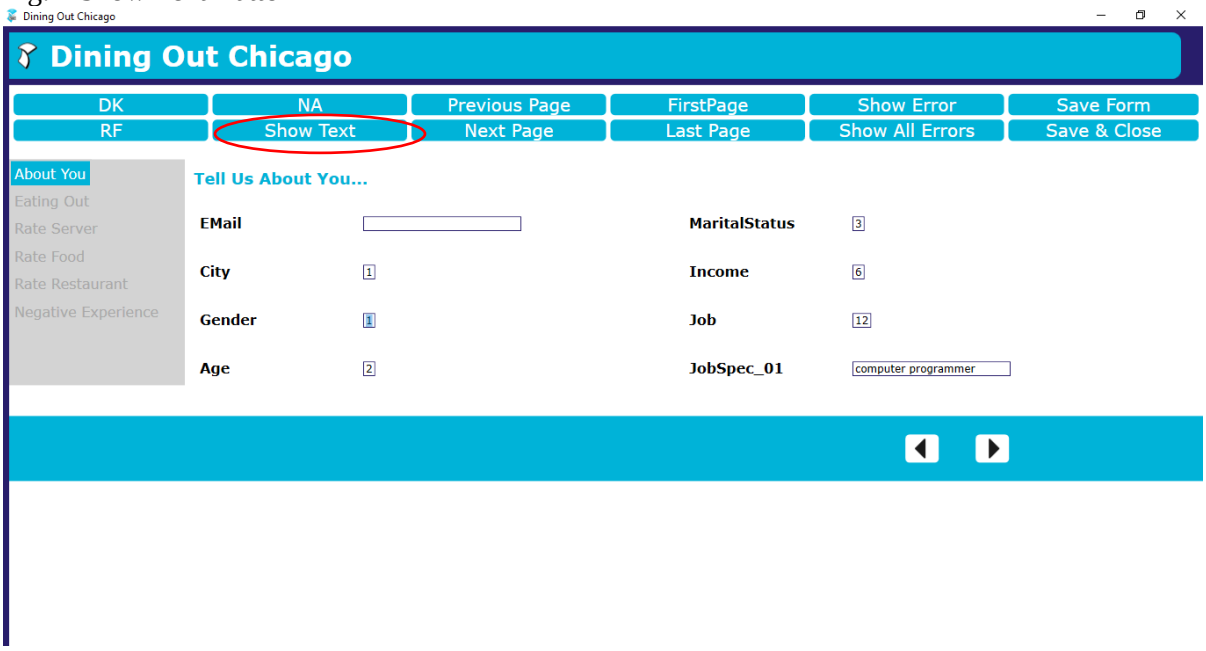
```
IF Field.Errors.IsRelevant THEN
  IF (Field.Errors.FirstRelevant.ErrorKind <> Route OR
(NOT State.RuntimeSettings.SolveErrorsImmediately)) OR
(State.PreviousPageIndex = Page.Index AND
State.NavigationDirection = Forward) THEN
    'Visible'
  ELSE
    'Collapsed'
  ENDIF
ELSE
  'Collapsed'
ENDIF
```

3. Common Usages of Expressions

For this paper we are going to examine uses of Expressions in button controls. In the OnClick property of a button control, there are a number of actions available. Some such as FirstPage, LastPage and Save need no further help to accomplish their task. Others, such as Assignfield, will require at least another step. If the Action you have selected requires more steps, the Editor will show the Properties on the right side of the screen to let you know other work is needed. For the Assignfield action, you need to tell what field is to be assigned and what value you want to assign. Remember that if the field you are assigning is in the Blaise code, you need to add it as a field reference in the Resource Editor.

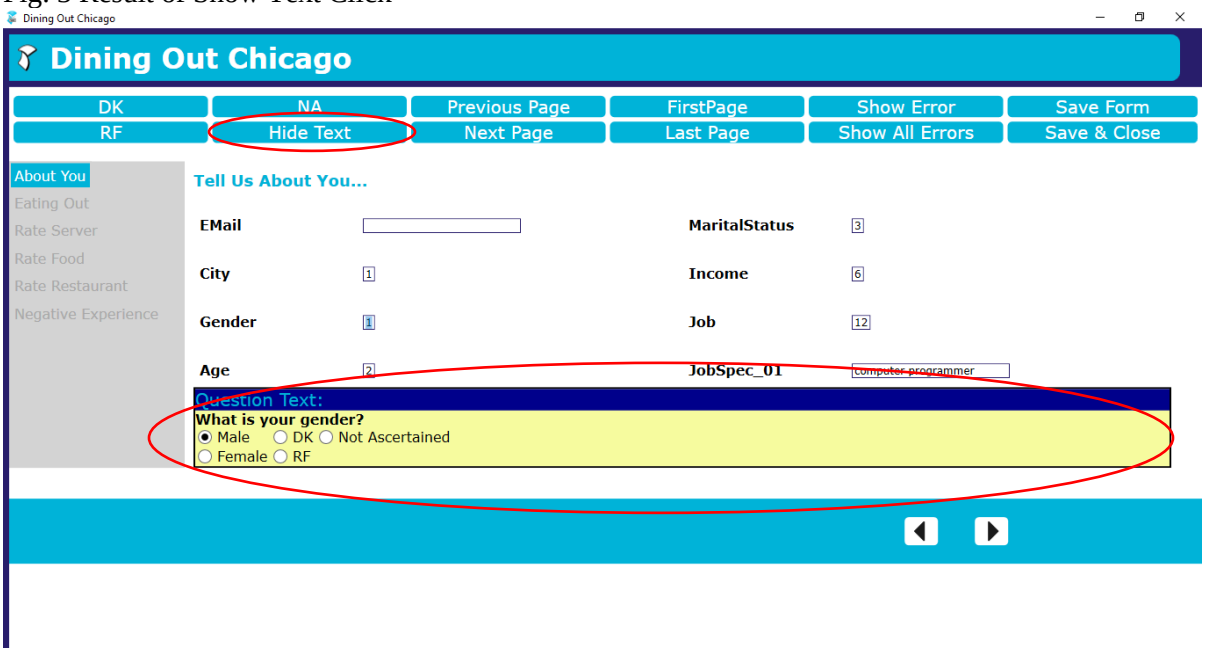
In this example, we are talking about the 'Show Text' button. (Fig. 2) To have all the pieces work in tandem, we used Expressions for both properties, (Text and Visibility) and events (OnClick). In this picture, the button has not been clicked, so no Question text is showing on the screen.

Fig. 2 Show Text Button



When the button is clicked, a couple of events take place. The text on the button changes to 'Hide Text' (Fig. 3) and the Question text and Answer categories show up at the bottom of the screen for the Active Field Gender.

Fig. 3 Result of Show Text Click



In the OnClick, event we have added a Conditional that uses Procedure Calls to change the value of the newly created Server Variable from its default of 'False' (Fig. 4) to 'True' (Fig. 5) and back again. The Server Variable needs to be set back to 'False' in the Else statement otherwise the button will not return to its initial state. (Fig. 6)

This Server Variable can then be used in other Expressions to toggle off and on, Visibility and Text.

Fig. 4 Server Variable ShowText default False

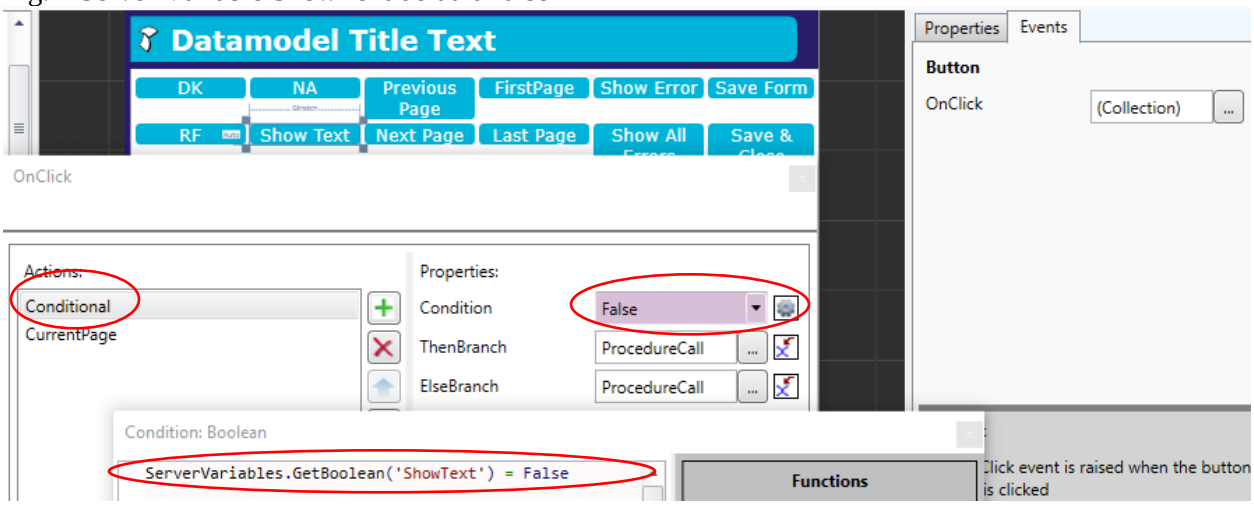


Fig. 5 Setting Server Variable to ‘True’ in the Then Branch

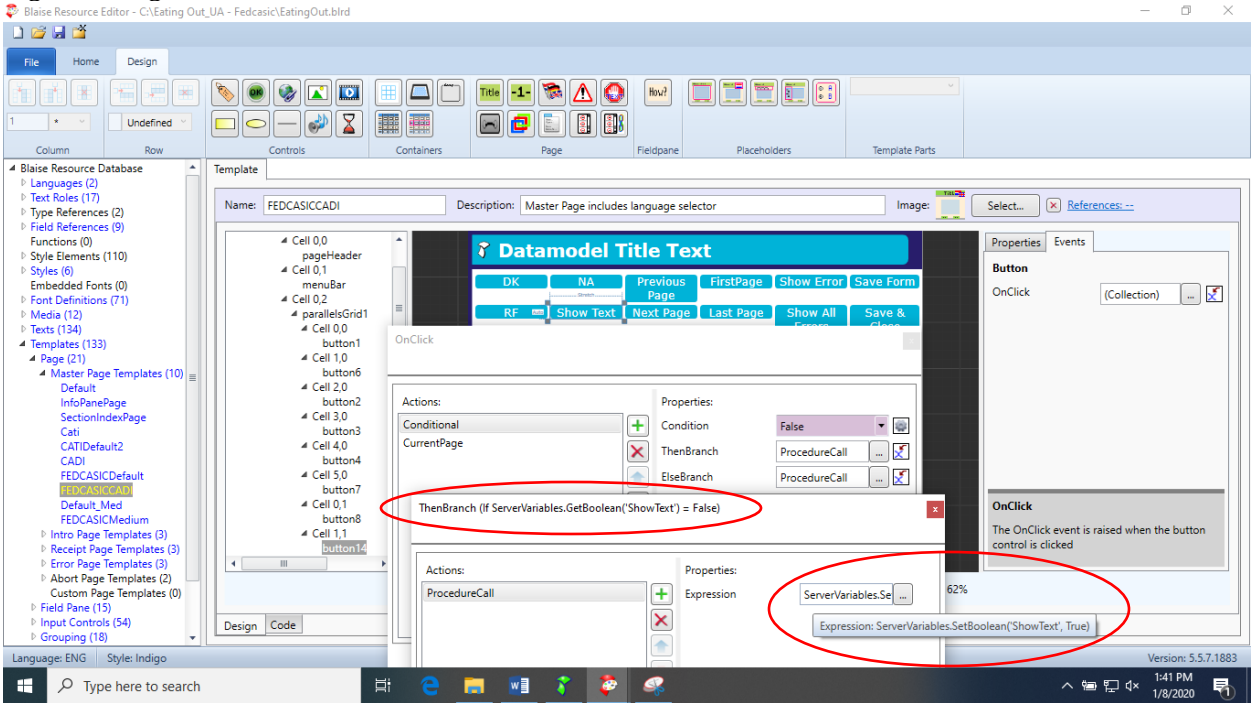
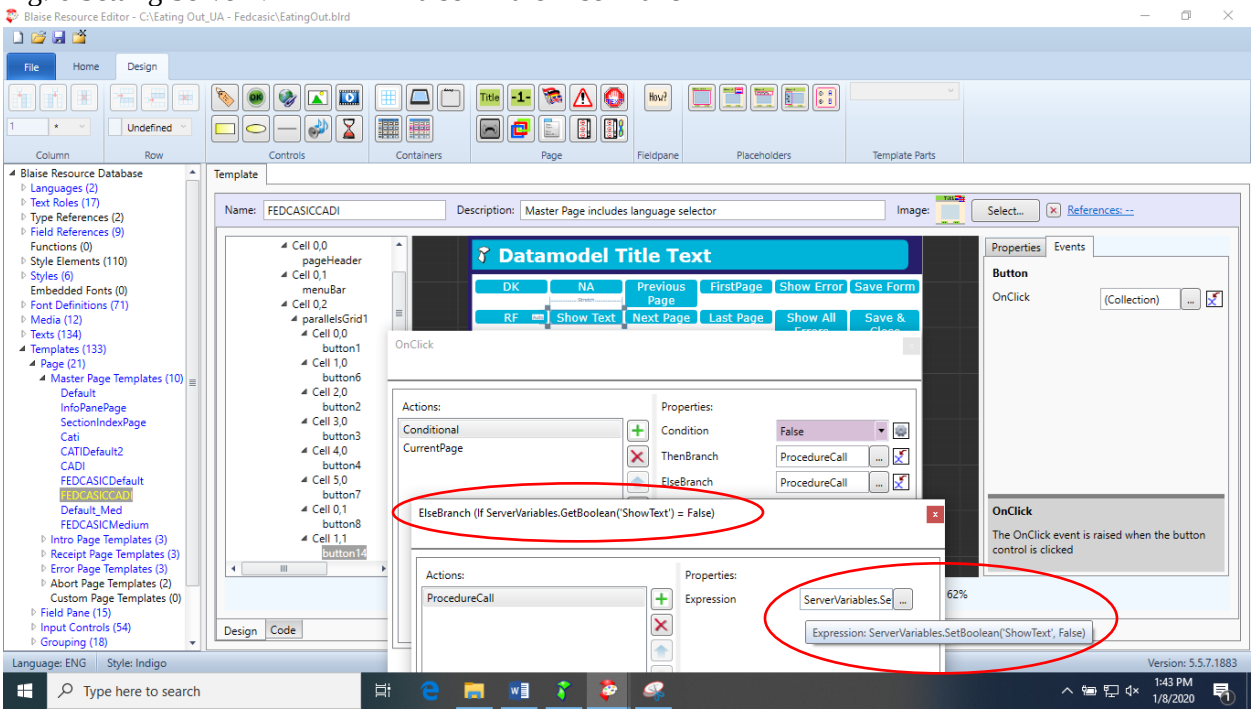
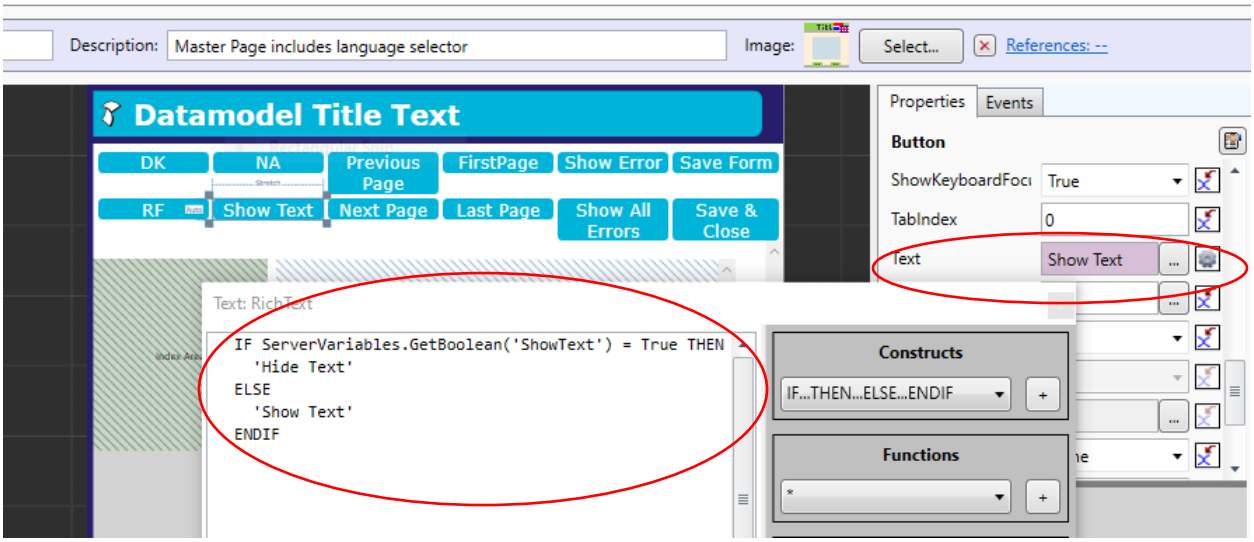


Fig. 6 Setting Server Variable to ‘False’ in the Else Branch



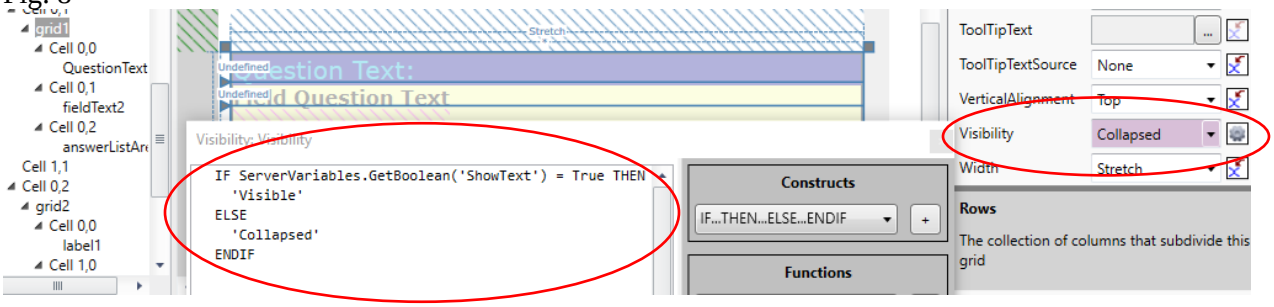
Here we refer to the Server Variable to pick which text to display on the button. (Fig. 7)

Fig. 7



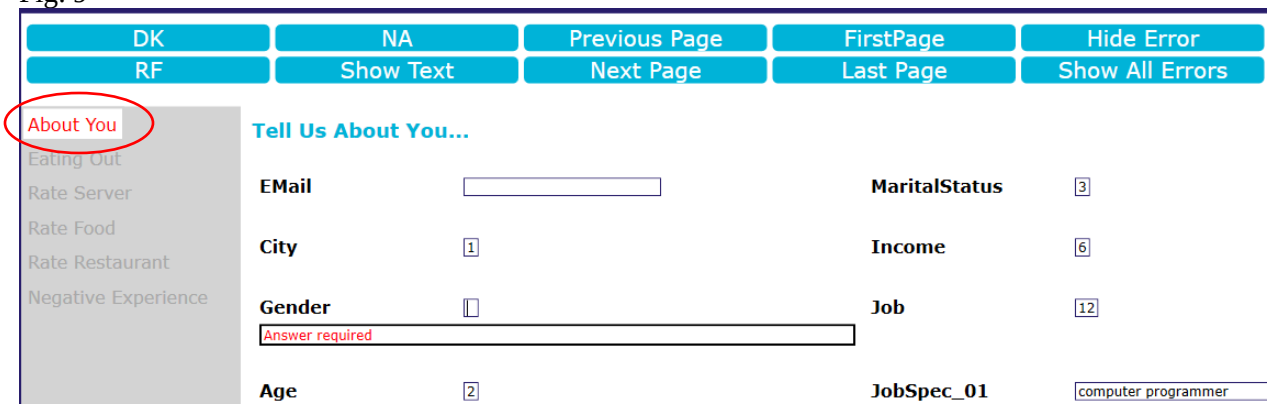
Here we refer to the Server Variable to toggle off and on the Visibility of the grid that displays the Question Text and Response Values for the active field. (Fig. 8)

Fig. 8



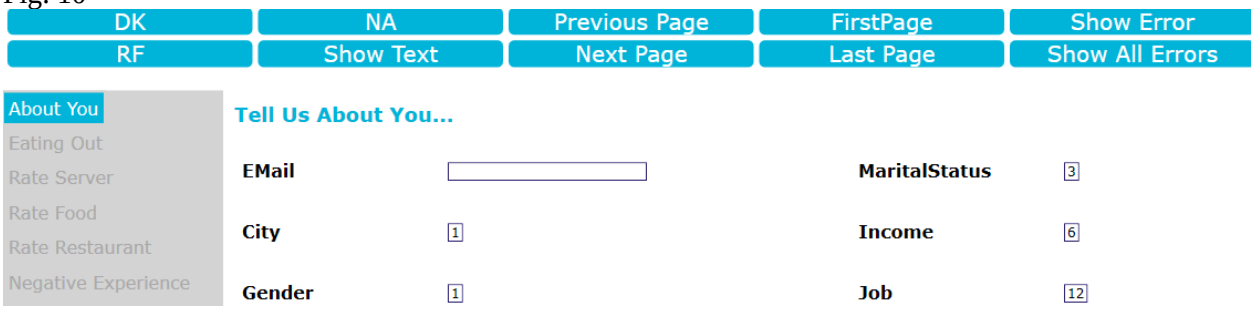
The ‘Show’ Error button used below went a step further and when clicked, changed the Font Color and Background of the Section Header in the IndexItem template. (Fig. 9)

Fig. 9



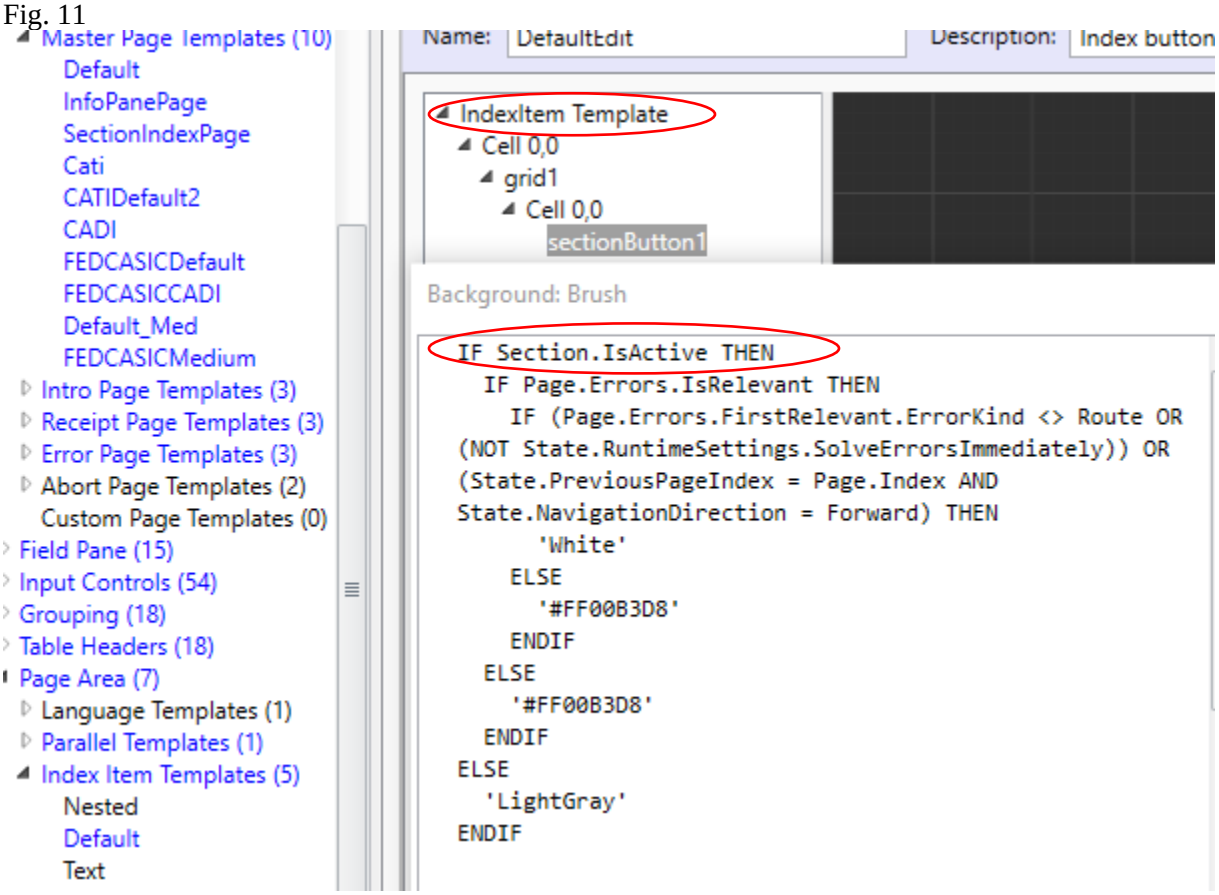
When the error was fixed by entering a value for the ActiveField Gender, all the changes were returned to their original state. (Fig. 10)

Fig. 10



In the IndexItem Template we used the same Expression from the default resource editor that controls the Visibility of an Error control for changing the Background and Font Colors of our Section Headers. We added “If Section.IsActive” to the Expression so that an error in one section wouldn’t change the colors of all sections. (Fig. 11)

Fig. 11



If this Expression had been copied to a different control, an error (Fig. 14) could occur because the controls do not have the same lists of functions and variables available to them. The Expression Editor is context aware. A button placed in the IndexItem template has a choice of Section available (Fig. 12) but in the Vertical template it does not. (Fig. 13)

Fig. 12 Index Template Section top of list

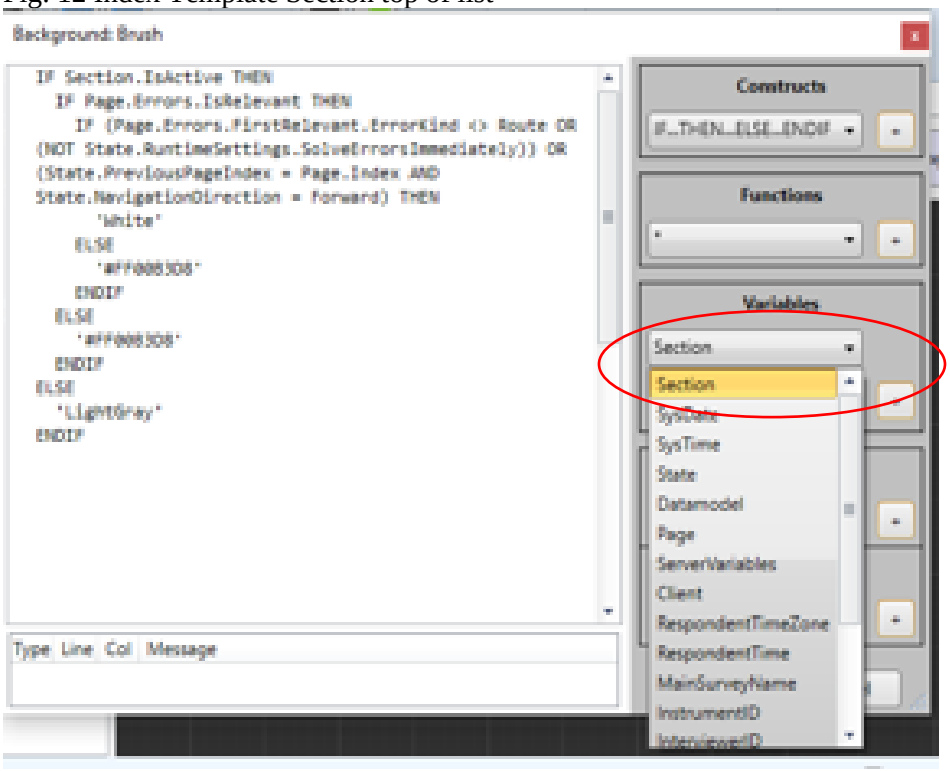


Fig. 13 Vertical Template State top of list (ie no Section)

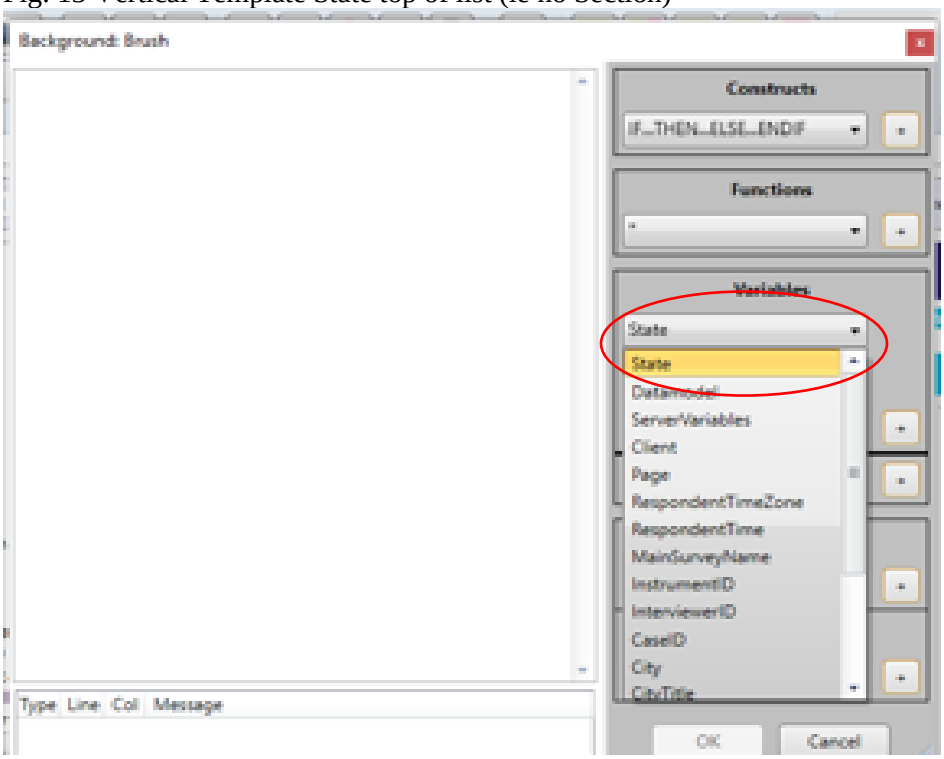
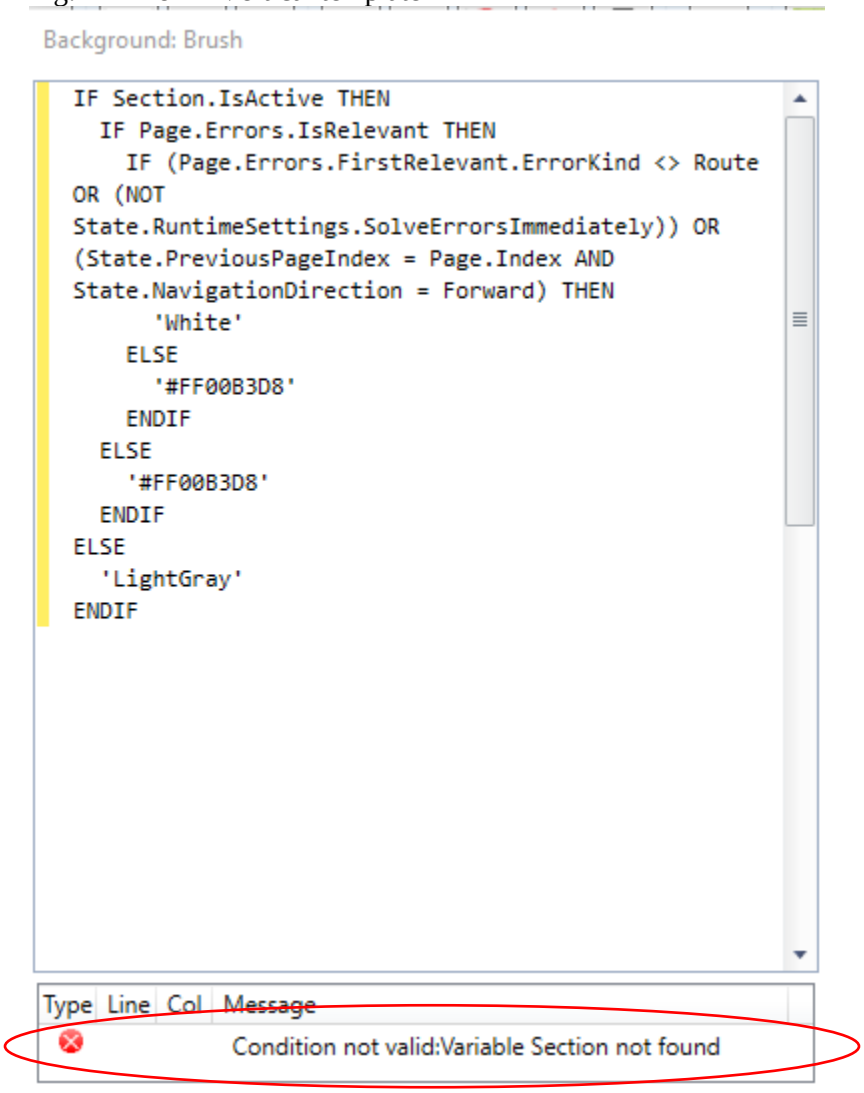
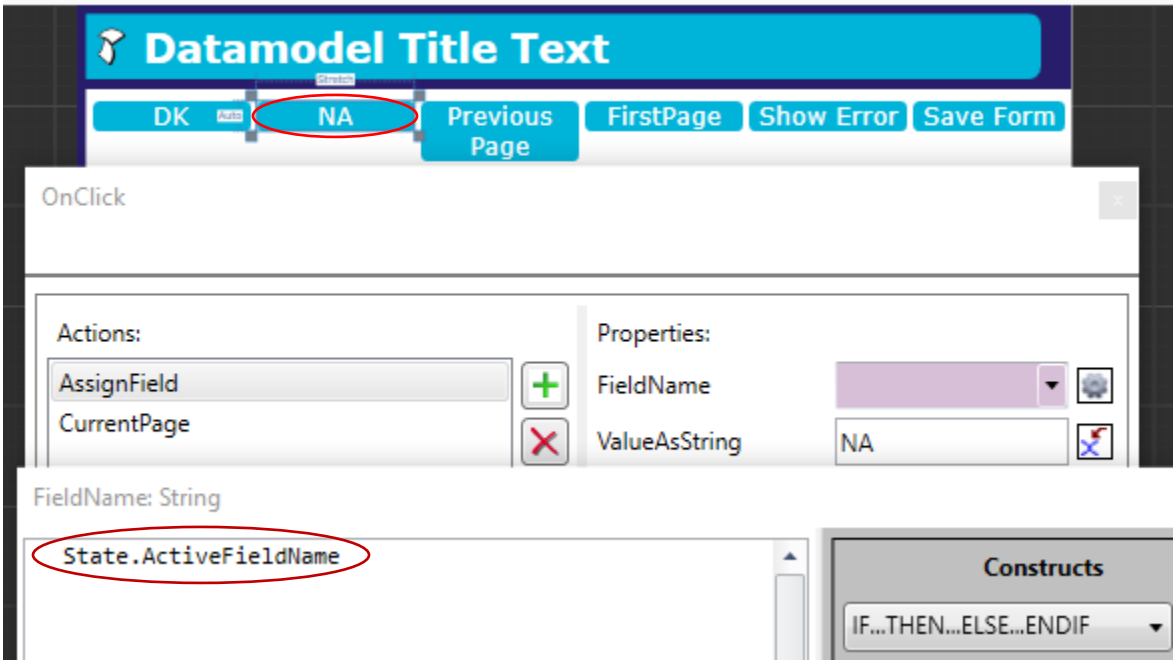


Fig. 14 Error in Vertical template



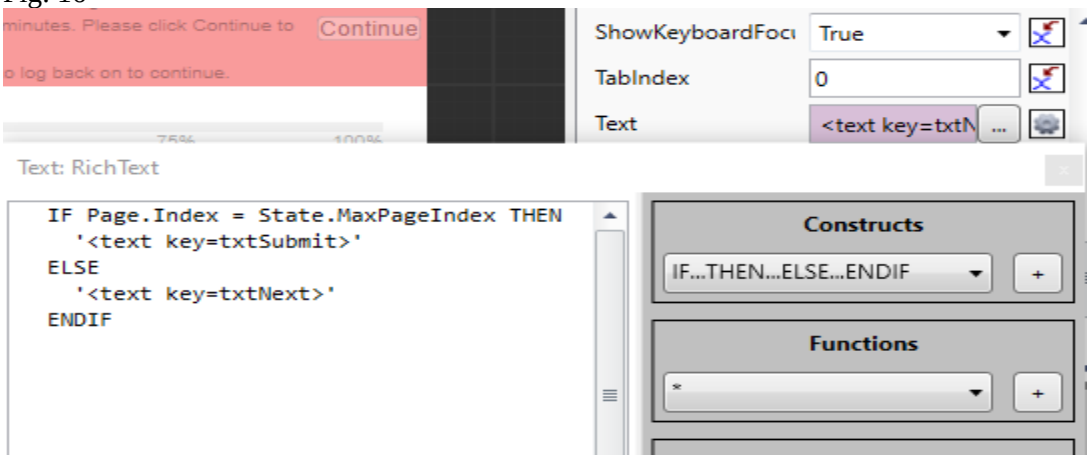
In this multimodal demo survey, the self respondent mode was allowed to leave fields empty. In editing mode most fields were required. We created a special attribute "NA" for Not Ascertained that editors can use to fill unanswered questions. The NA button uses an AssignField action. (Fig. 15) Since the field is actually on the route, it does not have to be added to the field references which would be cumbersome if all database fields needed to be added and mapped. The Expression just uses the State.ActiveFieldName for the assignment.

Fig. 15 NA attribute assignment



Another common use of an Expression is for setting the text of the Next button to Submit if on the last page of the survey. (Fig. 16)

Fig. 16



Off route fields such as statuses or time and date stamps, need to be added to the field references (Fig. 17) and mapped (Fig. 18) if values are going to be assigned to them in an expression. Remember that you have to map them if your Field Reference doesn't match the database exactly. If the database field is not at the .blax level (ie in a sub block) it won't match because it needs the block name attached to it.

Fig. 17 Field References

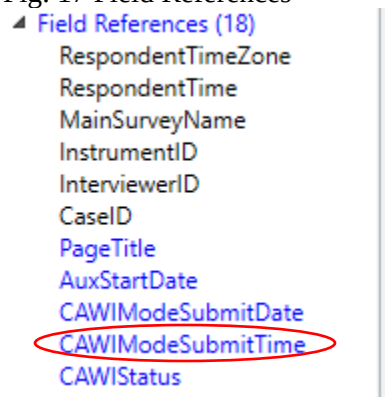
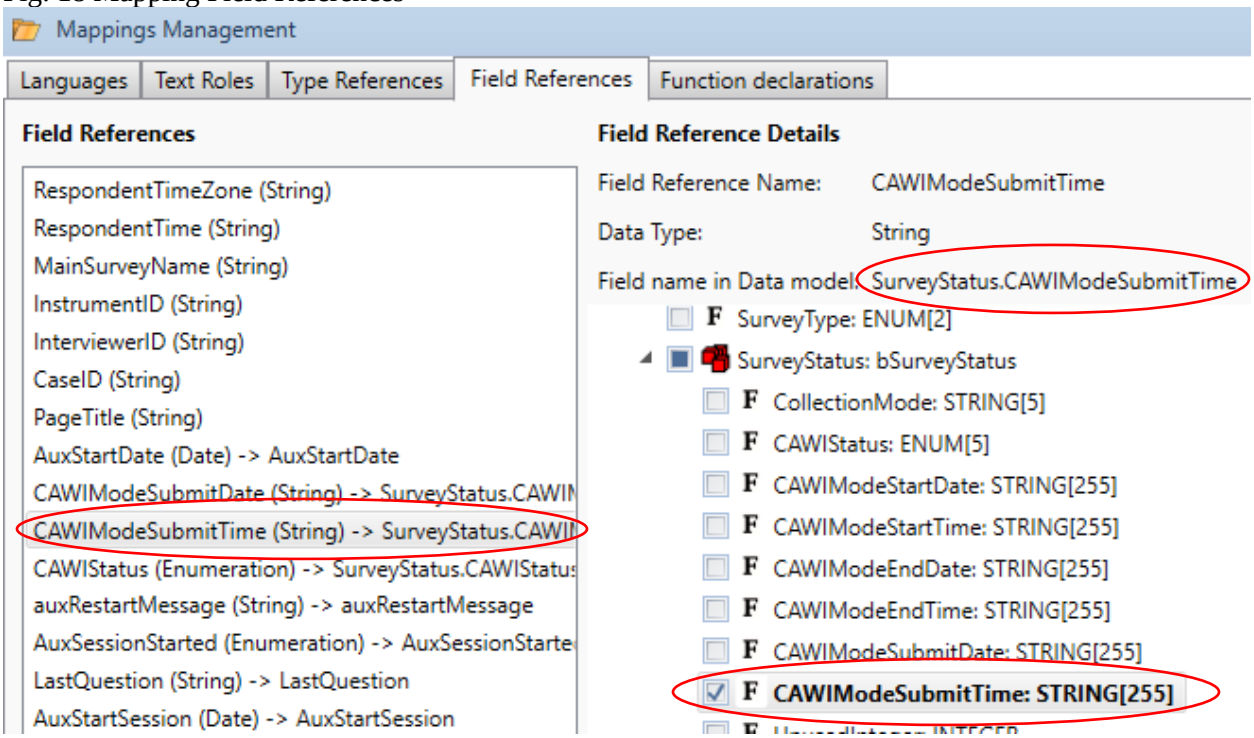
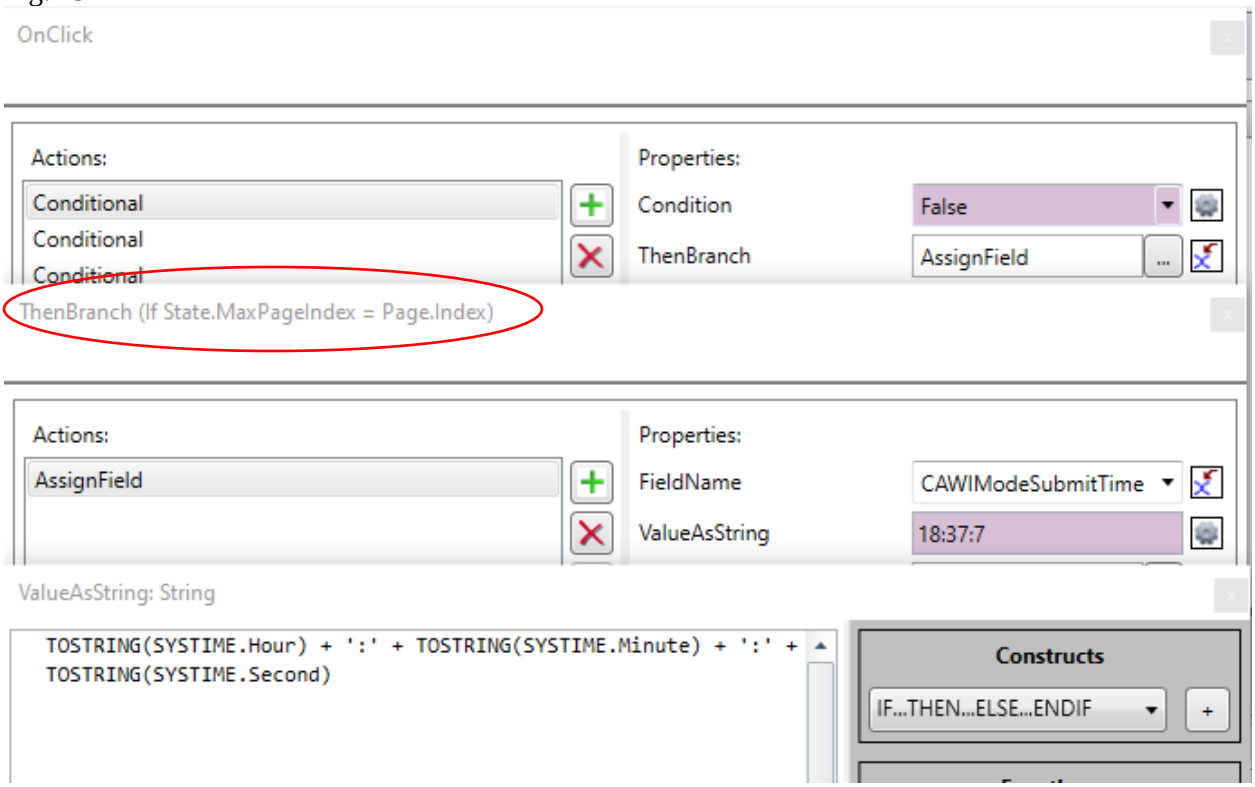


Fig. 18 Mapping Field References



After mapping the fields, you can make the assignments. In this case we assigned based on the same condition used for the Submit text. (Fig. 19)

Fig. 19



4. Conclusion

This paper examined some of the many ways that the Expression Editor can be used to support your survey using button control examples. It is a powerful context aware tool with broad capabilities that is evaluated at runtime to support survey functionality.

Even if you haven't written any yourself, your surveys are using the Expressions that come in the default templates. As we all learn in different ways, remember that there are many examples in the Resource Editors that come in the Blaise Samples and the default templates. If you do something novel with an expression, please share it with your organization so everyone can benefit from it. If you encounter this message frequently when learning to work with expressions -- "An unrecognized error has occurred" -- try building the expression one step at a time to locate the error. P.S. Don't forget to have fun with them.

Using the Resource Database to Control Web Security

G J Boris Allan and Peter Stegehuis, Westat, USA

1. Abstract

When mounting most surveys, it is common for web interviews (WIs) to be hosted by a management system (MS). The MS organizes necessary data and then runs a chosen WI. The MS is programmed in some language such as C#, in combination with HTML and similar, and we assume the WI is programmed in Blaise 5. The MS controls web data-security for the WI, and our aim is to program the WI without any need to take into account the MS and associated concerns such as web security of the WI. By adding a standardized single parallel block that does not affect WI program code, and by adding the necessary intelligence to standardized expressions in the WI resource database (the .blrd), the WI can be made secure without modifying the basic Blaise 5 program code.

2. Introduction

There are several fundamental differences between web interviewing on the one hand and CAPI or CATI interviewing on the other that have a significant impact on programming. This is true both for the (Blaise 5) interview itself and for its interaction with a management system.

In CAPI and CATI we have a program on a computer with an interviewer running the program - Blaise 5 as a Windows application – and we can control many aspects because the environment is relatively stable. The interviewer is trained and knows the instrument, special key strokes for DontKnow and Refusal answers, how to use QxQ help and deal with error messages, or how to make remarks when needed. The systems are generally secure, as they are not accessible to outside users.

However, once we deploy a survey to the web, using the Blaise 5 server manager, the extent of our control is greatly reduced:

- Respondents know how they think web pages should behave, and have their own patterns of use. As part of a general trend, mobile phones are becoming used by respondents to answer surveys, and many “web” techniques such as the use of tooltips and other features based on mouse-hover are not valid. We are all susceptible to a cavalier attitude towards any web site we use, and have short attention spans.
- Respondents can be careless (not realizing they are careless) and provides ways for sites can be hijacked by intruders who wish to profit from their intrusions.

Web security is important, but it should not have to intrude into the instrument programming process if possible. That is, the instrument code should be written to do the job of serving the interview design specification and (as much as possible) the code should not be encumbered by trying to recover from accidental or purposeful web misuse.

In the remainder of this paper we will look at a way to separate ‘security code’ from regular Blaise code by using the resource database (.blrd) and a separate management block.

3. Adding a management block

If we want disengage Blaise interview code from having to deal with instrument security, we still need to have some way of coping with events that are out of the normal sequence of affairs. Taking this a bit

further, consider a common treatment of interview break-offs, that is, situations where the interviewer hits a key combination (we tend to use CTRL-B key combination), and a new window appears not part of the interview sequence. The interviewer enters a reason for breaking off, exits the window, and the interview ends without existing data being changed. Practices will vary, but a common method is to have a break-off block in the code, distinct from the main interview – a parallel block that may never be used.

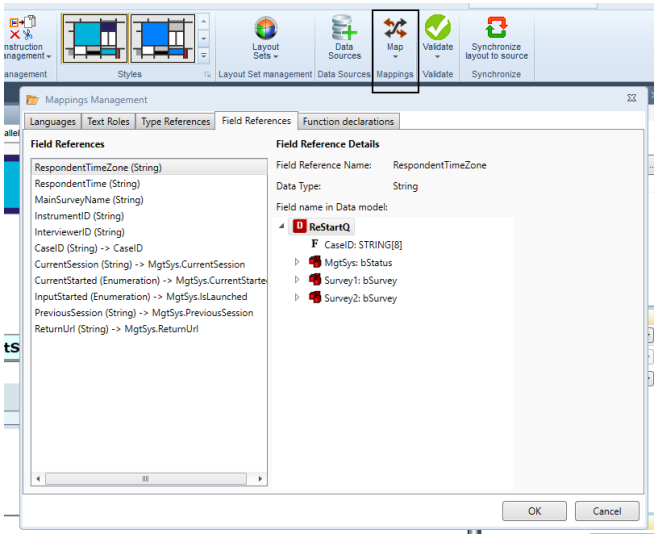
Using the break-off block cannot (usually) be left unrecorded within the management of a survey, and so a break-off status is saved to the instrument status (to be counted as part of the management process) – and saving the status is the only change to the main interview code. The instrument status might be a field in a special block for statuses, say, whether CAPI was face-to-face or whether it was by phone, and similar. We call our equivalent a "management" block, partly because some fields in the instrument management block are loaded by values generated in the survey management system – not to be confused with the Blaise survey manager. The survey management system does many things, starting with authenticating users trying to logon to the survey.

We use the management block (which is never visible to respondent) to store information we need for our security system. For example, here is such a block for one instrument (slightly reformatted):

```
BLOCK BMgtSys
  FIELDS
    // Security fields
    PreviousSession "Previous session GUID"
      : STRING
    CurrentSession "Current session GUID"
      : STRING
    CannotStart "Cannot start interview"
      : STRING[1], EMPTY
    CurrentStarted "Start this interview?"
      : (Yes,No)
    IsLaunched "Set by management system to control access to the instrument"
      : (Yes, No)
    ReturnURL "Management system will write the return URL upon launching the instrument. The instrument
        will use this URL when returning control to the management site. The instrument should not hard
        code the return URL."
      : STRING[100]
    // End of security fields
//
    // Instrument status fields
    IsReturned "Set to Yes by the instrument prior to return to the instrument (as late as possible).
        Management system consumes the Yes (flips it to No) and then performs extraction."
      : (Yes, No)
    StatusCode "Instrument Status code.  Values are instrument specific and are provided by the project."
      : STRING[2]
    EntryMode "If used, variable can be used by the instrument to behave a certain way, depending on the
        value.  D=Development;  T=Test;  P=Production."
      : STRING[1]
    LastSectionStarted "If used, stores value of the last section that has at least the first question
        answered."
      : STRING[10]
    LastSectionStartedDate "If used, stores date when LastSectionStarted being set"
      : DATETYPE
    LastSectionStartedTime "If used, stores time when LastSectionStarted being set"
      : TIMETYPE
    IsRestart "No immediate need for the variable, but would like to include in management block just in
        case."
      : (Yes, No)
    // End of status fields
ENDBLOCK//BMgtSys
```

In general, status fields are assigned in the Blaise code[1] and stored in the case data, whereas values for fields associated with security (such as PreviousSession), though not assigned in the Blaise code, are still stored in the case data because security-field values are assigned by program expressions in the resource database. The intelligence behind web security is contained in the resource database (a blrd file) because the resource database is capable of far more than designing appropriate layout, or defining what happens when you select a certain button.

Controlling what happens is a result of programming expressions. The connection between expression programming objects (known as field references in the resource database) and corresponding Blaise fields in the current interview session (if necessary) is provided through a mechanism known as mapping. An easy way to look at what has been mapped is by viewing the mapping connector in the Blaise control centre for a simple test instrument (`ReStartQ`) – the Map/Mappings tab is highlighted by the rectangle:



As you can see the name of Blaise field in the management block (`MgtSys.IsLaunched`) and the name of the equivalent field reference (`InputStarted`) in the resource database need not be the same (`InputStarted -> MgtSys.IsLaunched`). This means we can change the Blaise field names, remap. If a hacker found that field X was related to resource Y they might try to change values for field X, but the next study could have a new mapping – field W (not field Y) could now be related to resource Y and so previously hacked knowledge would be of no use.

3.1 The blax.layout file

In general, what can be more useful to get a feel for what is happening, is for us to look at the content of the *<Instrument>*`.blax.layout` file, an XML file that makes explicit what might be hidden in the work that goes into establishing layouts for an instrument. If we look at the content of the `RestartQ.blax.layout` file,

[1]
```
IF SCR.SCR111b <> EMPTY THEN
    IF ACTIVELANGUAGE = Eng THEN MgtSys.StatusCode := '35' ELSE MgtSys.StatusCode := '36' ENDIF
  ELSEIF (SCR.Box5Route = 'SCR095') AND ((SCR.SCR120 = '1') OR (SCR.SCR120 = SK)) THEN
...
```

and – in particular – how the `LayoutSpecFieldReference` items are listed to match the visual presentation of the mappings, and – in general – how layouts are listed in an understandable pattern. Here is a snippet:

```
  <LayoutSpecFieldReferences xmlns="layoutspec">
    <LayoutSpecFieldReference Name="RespondentTimeZone" />
    <LayoutSpecFieldReference Name="RespondentTime" />
    <LayoutSpecFieldReference Name="MainSurveyName" />
    <LayoutSpecFieldReference Name="InstrumentID" />
    <LayoutSpecFieldReference Name="InterviewerID" />
    <LayoutSpecFieldReference Name="CaseID" MappedDatamodelField="CaseID" />
    <LayoutSpecFieldReference Name="CurrentSession" MappedDatamodelField="MgtSys.CurrentSession" />
    <LayoutSpecFieldReference Name="CurrentStarted" MappedDatamodelField="MgtSys.CurrentStarted" />
    <LayoutSpecFieldReference Name="InputStarted" MappedDatamodelField="MgtSys.IsLaunched" />
    <LayoutSpecFieldReference Name="PreviousSession" MappedDatamodelField="MgtSys.PreviousSession" />
    <LayoutSpecFieldReference Name="ReturnUrl" MappedDatamodelField="MgtSys.ReturnUrl" />
  </LayoutSpecFieldReferences>
</LayoutSpecification>
```

Note that the defined field references are the same in the mappings display as they are given as the items labelled as LayoutSpecFieldReference.

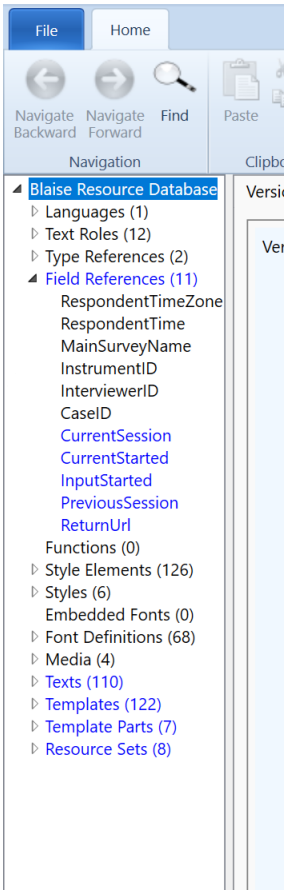
A full listing of the blax.layout is in the Appendix.

3.2 Resource databases and field references

We have mapped some instrument fields to items in the resource database, but what are those items? Those items are field references in the resource database. As we have seen, there are variables (field references) in the resource database that take their initial value from a mapped field in the instrument, and – if the value of the field reference changes – then that change is reflected in the value of the mapped field in the instrument.

When we open the resource database, the field references are listed. Some field references are standard, and appear by default – quite often we have no use for them, and we do not map them to instrument fields. We can add our own custom field references, and for our security system the references are mapped to instrument fields in our management block.

Here are the standard field references (RespondentTimeZone – CaseID) and the added security references (CurrentSession – ReturnUrl) contained in a model security resource database:



The principal places these field references are used are the `OnLoad` events for different master page templates (less frequently, `OnLoaded`).

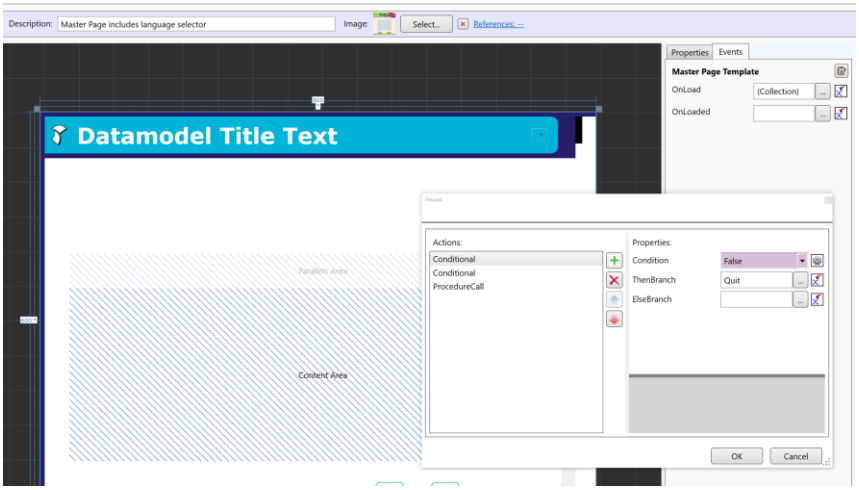
4. Loading web pages

A key idea in web-site design is the idea of distinct web pages. Apart from the case where each successive page overwrites the previous page, web pages can be displayed in new windows (different instances of a browser) or in new tabs (different pages within the same browser instance). One security concern is that we do not pass information in a URL that could be used to create a second instance of an interview in a new tab, or different browser instance.

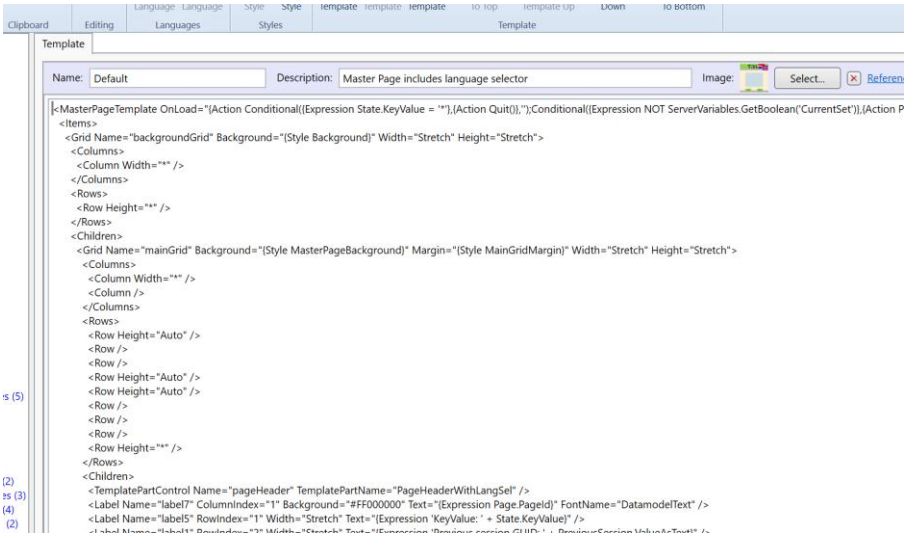
In Blaise 5 there are certain fixed patterns in which information is presented as a page. These fixed patterns are known as master-page templates, that is, designs of how different classes of web page should look and behave. Each different type of master-page template will have different looks and/or behaviors. What we do is attach compound expression code to the `OnLoad` event of each relevant master-page template. In fact we use the same `OnLoad` code for each master page, because we can never be sure where an interview might end. We have to assume that an interview can start/restart at any page.

However, once we have made our security checks, we do want to have a check at every page, so we have to have a mechanism that checks once only per session and doesn't waste time checking again until the interview is restarted.

Here is a master page layout with an expanded `OnLoad` event:



If we change the view of this master page to show the code, we see:



It is difficult to read the `OnLoad` expression above, and so I have reformatted the code to show the structure of the expression in a clearer way.

5. The `OnLoad` expression code

If you look at the master page layout with an expanded `OnLoad` event (above), there are two conditionals followed by a procedure call. This is the first conditional:

```
<MasterPageTemplate OnLoad="
  {Action
    Conditional({Expression State.KeyValue = '*'},
      {
        Action Quit()
      },
      ''
    );
```

That is, if the interview key (case ID) is `*` then quit. This conditional is relevant for when an interview is timed out. We create a case with primary key '*' for the sole purpose of sending control back to the management system in case of a session timeout. The `*` case is created before interviews start as an empty record with only one field (`MgtSys.ReturnURL`) having a value – which sends control back to the management system.

In addition to field references, the expression language knows about server variables, where a better name might be "session" variables. We must provide the name and type of any server variable we use, and when a server session starts the initial value of a server variable is a default. We define a Boolean server variable (`CurrentSet`) and, in the second conditional, if the Boolean value of `CurrentSet` is false (the initial value of `CurrentSet`), we proceed with making assignments to field references, otherwise we skip and do nothing more for the `OnLoad` event.

Later we assign the Boolean true to `CurrentSet`, and so all the expressions under this conditional are skipped.

```
    Conditional({Expression NOT ServerVariables.GetBoolean('CurrentSet')},
      {
        Action ProcedureCall({Expression ServerVariables.SetString('StopReason', 'TimeOut')});
        AssignField('CurrentStarted', {Expression InputStarted.ValueAsText}, '');
        AssignField('InputStarted', '2', '');
        Save();
```

We initialize the server variable `StopReason` to "Timeout" as a default. We don't use `StopReason` for anything at the moment, it is just available in case we make enhancements.

If you look at the mappings for the field reference `InputStarted` then it refers to the field `MgtSys.IsLaunched` (set directly by the management system by means of the API). If `IsLaunched` is true (1) then the instrument has a valid start, and any other value is an invalid start. For security reasons we might want at some point to change the name from `IsLaunched`, and we need only change the mapping.

The value of `InputStarted` is copied to `CurrentStarted`, and `InputStarted` is set to 2. This means that, if the case is restarted someway outside the management system, then the value 2 is copied to `CurrentStarted`. `CurrentStarted` is mapped to `MgtSys.CurrentStarted`. If `CurrentStarted` is 2, then we quit because `MgtSys.IsLaunched` was not true:

```
        Conditional({Expression CurrentStarted.ValueAsText = '2'},
          {
            Action ProcedureCall({Expression ServerVariables.SetString('StopReason', 'NotLaunched')});
            Quit()
          },
          ''
        );
```

We have two field references that refer to session IDs (GUIDs), and we assign what was the previous current session to a field reference `PreviousSession`. We then look at the current session GUID, and assign that to the field reference `CurrentSession`. If a session was broken off without any Blaise knowledge (such as an X-out) then the session could be still alive, and so if the previous session has the same GUID as the current session, something is wrong, and we quit.

```
        AssignField('PreviousSession', {Expression CurrentSession.ValueAsText}, '');
        AssignField('CurrentSession', {Expression State.RuleSessionId}, '');
        Conditional({Expression CurrentSession.ValueAsText = PreviousSession.ValueAsText},
          {
            Action ProcedureCall({Expression ServerVariables.SetString('StopReason',
        'SessionNotClosed')});
            Quit()
          },
          ''
        );
```

Finally, we set the `CurrentSet` server variable to true, so that we don't go through the whole set of assignments again, we have a (gratuitous) check on whether `CurrentStarted` is yes, and we have a final assignment of the exit URL to a server variable (`URL`).

```
        ProcedureCall({Expression ServerVariables.SetBoolean('CurrentSet',
            NOT ServerVariables.GetBoolean('CurrentSet'))});
        Conditional({Expression CurrentStarted.ValueAsText = 'No'},
          {Action Quit()},
          ''
        )
      },
      ''
    );
    ProcedureCall({Expression ServerVariables.SetString('URL', ExitURL.ValueAsText)})
  }
">
```

You will have noticed that many of our checks end with something like `{Action Quit()}`, and that implies a knowledge of how an interview ends and relates to the main Blaise server. This is part of a larger task, discussed in the paper on session timeouts. [2]

6. Conclusion

Security is a main concern, more so even with web interviewing than it already is with other modes of interviewing. Guarding against unwanted attempts for intrusion in our networks is a prime concern, both for any direct results of hacking as well as damage to reputation, which in turn can lead to a diminished willingness by respondents to engage in future surveys. On top of that, dealing with time-outs and unintended X-out by online respondents, in a way that is both safe and user-friendly is an important issue we have been trying to deal with.

The approach described in this paper is a step in an ongoing process to deal with these new realities.

[2] See "Using the Resource Database to adapt to session timeouts", G J Boris Allan, Joseph Allen, and Siu Wan (IBUC 2020)

7. Appendix: blax.layout

```
<?xml version="1.0" encoding="utf-8"?>
<LayoutSpecification xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Name="ReStartQ" Version="10"
        GenerateAllSections="true" GenerateClientRules="true" UseGenericPages="true">
  <Languages>
    <Language Name="" />
  </Languages>
  <LayoutSets>
    <LayoutSet Name="Interviewing1" ResourceSetName="Large" StyleName="Indigo" RequiredStyle="false"
        DesignHeight="600" DesignWidth="800" ResizeMode="Auto" RouteItemsPerPage="4" RowsPerTable="999"
        ReceiptPageName="Default" IntroPageName="Default" AbortPageName="Default">
      <InstanceLayoutInstructions>
        <RouteItemLayoutInstructions RouteItemName="Survey1">
          <Instructions>
            <NewPageInstruction Locator="Before" />
            <GridInstruction Locator="Before" RouteItemsPerPage="1" />
          </Instructions>
        </RouteItemLayoutInstructions>
      </InstanceLayoutInstructions>
      <Parallels>
        <Parallel Name="">
          <IsGeneric>false</IsGeneric>
        </Parallel>
        <Parallel Name="PRIMARY">
          <IsGeneric>false</IsGeneric>
        </Parallel>
        <Parallel Name="MgtSys">
          <IsGeneric>false</IsGeneric>
        </Parallel>
      </Parallels>
    </LayoutSet>
  </LayoutSets>
  <LayoutSetGroups>
    <LayoutSetGroup Name="Interviewing" DataEntrySettingsName="StrictInterviewing">
      <LayoutSetNames>
        <string>Interviewing1</string>
      </LayoutSetNames>
    </LayoutSetGroup>
  </LayoutSetGroups>
  <RoleReferences>
    <RoleReference Name="Help" />
    <RoleReference Name="Watermark" />
    <RoleReference Name="ToolTip" />
    <RoleReference Name="SpecialAnswer" />
    <RoleReference Name="CategoryGroup" />
    <RoleReference Name="EditMask" />
  </RoleReferences>
  <TypeReferences>
    <TypeReference Name="THeader" />
    <TypeReference Name="TCurrency" />
  </TypeReferences>
  <LayoutSpecFieldReferences xmlns="layoutspec">
    <LayoutSpecFieldReference Name="RespondentTimeZone" />
    <LayoutSpecFieldReference Name="RespondentTime" />
    <LayoutSpecFieldReference Name="MainSurveyName" />
    <LayoutSpecFieldReference Name="InstrumentID" />
    <LayoutSpecFieldReference Name="InterviewerID" />
    <LayoutSpecFieldReference Name="CaseID" MappedDatamodelField="CaseID" />
    <LayoutSpecFieldReference Name="CurrentSession" MappedDatamodelField="MgtSys.CurrentSession" />
    <LayoutSpecFieldReference Name="CurrentStarted" MappedDatamodelField="MgtSys.CurrentStarted" />
    <LayoutSpecFieldReference Name="InputStarted" MappedDatamodelField="MgtSys.IsLaunched" />
    <LayoutSpecFieldReference Name="PreviousSession" MappedDatamodelField="MgtSys.PreviousSession" />
    <LayoutSpecFieldReference Name="ReturnUrl" MappedDatamodelField="MgtSys.ReturnUrl" />
  </LayoutSpecFieldReferences>
</LayoutSpecification>
```

Blaise 5 Scaling Experience – A Case Study

Mangal Subramanian, Kathleen O Reagan, Arthur Menis, and Ray Snowden, Westat

1. Abstract

This paper describes Westat's experiences in implementing a large scale web survey using Blaise 5, focusing on the load testing and scaling part of the project. We also discuss various aspects of Blaise 5 as it relates to programming, server configuration, issues faced etc. The paper also touches on some alternative approaches to hosting Blaise 5 for such large scale projects, contrasting it to the one Westat used for the final implementation.

2. Project Background

Westat conducted two simultaneous surveys (Main and Mode), using a common screener and separate extended surveys for eligible respondents, to provide a broad overview of the characteristics, attitudes, and experiences of two groups of respondents. The Main study was done entirely in WEB and a percentage of the Mode study was done in CATI and the remainder was done in WEB using the Main study's instrument. The mode study only allowed for only one of the two extended questionnaires.

Westat's role was focused on sampling, data collection, and weighting. Respondents were encouraged to complete the online screener and, if eligible, the appropriate online extended survey. The screener was offered in English plus two additional languages. Those who failed to respond were followed up with additional mailings that included a hard-copy instrument.

3. Blaise Instrument Characteristics

The survey instrument was programmed in Blaise 5 (version 5.6.5.2055) as one instrument instead of 3 separate instruments in order to facilitate security measures which would allow respondents to go directly from the screener into a separate extended instrument without having to pass thru the management system repeatedly.

There were over 20 randomized questions programmed either by question order or category order in both extended instruments and 4 randomized question options in the screener. As we were using Blaise 5 version 5.6.5 we did not have the advantage of the randomization functions included in Blaise 5.7 so we randomized the "old fashioned way" with multiple arrays and voluminous code.

The client insisted on multiple questions on the same page but with individual templates which made movement from screen to screen sometimes noticeable. The survey instrument also included preloaded sampling random numbers and required sampling done in the screener.

4. Blaise web application performance requirements

The total sample size for the project was over 350,000 and the initial plan was to send out all the mailers at the same time. So the estimates given by the project to the systems team were as follows:

- Initial estimate of 2,500 users per day accessing the survey,

- Approximate estimate of 400 concurrent users for the screener,
- 10% of the sample expected to qualify for the extended (ball park estimate of 30-40 concurrent users for the extended), and
- Acceptable page load time <= 3 seconds.

5. Application set-up

The Blaise survey was managed by our in-house web survey management system. The role of the management system was to validate a pin entered by the user and launch the survey. At the completion of the survey the completion status was captured back into the management system.

Web security was an important requirement of the research. To ensure that the Blaise web session was secure from 'URL' hijacking or impersonation, the management system created a security token during launch and inserted it into a SQL table. When the Blaise instrument started, the token was checked to see if it was a valid one and the user was allowed to proceed only if it was valid.

6. Load Testing Process

The load testing tool used was the Visual Studio Test Studio combined with the Azure Load testing service, which enables large number of users to be simulated through its cloud based tools, eliminating the need to set up and configure agents on premise.

6.1 Challenges with the Visual Studio Tool

We set up a couple of test scripts and ran them with a few users to validate that the tool is able to produce clean completes. Review of the Blaise DB results showed records were created, but no data were present that would have been expected based on the test scripts. We contacted the Blaise developers about this and they reported that the VS web test script worked with only the primary key as a script parameter. Blaise also has other data that changes from case to case; in particular the sessionID was most important, but there were others as well. The Blaise developers indicated that custom code would be needed to dynamically extract the sessionID that Blaise had generated on the server and substitute this value into later script requests to the server.

The Blaise developers offered a tool they used, but it did not work in our environment. So we reviewed our options and decided that the handling of a dynamic parameter such as sessionID was too involved to address within our limited schedule for setting up and testing a server configuration.

6.2 Implementing Load Testing

To mitigate the issue with load test tool, we decided to use a low-tech and less risky option – Manual Testing. We asked staff throughout Westat to access a test survey site and complete 5-6 user sessions in a span of 30 minutes. The final load test plan included the following elements:

- Over 70 staff completed the testing process;
- Each tester was provided with 5-6 unique URL's and asked to complete at least 3 sessions;
- Test scripts identified responses so that the routes chosen would be completed within the test period;
- The management application was eliminated from the load testing so the focus would be on the Blaise instrument, and a custom instrument was set up to accept the PIN directly; and

- Performance counters were set on the Web and Database servers to capture the findings.

7. Results

Below we summarize some of the key metrics from the load test.

- A total of 333 completes were validated in the Blaise DB.
- This included mostly screeners, and a handful of extended interviews.
- The total test run time varied between 30 and 40 minutes.
- During this period the server monitors on both the DB and Web servers did not exhibit any abnormal behavior, sustained CPU spikes, or high memory usage.

This indicated that we had no memory leaks or other application performance issues. Therefore, we concluded that we had a successful process, and the manual load test helped us make a decision about the server configuration used for the project.

7.1 Interpreting the results

The following factors were considered in making a final decision for the project:

- With the manual load test we got 300 + completes in less than an hour with a single server configuration. The response times were less than 2 seconds for most pages except for one or two which had randomizations or complex layouts.
- The load test also had 70 users hitting the application hard, and trying to complete 4-5 sessions in 30 minutes; this probably would be less intense in actual production.
- From our past experience with load testing large web applications, the Database server was always a possible bottleneck and even if we had multiple application servers the DB would still be one.

So based on the above factors the final server configuration we came up with included:

- Two virtual web servers,
- One virtual database server, and
- Network load balancing using 'Kemp'.

7.2 Issues Found During the Load Test

During load testing, we did encounter a load time issue. A page that was reported as being slow by all users (10-11 seconds to load) was a transition from a question that had several randomizations. After consulting with the Blaise developers, we learned that the additional load time was due to the client rendering time (in this case a browser). And that with a large table the server time will increase some as it has to build a bigger page definition, while the client-side rendering will increase much more in these circumstances.

Our slow page had a layout that involved a group with seven enumerated type questions. We concluded that the problem was due to a client preference for separate questions, each with its own template, rather than a table. Although we do not have benchmarks for page loading with different layouts, we found manual testing to be very helpful in identifying these types of issues.

8. Production set up

The production servers were each configured with a serverpark with all the roles: Management, Audittrail, CATI, Data, DataEntry, Resource, and Session. A local server (logical server) was created to allow the serverpark to be mapped to the website and the physical location of the installed survey. Figure 1 and 2 (below) shows the web and database server set-up and configuration, and Table 1 provides the server specifications.

Figure 1. Web and Database Server Set-Up

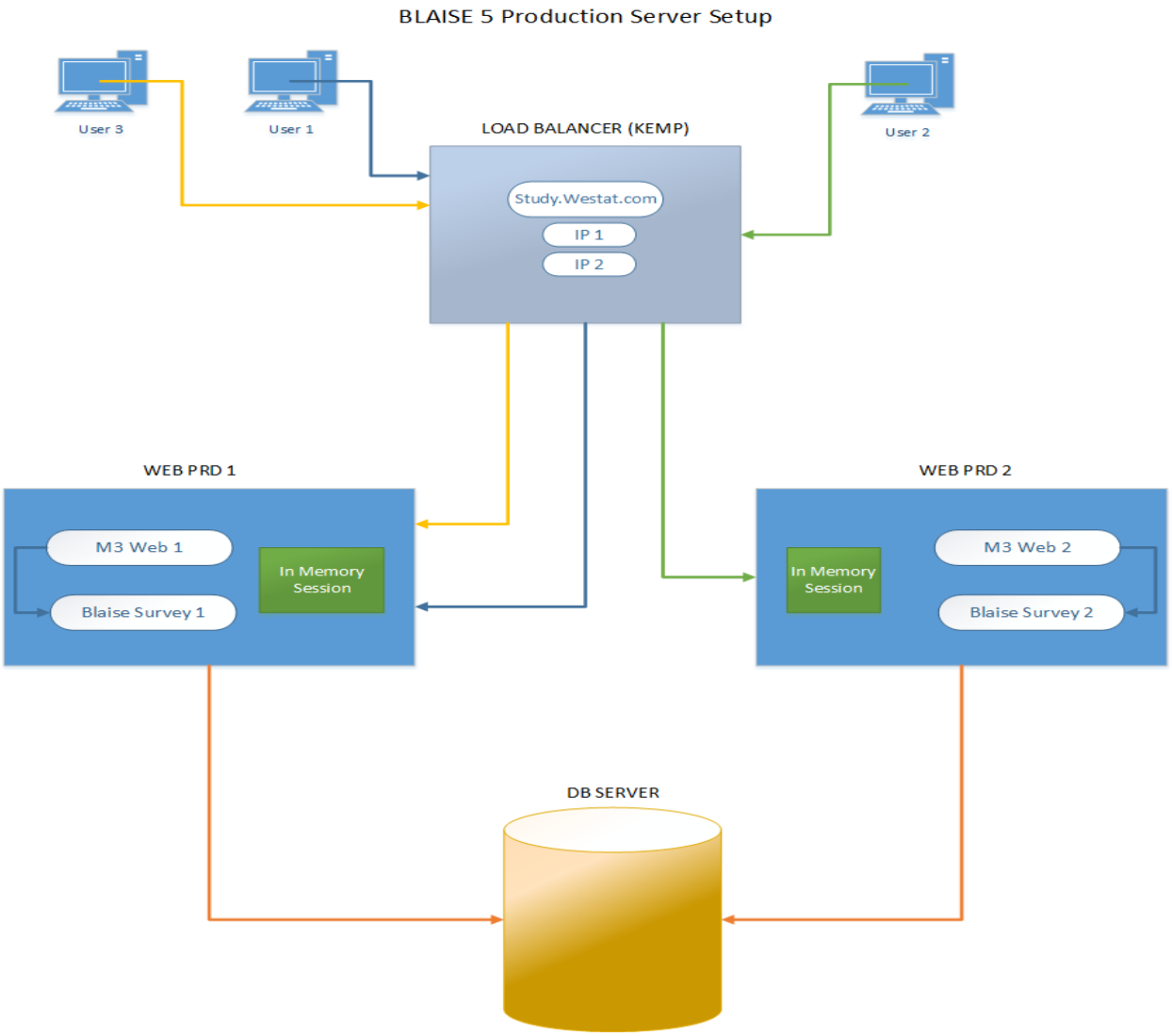
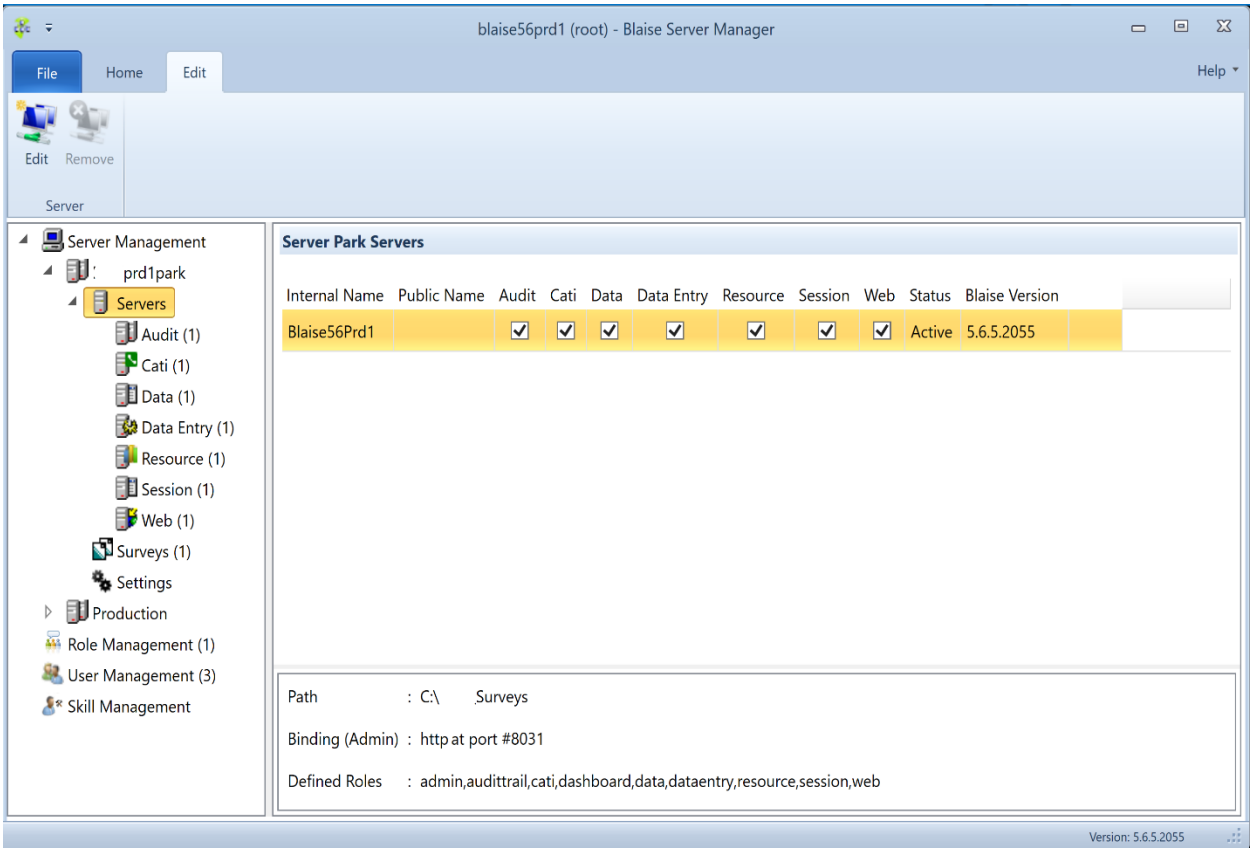


Table 1. Server Specifications

Server Type (virtual)	OS Version	CPU	Memory
DB Server	Windows 2016	Quad Core	32GB

Web Server 1	Windows 2016	Quad Core	32GB
Web Server 2	Windows 2016	Quad Core	32GB

Figure 2. Blaise Server Manager Configuration



9. Production Validation

Once we had the production environment set-up we had to integrate the management system. We also conducted a small-scale load test to ensure that the production server set up with the load balancer was working smoothly. We report some observations below.

- The security implementation for the instrument involved Blaise doing a check with the token that was created by the management system. Since this was done using a 'Blaise' look-up and not a SQL read, there was a concern about the speed of doing the validation. This lookup was done by a Blaise search (TokenModel.SEARCH(pToken)) followed by a Blaise Read (TokenModel.READ). The lookup table itself was a SQL Server table with a primary key where keyvalues are GUIDs. In order to check whether this look-up would be a performance issue when we had 100k + tokens in the DB, we prefilled the table with two hundred thousand (200,000) records and manually tested running cases. No perceptible slowness in response time was found so we concluded that the Blaise Read performed very well.

- Our initial smoke testing in production was not successful. Some sessions worked fine while others failed. On further analysis of the set-up, we realized that the management system to launch the instrument was set-up on only one server and we had to replicate it on the second server.

10. Performance in the field

The project went live in early 2020, and overall performed really well, with very few issues reported through the help desk. Some issues that we did encounter are detailed below.

- *Session Data Overwritten*: The 'session data' for a case exists for two devices for the same case at the same time. When a respondent started the interview on one device (phone), but shortly thereafter decided to do the interview on a desktop, the phone session did not timeout until after the desktop session was completed. Because the respondent went to a device with a different ip-Address, the load-balancer didn't know to send the desktop case to the same server that the phone had been on. The respondent did successfully complete the case on the desktop. However, although the phone's 'session data' really didn't have much data, the phone's data overwrote (wiped out) the data collected on the desktop since it was the last to execute. This was fixed by looking at audittrails and re-entering the data.

- *Submit button click issue*: When a respondent reached the last page, but did not click the submit button (instead perhaps closing the browser to end the session), the case status and associated variables such as the timestamp were not set correctly. The submit button does some AssignField actions to set status and timestamp variables. This is something to be aware of when reporting on case completion.

- Metrics related to completions during peak user loads compared to an average of 496 completes to date. The maximum completes per hour during this peak load week was 197 cases.

Table 2. Peak load following project mailings in early January

Day of Week	Started	Completes
Monday	4,225	2,250
Tuesday	3,850	2,175
Wednesday	3,100	1,650
Thursday	4,350	2,300
Friday	4,400	2,400
Saturday	3,400	1,850
Sunday	2,500	1,275

11. Alternate Server Configurations

Blaise 5's inherent architecture that consists of multiple server roles, lends itself to a distributed server farm set up to handle large-scale applications. The recommended server set up from Statistics Netherlands is represented below.

- Application Server 1 - Session, Data, Resource, Management, Audit Trail Web and DataEntry Roles.
- Application Server 2 - Web and DataEntry Roles
- Database server – That hosts the SQL DB.

Since we had no experience working with this type of configuration and the project was on a tight deadline, we did not have the time to set-up and test this. But we plan to test this server set up and to replicate our load tests.

12. Conclusion

Based on our experience we believe that the Blaise 5 architecture is capable of handling large-scale web applications. Though we could not leverage the improvements in randomization functions in Blaise 5.7 due to the timing of this project's schedule, the web survey performed well in production with very few issues. There is still more work to do in the area of web load testing tools suitable for Blaise 5 and we are actively exploring multiple options.

Every project will have some unique characteristics that may require some tweaks and adjustments. This case study shows the powerful capabilities of Blaise 5 for large-scale web surveys.

Using the Resource Database to Adapt to Session Timeouts

G J Boris Allan, Joseph Allen and Siu Wan, Westat

1. Abstract

If an interview session times out, an obvious method of recovery is to use an error page to trap the error and then launch a call to a recovery URL. Unfortunately, because we have lost contact with the session data, it is problematic to recover the case ID of the interview we just lost (State.KeyValue is null). Also we have to hardcode the value of the recovery URL in the error page because any value stored in the survey database is unavailable as we have lost contact with the session data. Using our resource-database methodology developed for web security we can handle session timeouts and do not have to hardcode URLs – we can modify the URL without changing the package, because we just change the preloaded data.

2. Introduction

There are three common ways to stop and possibly restart any web page:

1. Users have had enough, and there is no Exit feature available on the page[3] so they close the tab by selecting X top right on the browser or selecting X on the relevant browser tab. We will term this "X-out". The tab is lost and there is no currently-visible URL.
2. Users want to refresh the information on a page for some reason such as wanting to re-enter the data typed on the page or to unfreeze a page that seems stuck. Sometimes selecting refresh is accidental. By refreshing, the currently-visible URL is relaunched.
3. Somebody has made a copy of the web-page URL, the page is closed, and later the web page is relaunched using the copy of the URL.

All three forms of restart involve security issues. For example, copying a URL with the intention to restart an interview is a clear way in which malefactors can try to highjack an innocent encounter with the intention to hack. The types of check discussed in an earlier paper deal with these issues.[4] The result for each check is a Quit command, and what to do after the Quit is not specified.

Issuing a Quit is related to our final problem – how to handle a Blaise 5 server session timeout. A server timeout happens when the user does nothing in the interview for more than a specified time (the default for Blaise 5 is 20 minutes). For a Blaise 5 server, "do nothing" does not necessarily mean nothing has been done, rather it means that no data were sent to the server from the web page for the specified time – data might be entered on the page but no next page action was issued.

3. Session data and the survey database

When you start a Blaise 5 web interview, the Blaise 5 server creates a server-session database by copying data from the survey database for the appropriate case. As the interview proceeds, data collected from

[3] However, for methodological reasons, some clients do not want an Exit or Save button on survey pages. Other clients, for other methodological reasons, want such an option available on all master pages.
[4] "Using the Resource Database to control web security", G J Boris Allan and Peter Stegehuis (IBUC 2020).

web pages are reflected in this session database. At the end of an interview, or normally after a `Quit` action, the data in the session database are copied to the survey database.

If you X-out from a web page (say, close the browser by selecting the X upper right) then the server does not know the user has disconnected, and the session is left open to be closed after the timeout period (we usually use the setting that says, on timeout, session data are saved to the survey database). Proactively, our mangement system (MS) could monitor the status of the remote user, and know when connection was lost by an X-out or a closed browser (the Blaise server is non-proactive, and only after it has had no connection from the user for the specified period is the session closed and data saved).

Suppose we have this clean break in connection, and the user tries to get back into the interview legitimately via the MS. It is easy to imagine that closing the browser (or a tab) could happen unintentionally and an attempt made to restart the interview within the timeout period. So here we are: the respondent accidentally has closed the tab, yet wants to continue, and comes back to the MS, going through all the recognized steps.
As all steps are correct with the login, the MS (using the API) starts the interview. The MS tries to set the IsLaunched field to Yes, but cannot because IsLaunched is inaccessible – we cannot start because a locked session is in progress. That is, the session data are still live (have not been deleted) because not enough time has elapsed for a timeout.

On a second login, the management session can know that session data are live, so it would seem obvious that the session data should be saved to the survey database, and a new session started. If the session data are not deleted then a new (second) instance of the interview cannot be started, because the Blaise server thinks the record is "locked". The record in the survey database is not locked, rather the session-data record is locked, and will remain locked until the first instance times out.

There is no way that it is possible using the API to nullify the lock ID attached to a session database instance. We have proposed an addition to the API which will give this flexibility, but until then we use a more elaborate method with the resource database. There are other reasons we would like this flexibility with the lock ID, because the less complex the security mechanisms we have to use the more robust our defences. The integrity of the Blaise database has the highest priority so allowing a user (however well-qualified) to remove a lock would not be implemented. In addition, automatically locking a record is a good default. In some situations this can create issues, and further discussions are underway on this point. Currently, the integrity of the Blaise database is paramount and removing a lock is not permitted.

Another less complex approach we considered was using the API to see the original session instance, and then to call the `Quit` method on the session. The MS (via the API) would then loop waiting for the session to close and, once closed, launch a new session instance.

There was, however, a problem with the time taken for the original session to close because, for any particular session, the time taken to close was unpredictable. In some tests closing took seconds and in other tests closing took minutes. We did not think, given the variation, that this approach was consistent enough to field. If the closing time could be made consistent and short then we thought this approach would solve the restart problem and might be easier to implement than the unlock feature.

If the user (or an intruder) tries to restart the case by using the original URL after an X-out before the session is closed:

1. Whenever we start a new interview session, we compare the previous session ID to the current session ID,

2. If the two IDs are the same, then we can

    2.1. `Quit` and `Save` the session,

    2.2. give a warning,

    2.3. return to the MS.

To return to the MS, we use a URL that we have stored in the survey database (and thus in the session database).
When a session times out there are no session data available, because in theory the data have been saved to the survey database and the session has been removed from the server. However, there were problems when the session had not been removed before a new survey was opened from the same IP address – this meant a case was launched with the new case ID but with an old session.[5]
Once the session no longer exists, we cannot access the values in the survey database directly but, however, Blaise 5 opens *a generic* error page that reports the type of "error". We suggest that Blaise 5 should open a *specific* session timeout error page. The specific session timeout error page will have a custom action that will enable us to act on the timeout event.
We know of two ways that we can approach this custom action, either launching a hard-coded URI or launching a special survey case that has an exit URI that can be changed as the needs of the project changes.

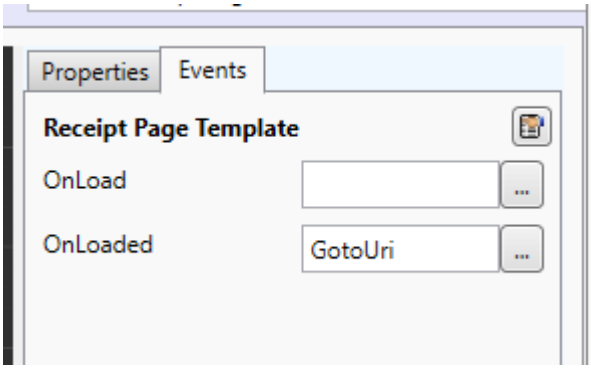
4. Different ways of ending an interview

There are various types of page template that are used to cope with reasons to leave an instrument.

4.1 Receipt Page Templates

This is the way we would like all interviews to end. That is, the interview has gone all the way to the end, and is complete.

Often receipt pages thank the user and, to end the interview, have a button to select for a return to the MS. Other clients like to thank the user, but want to return automatically to the MS and thank them there. We accomplish the return to the MS by use of a redirection of program control to the management site when the page has been loaded (`OnLoaded`):



[5] We think this was remedied in Blaise 5.6.9.2082.

The `OnLoaded` program expression is:

```
<ReceiptPageTemplate OnLoaded="{Action GotoUri({Expression ExitURL.ValueAsText},{Values
        'Target=_parent'})}">
```

That is, when the page has been loaded, go to the URI (ExitURL) that has been mapped from the MgSys block – this is, the address of the MS web site.

4.2 Abort Page Templates

There are times, within the set of rules in the resource database, when there is an instruction to Quit and this action opens an Abort page (calling this action an abort seems a little extreme). In our system, the Default abort page template does not use a field reference but uses a server variable named URL – the idea is that we have extra flexibility because (if necessary) we could change the server variable value depending on the source of the Quit – even assign a hard-coded reference.

The expression is:

```
<AbortPageTemplate OnLoaded="{Action GotoUri({Expression ServerVariables.GetString('URL')},{Values
        'target=_parent'})}">
```

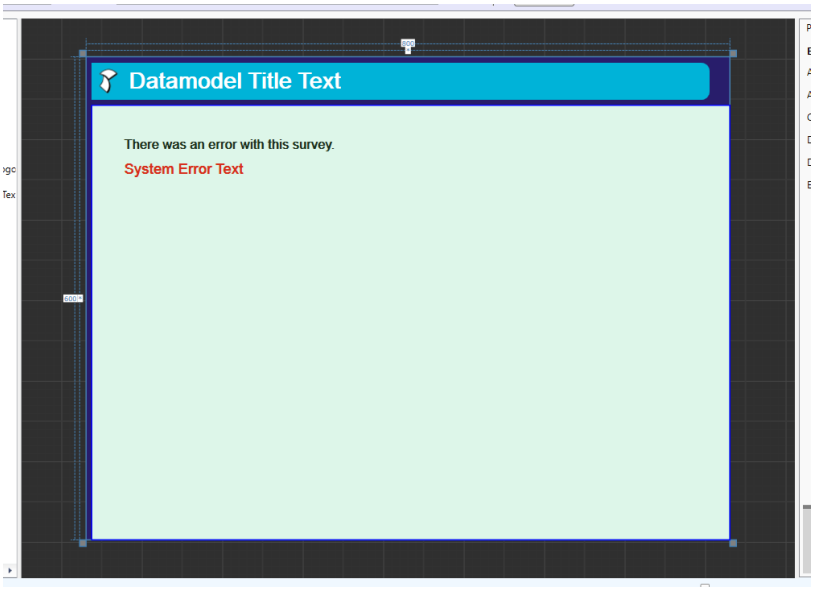
This follows the same format as the receipt page `OnLoaded` event.

4.3 Error Page Templates

If an instrument encounters a “system” error, such as a record locked by another user, then an error is shown. The list of system error codes is long, and includes:

- `InvalidKeyValue,`
- `RecordExists,`
- `ServerAccessDenied,`
- `SessionExpired,`
- `BrowserNotSupported,`
- `SurveyNotActive,`
- `RecordNotFound`

You will probably be aware of these error pages because of the Default error-page template:



In general, we do not change the default error page template because we do not want the user to have any easy way back to the survey MS, as the user could be an intruder testing ways into the instrument (hacking). We do, however, have a special TimeOut error-page template for one specific error – in the list of error codes the code named “SessionExpired”.

5. The Timeout error page template

In the resource database Error Page Templates list we have a new template named “TimeOut”, which is first in the list of error templates. This means that Blaise 5 looks at this template first, and decides if the template is applicable for the current session error.

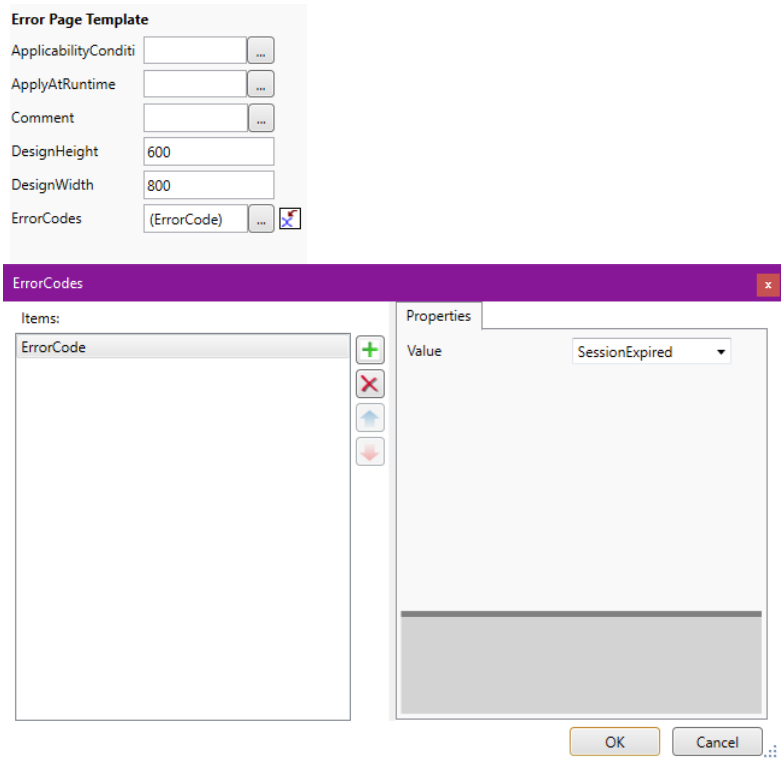
5.1 TimeOut version 1

Looking at the underlying expressions for this version of the Timeout error template, two interesting aspects are `OnLoaded` and `ErrorCodes`:

```
<ErrorPageTemplate
        OnLoaded="{Action GotoUri({'https://www.westat.com',{Values 'target=_parent'})}"
        ErrorCodes="SessionExpired">
```

We cannot use server variables or field references for our `GotoURI` because the session expired and, with the session expiration, the variable references also expired (and we have not found a way to keep them alive). Only errors that are “SessionExpired” are recognized by this template, and so – as TimeOut is first

in the list of error page templates – whenever the session expires this page is selected. Our problem is the hardcoded value of the URL.



5.2 TimeOut version 2

In an attempt to give more control over where we send the user after a session has expired, we needed to start a new session about which we had clear knowledge, and then act on information in that new session (a bit like our action with `Quit`/`Abort`) to send the user to a web page. If we hardcode the web-page location in the `OnLoad` event for TimeOut, whenever the location changes then we have to make the change and reinstall the survey.
The new session we create is back on the Blaise server and is attached to a "dummy" record/case we know as "*". The * case is part of the main survey database, and we can assign (via the API) appropriate values to fields in MgtSys. We do not assign a value to the field that says the case was started correctly, and so the resource database code issues a `Quit` instruction, that generates a Default abort page which then goes to a URI defined in the * case MgtSys block.

To start the survey with * in the current survey database, we use the `OnLoad` event (not `OnLoaded` because the action we want cannot be selected for that event). Here is the expression:

```
<ErrorPageTemplate
        OnLoad="{Action StartSurvey({Expression State.InstrumentId},{Values '*'},'','')}"
        ErrorCodes="SessionExpired">
```

When we load the TimeOut page, we run the `StartSurvey` action for the current survey (which is based on the resource database `State.InstrumentId` variable, not lost with the session). The survey to be started has a key value of *, and the only time the page is relevant is when the Blaise system error is `SessionExpired` and we start the * case.

This technique seemed to work during development but then we put it through more extensive testing. What happened in practice is that the previous session seemed (sometimes) to be still alive and not ended, so that the previous key value was replaced by *, but the rest of the data from the previous key value remained. We decided for the pretest to hardcode the return URI.[6]

[6] After that decision was made, Blaise 5.6.9.8022 was released and in the change log we read that there had been a problem with `Action StartSurvey`, very relevant to our experience. The change log read: `2019.11.25 | HELS | Resource Editor | BL56-993 | BSR-3698 | End current session before StartSurveyAction is executed`. This explains our experience. We are not, however, changing anything until the pretest is in the field.

Managing a Complex Infrastructure for the Collection of Business Report Forms

Leif Bochis Madsen, Statistics Denmark

1. Abstract

The last two years Statistics Denmark has been working on the replacement of MS Infopath forms in our Business Survey Division.

By February 2020, seven questionnaires are used in production, while eight others are under development and should be ready for production by the end of 2019. Still, approx. fifty questionnaires should be converted to Blaise 5 within the next three years.

The paper describes the status of the project including a number of issues that have been addressed and some of the tools we are building in order to incorporate the use of Blaise in the existing environment for collecting business form reports.

In detail, the paper describes our work on managing a range of servers needed in the environment.

2. Introduction

2.1 Statistics Denmark Business Survey System (BSS)

As described in my paper for IBUC/2018, Statistics Denmark has introduced Blaise 5 into an existing infrastructure to replace report forms written in MS Infopath with Blaise 5 questionnaires.

Put briefly[7], this infrastructure implies the use of a Frontend server and a Backend server. The **Frontend** server is accessible to the respondents and here the surveys are installed as standard Blaise 5 Web surveys. The **Backend** server takes care of reading, writing and removing data, i.e. initialization of the forms with background information, and – after completion – removing data from the frontend Blaise database to store it in the background database.

[7] For extended explanation, see ref. [BSS].

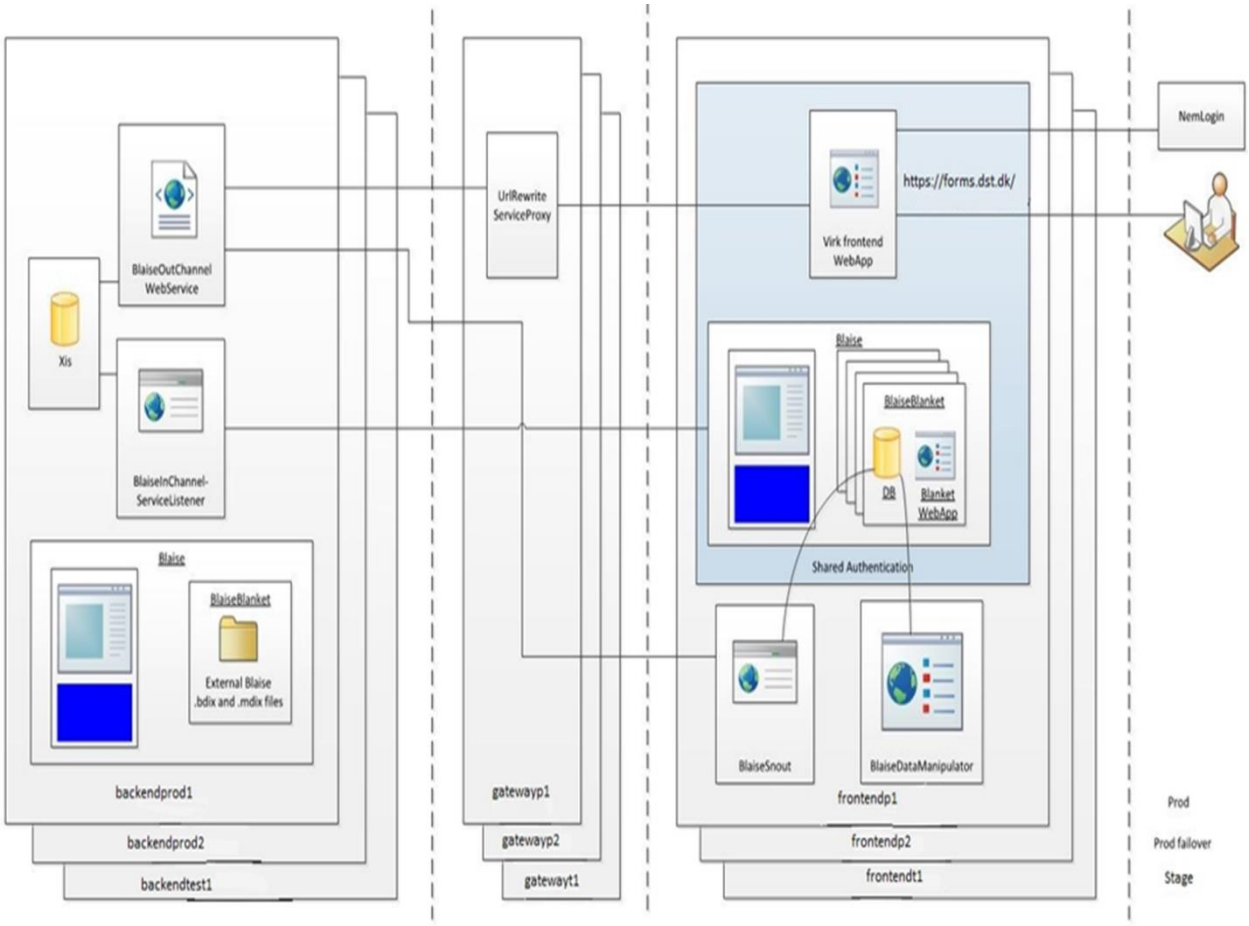


Fig. 1. Architecture of Business Survey Portal

The concept of using backend and frontend servers for the different parts of the functionality is also used for a lightweight version of the form report system, the so-called 'link access' mode. This mode is used as an easier way for respondents to report data. They will not need to set up the entire public digital key interface before starting to fill, but on the other hand, the requirements prescribe that no previously filled data may be exposed to the respondent during the session. For some surveys, mainly the voluntary, this mode tends to increase response rates.

However, not all environments fit into this overall architecture.

Another two environments in the BSS – one for test and another for production – are used internally to fill forms if e.g. a respondent calls by phone and wants help to fill the form. These two environments share most of the functionality, like database access to prefill and store data, with the other BSS environments, but there is no need to split up the functionality between backend and frontend servers. In other words, these servers are COMMON.

At last, a number of servers that do not belong to the BSS. These servers are mainly used for household surveys and are set up as 'classical' Blaise servers, i.e. COMMON.

Environment	Test/Prod	EnvType	BACKEND	COMMON	FRONTEND	FW	COMMON	FRONTEND
VIRKTEST	TEST	VIRK	backendtest1		frontendtest1			
LINKTEST	TEST	LINK	backendtest1		linktest1			
VIRKSTAGE	TEST	VIRK	backendtest2					frontendtest2
VIRKPROD1	PRODUKTION	VIRK	backendprod1					frontendprod1
LINKPROD	PRODUKTION	LINK	backendprod1					linkprod1
VIRKPROD2	PRODUKTION	VIRK	backendprod2					frontendprod2
INTERNTEST	TEST	INTERN		interntest				
INTERNPROD	PRODUKTION	INTERN		internprod				
BlaiseTest2	TEST	BLAISE					blaisetest2	
Blaiseprod2	TEST	BLAISE					blaiseprod2	
Blaiseprod3	TEST/PROD	BLAISE					blaiseprod3	
Blaiseprod4	PRODUKTION	BLAISE					blaiseprod4	
Blaiset3	PRODUKTION	BLAISE					blaiset3	
Blaise2	TEST/PROD	BLAISE		blaise2				

Fig. 2: Overview of environments

To summarize, our Blaise environment can be described as a diversified infrastructure:

- Some servers are FRONTEND/BSS
- Others (data processing servers) are BACKEND/BSS
- A few are both, i.e. COMMON/BSS
- A handful are COMMON, but not related to the BSS
- Some are for PRODUCTION, others for TEST
- Some are INTERNAL, others EXTERNAL
- All of them has Blaise 5 installed
- All need to be administered

3. Server administration tasks

The common, administrative tasks comprise:

- Install surveys in the BSS environments (in Non-BSS environments the Blaise Server Manager interface is still used)
- Upload supplementary files to servers
- Get info about currently installed surveys
- Get overview of servers and environments
- Get info about Blaise versions installed on servers
- Update Blaise licenses (automatically)
- Update services

Some of the required tasks we shall describe further below.

3.1 Install survey in a BSS environment

The process of installing a survey in a BSS Backend/Frontend environment consist of a number of operations.

First, any possibly existing installation of the survey must be removed. The general policy is that it is allowed to replace a survey in test environments, but not in production environments. Thus, if a replacement is requested in a production environment the entire operation should be cancelled automatically.

Second, the package file must be uploaded to a proper placement on the backend server.

Third, from this position the survey can be installed on the frontend server, and afterwards, a remote data interface (bdix) can be generated using the metadata and manifest files extracted from the package file.

At last, the survey installation on the frontend server is modified by inclusion of an http module and some extensions added to the configuration file in order to install our security procedures.

Albeit a complex operation that requires actions carried out on backend as well as frontend servers, with differences between environments, it is still possible to automate it.

3.2 Overview of servers and environments

Instead of keeping track of the properties of each environment in a separate document, it should be possible to extract the current properties of the environments and servers directly.

For example, these overviews can be used to help managing upgrades, as well as to display the current status of each environment.

3.3 Update services

Developed as a prototype, these services are subject to constant change. It is therefore necessary to update these services regularly, and in order to decrease the burden of updating, it must be carried out automatically.

4. Implementation

The application consist of three, separate parts:

1. A WCF service implementing a series of methods carrying out specific tasks on a specific server
2. A Console application working as a client for the servers and implementing the knowledge of all the environments, servers and their mutual relations
3. A Graphical User Interface making the provided tasks easily available for the users

4.1 WCF Service

The services are implemented as a DLL serving as a general WCF service based on basic .NET technology. At present, the services are not making use of the Blaise API, but only communicating with Blaise via e.g. the Server Manager program.

The service DLL comprises a set of basic, non-public methods for reading and writing information from and to XML files, the Windows Registry and directories, uploading files, executing commands, updating log files, etc.

Public methods comprise methods of a more specific character like upload Blaise 5 installation packages (.bpkg files), install these packages on a Blaise 5 web server, return a list of installed surveys, etc.

It is important to keep a firm separation between the non-public and public methods in order to control which actions are available. Also, the services must be unavailable from outside and only exhibit their methods to relevant, authenticated users inside the organization.

The general WCF service exhibits two enumerations that a client can use in order to manage proper operations:

The **server type** defines three kinds of servers (BACKEND, FRONTEND, COMMON).

The **form type** defines four kinds of reporting (Web Form, Internal Report Form, Link Access Form, Classic Blaise Form). The first three denote the kinds of form reports in the BSS, while the latter denotes all other Blaise surveys. If needed, it should be easy to extend the list.

The exhibited operations comprise, e.g.:

bool DoesSurveyExist(survey name, server type, form type);
string GetOS(); // version of the operating system on the server
string[] GetVerifiedSurveyList(server type, form type); // List surveys of specified server and form type
string GetSurveyInfo(survey name, server type, form type);
string GetInstrumentID(survey name, server type, form type);
DateTime GetSurveyInstallDate (survey name, server type, form type);
DateTime GetSurveyPrepareDate(survey name, server type, form type); // Timestamp from bmix file
string GetBlaiseVersionNumber(); // Current Blaise version on server
string GetServerParkName(); // From registry info
string UpdateWCFservice(); // Replace service dll with newer version
string UpdateBlaiseLicenseInfo(license key, licensee, activation code);
bool RestartBlaiseService(out string logMsg); // Restart Blaise 5 services on request

Some operations are only relevant on backend/common servers:

string UploadBpkgFil(byte array, file name, file date);
string ServerManagerInstallation(destination server name, port number, server park name, user, password, package file name, form type); // Install, frontend/common server
string UnInstallSurvey(destination server name, server park name, instrument ID); // Uninstall, frontend/common server
string RemoveBackendInfo(survey name, form type); // Remove e.g. remote bdix and other files

Some operations are only relevant on frontend/common servers:

string GetSurveyBlaiseVersion(survey name); // Blaise version of instrument (info not available on backend)
string RunBlaiseHttpInstallScript(survey name); // Install security patches

4.2 Client Application

The client is a console application programmed in C#.

It implements the knowledge of the various environments and the servers belonging to each environment. Because the WCF service is installed as an exact copy on each of the involved servers, connecting to a service can be done by using a common method and only providing the needed endpoint (server name) to the specific server.

The client application can perform operations on all servers, on one server, on an environment or on a subset of servers or environments.

Currently, implemented operations comprise:

- Overview over all servers, including status info
- Overview over servers and environments belonging to the BSS
- Overview over an environment, including the installed surveys
- Overview over an installed survey, for each server
- Installation of a survey in one or more environments (BSS only)
- Update of the WCF Service on all servers
- Update Blaise license keys on one or all servers
- Running a test suite to produce a report on the health of the service(s)
- Remove a survey from an environment

As an example, figure 3 shows an overview of all servers and their properties.

Oversigt over RAMBLA-servere (i alt 17 stk.)

Oversigt genereret: 12. februar 2020 kl. 11:05

SERVERE

servernavn	status, service	OS	Blaisever.	serverpark	servertype
backendtest1	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2008 R2 Standard	5.6.9.2082	LocalDevelopment	VIRK+LINK/BACKEND
frontendtest1	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2016 Standard	5.6.9.2082	Production	VIRK/FRONTEND
backendprod1	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2008 R2 Standard	5.5.3.1862	LocalDevelopment	VIRK+LINK/BACKEND
frontendprod1	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2016 Standard	5.5.3.1862	Production	VIRK/FRONTEND
linkprod1	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2016 Standard	5.5.7.1883	Production	LINK/FRONTEND
internprod	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2016 Standard	5.6.9.2082	Production	INTERN/COMMON
blaisetest2k	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2012 R2 Standard	5.6.9.2082	Production	CLASSIC/COMMON
blaiseprod3	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2012 R2 Standard	5.5.11.1902	Production	CLASSIC/COMMON
blaiseprod4	RamblaServices.dll, build: 10-01-2020 16:59:22	Windows Server 2016 Standard	5.6.5.2055	Production	CLASSIC/COMMON

Fig. 3: Generated overview of servers by name, version of service, version of operating system, installed version of Blaise, name of server park and supported type(s) of forms and server.

Another example (figure 4) shows a list of surveys in a specific environment.

VIRKTEST - frontendtest1 / backendtest1 (VirkBlanket)

Oversigt genereret: 12. februar 2020 kl. 15:54

BLANKETTER

Blanket	ID	Inst.dato	Forb.dato	Blaisever.	Backend
bba2019a	5188651a-6ab2-4987-b209-8678b8697004	06-05-2019 16:45:05	06-05-2019 16:41:55	5.5.3.1862	OK: bba2019a, ...
bba2019b	dde95a45-4935-4bfb-b07e-ac5cb8fea61a	19-11-2019 09:44:17	19-11-2019 09:42:44	5.5.8.1886	OK: bba2019b, ...
blaisetestv5	e0682b17-3241-4239-95ae-4200254e9b09	01-10-2019 16:16:52	20-05-2019 12:41:06	5.5.3.1862	OK: blaisetestv5, ...
doi	86451a56-1e58-4f13-b2c3-918ec5465fc8	27-02-2019 11:57:35	27-02-2019 11:21:33	5.5.3.1862	OK: doi, ...
doi2019b	968afd90-0d15-4d49-8873-b1077d4e9cbc	10-12-2019 11:22:56	10-12-2019 11:14:20	5.5.8.1886	OK: doi2019b, ...
ipo	58ee31bb-5d00-4f63-9911-6eba4f507512	29-01-2020 17:57:44	29-01-2020 17:56:59	5.5.8.1886	OK: ipo, ...
ipo2020a	b9952956-bb8e-4230-98de-8dff7cc2ee65	29-01-2020 18:20:07	29-01-2020 18:17:11	5.5.8.1886	OK: ipo2020a, ...
ipo2020b	22733f6a-83ef-49b1-ae78-162c7dcf1332	04-02-2020 11:32:37	04-02-2020 11:31:34	5.5.8.1886	OK: ipo2020b, ...
lvs	5bc0d93f-b8e9-4639-8a47-652fd5ad87c5	11-12-2019 13:27:03	11-12-2019 13:23:06	5.5.8.1886	OK: lvs, ...

Fig. 4: Generated overview of surveys in a specific environment (frontend/common) by name, instrument ID, install date, prepare date of instrument, instrument version of Blaise and status of backend (if relevant).

Last example (figure 5) shows a list of a surveys of name starting with “doi” installed on all test servers.

Oversigt genereret: 13. februar 2020 kl. 11:22

BLANKETTEN 'doi*' PÅ TEST-SERVERE

Server	Navn	ID	Inst.dato	Forb.dato	Blaisever.	Servertype	Blankettype	Miljø
backendtest1	doi	86451a56-1e58-4f13-b2c3-918ec5465fc8	27-02-2019 11:57:41	27-02-2019 11:21:32		BACKEND	VirkBlanket	VIRKTEST
backendtest1	doi2019b	968afd90-0d15-4d49-8873-b1077d4e9cbc	10-12-2019 11:22:58	10-12-2019 11:14:20		BACKEND	VirkBlanket	VIRKTEST
frontendtest1	doi	86451a56-1e58-4f13-b2c3-918ec5465fc8	27-02-2019 11:57:35	27-02-2019 11:21:33	5.5.3.1862	FRONTEND	VirkBlanket	VIRKTEST
frontendtest1	doi2019b	968afd90-0d15-4d49-8873-b1077d4e9cbc	10-12-2019 11:22:56	10-12-2019 11:14:20	5.5.8.1886	FRONTEND	VirkBlanket	VIRKTEST
backendtest1	doi2019b	968afd90-0d15-4d49-8873-b1077d4e9cbc	10-12-2019 11:23:20	10-12-2019 11:14:20		BACKEND	LinkBlanket	LINKTEST
linktest1	doi2019b	968afd90-0d15-4d49-8873-b1077d4e9cbc	10-12-2019 11:23:18	10-12-2019 11:14:20	5.5.8.1886	FRONTEND	LinkBlanket	LINKTEST
interntest	doi	86451a56-1e58-4f13-b2c3-918ec5465fc8	01-03-2019 10:37:19	27-02-2019 11:21:33	5.5.3.1862	COMMON	InternBlanket	INTERNTEST
interntest	doi2019b	968afd90-0d15-4d49-8873-b1077d4e9cbc	11-12-2019 15:48:02	10-12-2019 11:14:20	5.5.8.1886	COMMON	InternBlanket	INTERNTEST

Fig. 5: Generated overview of surveys named "doi*" by server name, survey name, instrument ID, install date, prepare date of instrument, instrument version of Blaise, type of server and form and name of environment.

4.3 GUI – Graphical User Interface

The services are available to the users through an interactive Manipula program that exhibits the relevant subset of the services.

Fig. 6: Selection of tasks[8]

The GUI gives access to a collection of local and server oriented tasks. Local tasks comprise utilities related to development of questionnaires. For example, to verify that a developed questionnaire is in accordance with the standard templates and the backend database, i.e. that data collected with the instrument can be exchanged with it.

The task chosen may lead to further dialogues. For example, installation of a survey leads to selection of the environment (or subset of environments) and to selection of a package file.

[8] "Rambla" is an acronym for *Framework System for Blaise Form Reports in Business Surveys*. "Blanket" is Danish for Form Report (questionnaire, survey).

Afterwards, the task is transferred to the Client program to carry out the subtasks needed to install the survey in the requested environment(s). I.e., the tasks of uploading the package file, possibly removing an old version, installing the survey, creating remote BDIX, and modifying the installed survey.

5. Current status

The management related services

- WCF Service is installed on all the servers
- Questionnaire designers can install surveys in test as well as production environments.
- January 2020 we were able to renew the license keys and activation codes for all the servers in our Blaise server portfolio automatically.
- We can use the extracted lists of servers including e.g. currently installed Blaise version to help planning our upgrade procedures. For example, all test servers are now (February 2020) running Blaise 5.6.9, while we are still planning to upgrade production servers as well.
- It was quite fast and easy to develop the user interface in Manipula, though interactive Manipula still performs relatively slow.

The overall progress of converting questionnaires

- By February 2020, seven questionnaires for business surveys have been developed in Blaise 5 and put into productio006E
- Eight more Blaise 5 questionnaires are under development
- Still, approx. 50 questionnaires must be converted before 2023

6. Future developments

We are planning to incorporate the above mentioned client program into an existing, general toolbox for management of Business Surveys.

The toolbox is running on a separate server and has finely grained access control implemented. Therefore, it should be possible to restrict access to the specific services to users that really need them and thus improve the general security.

Also, work is carried out in order to automate generation of Blaise 5 questionnaires that are in accordance with our background database for business form reports.

7. References

[BSS] Leif Bochis Madsen: Implementation of Blaise 5 web questionnaires into an existing Business Survey Management System, in: *Proceedings of the 18th International Blaise Users Conference*, Baltimore 2018, https://www.blaiseusers.org/2018/papers/5_5.pdf

Evolution of Blaise Survey Development in Statistics Finland

Joonas Salmi, Pyry Keinonen and Petri Godenhjelm Statistics Finland

1. Abstract

This paper discusses how Blaise Survey Development is organized in Statistics Finland and how the Survey Development process has evolved during Blaise5 implementation from 2017 to 2019. Statistics Finland has implemented Data Collection Management System which has influenced and set requirements for Survey Development. Data Collection Management System has been in production since January 2019.

This transition included change in work methods and culture. The renewed ways to work have caused a need to recreate the in-house survey development process and recognize the roles needed in daily basis work. Instead of starting Survey Development from scratch and creating customized surveys on demand, the focus was to find new ways to reduce the amount of effort by standardizing. The aim was to streamline Blaise Survey Development.

Standardization has helped both the transition process from Blaise4 to Blaise5 and creating new surveys from scratch. This meant in practice the implementation of the use of version control, standardized survey-independent-layout, and standardized survey-independent base code as a part of survey development. This way we have been able to harmonize the development phases of Blaise5 questionnaires and reduce the effects caused by person dependent practices.

The next step is to implement Agile methods in survey development starting from 2020. This step includes the processes and implementation how to lead and organize our Survey Development and work resources as a whole. The goal is to reduce wasted effort and make resources available efficiently when needed.

2. Background

Transition from Blaise 4 to Blaise 5 led to rethink the survey development process and practices. At the same time there was a transition to new generation of Blaise survey developers. With new generation of developers and new software, it was a convenient time to reform the whole survey development process.

The need for the reform of the survey development process was due to the fact that Blaise survey development got too person specific and the new data collection management system set technical requirements for Blaise surveys. On the old production environment Blaise 4 surveys were prepared to a network drive and surveys were run on local Blaise installations, whereas on the new production environment Blaise 5 survey packages are deployed to our new data collection management system called Ruuti.

Previously Blaise developers had their own Blaise surveys to develop and they were mostly stored on local or network drives. These practices showed as weak source code collaboration and created a lot of individual customized solutions. In the worst-case scenario everyone developed with their own survey-specific Blaise Resource Database file (.blrd), which meant that they weren't compatible with each other. This also fragmented the layout development and created major problems to managing it. Different surveys might have had functionalities, which weren't available in other surveys.

Developing without a uniform mode of work made Blaise survey development fragmented and project deadlines harder to reach. The experiences gained was the basis for the reforming of the survey development process. The aim was to streamline survey development process and make it easier to manage. Utilizing Git version control system to better handle source code collaboration and versioning was the backbone for the success of the survey development reform.

2.1 Process' progression & roles

Work roles and work management has stayed the same, while Blaise survey development has changed. This prompted to rethink the way how surveys should be developed to best fulfill the needs of the Statistics responsible for the data collection.

Prior survey development process helped to form core guidelines for renewed Blaise survey development process and different stages of work. These guidelines define all the different roles and their responsibilities associated with Blaise survey development. Different roles and their responsibilities aren't specified based on organizational titles but rather by real work tasks. These guidelines aim to make Blaise survey development processes clear to the whole organization.

Blaise survey development is starting to resemble more software development. Some of the projects have started to adapt more agile way of work. It started by working through different cycles or iterations. These iterations consist of updating the survey with the changes and then testing the survey. Defining clear iteration steps makes survey development more efficient by concentrating the work focus to one iteration step at a time with all relevant experts included. Changing the work culture and way of work has been a long process and it's still ongoing.

3. In practice

Improving Blaise survey development meant that not only the ways to work was changed but also the standardization in programming was implemented. In practice, the organization started the reform with the rational decision to standardize Blaise Resource Database file and layout. Using only one resource file across all surveys ensures that every survey has the same functionality available. Introducing version control and Ruuti pushed for standardizing base solution structure.

The base solution structure is the same in every survey. It includes always the main survey data model project as well as pre-fill data model project and a manipula project for pre-filling a Blaise Database with test cases. Also, the standardized projects that contains classification tables for lookups are included if needed.

Ruuti handles multi-mode surveys and all the related tasks. It utilizes Blaise as a data collection instrument. After Survey packages are deployed to Ruuti, they are automatically downloaded and installed to interviewers' personal workstations. Ruuti system sets also requirements for the survey project. These requirements are unified data model with standardized Blaise Settings and ontology variables for passing data from Ruuti System to Blaise and vice versa. In practice the survey is divided into sections by question themes and every section is programmed in their corresponding blocks in separate Blaise Include files alias .incx. This hierarchical structure makes the survey data model file alias .blax easier to maintain and the code more comprehensible.

Survey development is divided to three categories based on the complexity level: basic, advanced and master. Developer on basic level can only fix question texts. This is usually done on a text editor instead of using the combination of Blaise and Git version control. Advanced developer can program the whole

survey, if there aren't complicated structures. Master level developer is needed only for the most complex survey structures.

Developers in IT-department have the core competence to run Blaise in our architecture. Anyone, who has the skills in our organization, can program surveys in basic and advanced levels. This is encouraged to keep resources available for more demanding tasks. Blaise team handles also the survey installation on test and production environment.

3.1 Survey development process

Developing a survey starts with a new assignment. If it is a completely new survey, a new repository is created in version control with core survey solution. This is master level responsibility in Statistics Finland. Advanced or master level programmers then may program the survey data model and rules and commit their work to version control.

After the survey or a part of the survey is programmed it can be deployed to the test environment when needed. Deployment starts a test cycle where the programmed sections or the whole survey is tested and returned to programmers, if changes are needed. Any project member at the basic Blaise programming level can fix question text errors, when they are spotted while testing. All changes that require programming are distributed to more advanced programmers according to the challenge.

Thus, survey development is done in continuous cycles. After a cycle the changes are programmed, and the next cycle begins and new assignments such as new sections are programmed and issues from previous cycle are fixed. Survey development process is visualized in figure 1.

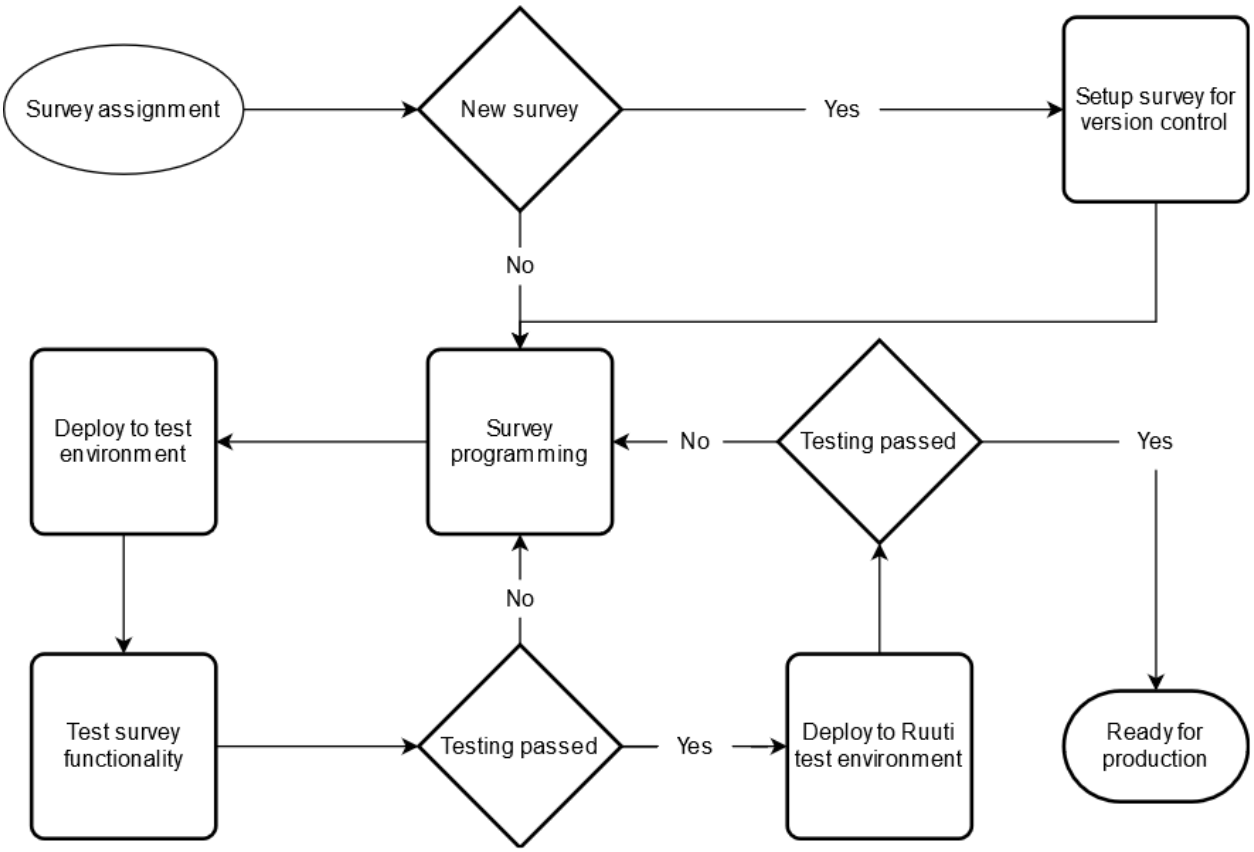


Figure 1: Survey development process.

When the survey development is in its final stages, the survey is deployed to Ruuti test environment. There it undergoes technical testing to make sure Ruuti system handles the survey correctly. Content and functionality are tested outside Ruuti environment. Ruuti tests are done to ensure survey compatibility with Ruuti.

After all the testing is done, the survey is deployed to Ruuti production environment. From there the survey is available to be downloaded to interviewer's workstation for installation. Respondent samples, timeframes and other specifications are done separately from the survey package. If the survey is in production, and there is an issue to fix the survey can be updated. Ruuti handles incompatible data model in the middle of a data collecting period, if all the previous fields still exist in the new survey version. Field numbers can be increased, and checks changed. However, between the data collecting period, the data model can be changed entirely. New published survey version is automatically downloaded and installed to interviewers' workstations.

3.2 Implementation of version control

After implementation of version control, collaborating on Blaise survey source code has become effortless. Instead of taking turns to work on different parts of the source code, it can be worked on simultaneously. Version control takes care of all the changes in the same place and version history readily accessible. The only exception to this is the Blaise resource file. Version control can't keep track of changes of a binary file.

Implementation of version control started with Microsoft Team Foundation Server using Git version control system. Statistics Finland has recently switched to a cloud-based Microsoft Azure DevOps. Because transition to new way of work is still ongoing, the new Azure DevOps features aren't yet fully implemented to daily work. At the moment the use of backlogs and management features is only at the design stage.

Git version control system keeps track of different survey versions. There can be more than one different version of the same survey running at the same time with different data models. The right version is easy to check out and work on. This makes implementing bug fixes easy to different versions of Blaise surveys, even in the middle of data collecting period.

3.3 Standardized practices

Blaise survey solution is standardized to allow straightforward integration with data collection management system (Ruuti). All Blaise surveys use a similar (.blax) source file. The source file defines modes, languages, roles, special answers, attributes and the necessary fields to fill out respondent and survey info. Ruuti sets requirements to fields that must be defined to ensure compatibility.

With the technical advances, the survey layout wasn't up to date anymore and it needed an upgrade. For a better usability experience, the mobile first principle was chosen while designing the new layout for self-administered surveys. The layout design was developed in collaboration with the cognitive laboratory, which is our questionnaire design and pre-testing team, to ensure great user experience. The new layout is standardized, and all survey specific texts are assigned to variables on the contrary to the old layout. This leads to better and easier maintenance when every survey uses the same layout. This ensures that all the Blaise surveys have similar structure that the interviewers are familiar with.

4. Conclusions & Future

The current way of work is a great improvement on the old way of work. Still, there are improvements to be made. There are multiple projects running simultaneously and allocating resources to the right projects at the right time has proven to be difficult. Last minute minor tweaks are still being done right until the beginning of the data collection period, although developer resources would be better used of on another survey. Survey development would benefit from system to allocate the resources where they are needed the most.

4.1 Agile practices

Programming in most survey development projects started to follow continuous cycle programming style. This makes it possible to adapt to a more agile survey development style. The implementation of agile survey development has started and is already in effect on some survey development projects. The agile way aims to resolve difficulties with developer resource allocation and with simultaneous work assignments. Instead of working on simultaneous surveys, the survey developer can focus on one survey at a time.

The switch to agile methods leads to defining new roles and responsibilities. Every survey has a product owner, who has the best knowledge of the substances of the statistics responsible for the survey. Product owners had to be trained to the new role and their responsibilities. Survey development is organized to one-week sprints. Product owners define, with core Blaise survey developers, what are worked on during the sprint and then propose a time when the sprint is held. Ultimately the steering group organizes the sprints and gives them a priority. This ensures that less important tweaks aren't prioritized to the same level as more major updates anymore.

Transitioning Blaise survey development to agile survey development is a bigger change to way of work than what was done recently. The process is developing, and it is more than likely to differ in the future from the one that is being implemented now.

Blaise 5 CAPI in Collaboration with COTS Software

Rogier Hellenbrand, Statistics Netherlands

1. Abstract

At Statistics Netherlands we are in full swing with the transition process between Blaise 4 and Blaise 5. Our main way to conduct primary data collection is CAWI and of the in total 125 different surveys, xx have already been converted to Blaise 5 CAWI. Also our new CATI channel is on the verge of completion after several pilots and the go-live of the first two surveys in may this year.

So our next task at hand was the CAPI mode, a mode that is quite expensive per interview conducted, but still necessary to get the right coverage for specific target groups in specific surveys.

2. Business case

Our current (legacy) CAPI channel faces a number of challenges:

- The technology used for synchronization between the host system and the local devices is outdated and file driven, therefor quite vulnerable. Loss of data has happened more than once.
- The assignment of addresses to interviewers is done by a 'as the bird flies'-algorithm, which in a country with as much water as ours leads to up to 30% manual alterations.
- Synchronization is a manually started process and leads to human errors.
- As the other interviewing channels are transiting more and more to Blaise 5, keeping CAPI in Blaise 4 would mean significant extra effort in building questionnaires.
- Field interviewers spend a lot of time (approximately two hours a week, roughly 10% of their working time) on administrative tasks, such as time keeping and travel expense declarations within the company's ERP system AFAS.

The architectural principles of the Phoenix program (responsible to rebuild the complete data collection landscape at Statistics Netherlands) are as follows:

1. Re-use if technology isn't outdated
2. Use Commercial-off-the-Shelf software if the vendor isn't too small
3. Build customized software ourselves only if 1 and 2 are not available

Bearing these principles in mind we investigated how we could form a CAPI channel meeting our specific needs.
As Blaise was already working on the Case Management App (CMA) for CAPI and our approach is to build omni mode questionnaires, Blaise ticks the boxes 1 and 2. But that left us with no solution for the assignment process and the manual labor of administrative tasks. Through our AFAS supplier we came in contact with a AFAS- certified system integrator named Way2Connect, who had already implemented several interfaces between their Link2 software suite and AFAS on the time keeping and expenses side. Furthermore this suite has a build in functionality for route optimization and planning, so it could take care of the address assignment problem.

After a Prove of Concept phase we concluded that the combination of Link2 and CMA would suit our needs the best.

3. Process description: preparation

The future business process starts with a tested Blaise questionnaire, which is deployed to a Blaise server farm. From the host system channel assignments are sent to the Blaise server farm if the addresses are allotted to CAPI. These addresses are not yet linked to a specific interviewer.

From statistical analysis it is known how certain variables have influence on the time it takes to complete the interviewer tasks. These are e.g. age of the respondent and level of urbanization. As these variables are used only within the CAPI channel, the choice was made to enrich the data within the channel by a custom made component: a master table is maintained within the channel linking surveys and variables to average process time and average re-visiting numbers. And each new address will be compared against this master table to enrich the right data. A custom made interface delivers the enriched addresses to Link2, where the assignment of addresses to interviewers takes place.

To be able to make these assignments, Link2 requires knowledge of the work rosters of the interviewers. These rosters can either be made inside Link2 or be transferred from an ERP system by interface. The assignment process within Link2 now optimizes available timeslots of interviewers against respondent addresses (minimization of travel time). The number of iterations is configurable. The result of this assignment process looks as follows (figure 1):

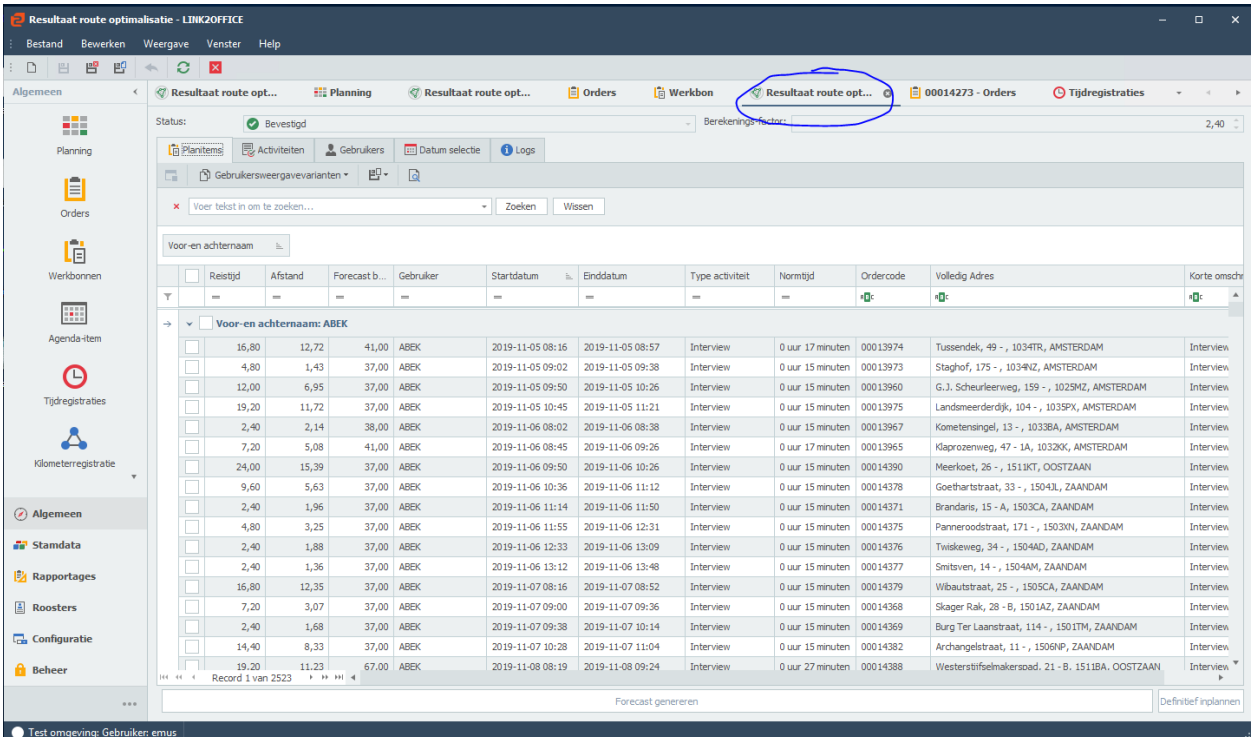


Figure 1: Result of route optimization within Link2Office

Furthermore a graphic presentation of the addresses on a map is available (see figure 2):

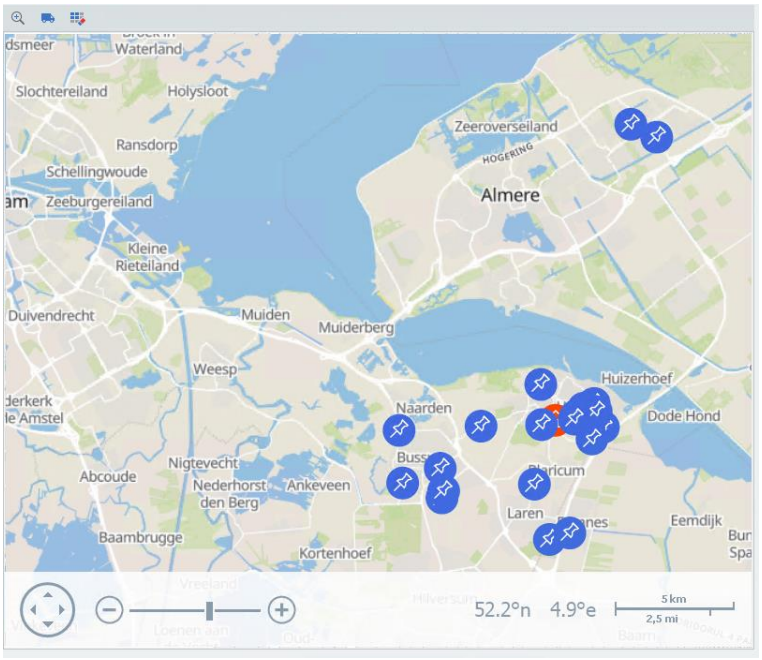


Figure 2: Graphical representation of address assignments for 1 interviewer (Link2Office)

Next step in this process is approval of these assignments, after which two separate things happen:

1. The addresses become available on the mobile device of Link2 via synchronization (see figure 3)
2. A custom made interface fills the CMA_ForWhom field in the Launcher database of Blaise CAPI. This enables synchronization to the mobile devices within the CMA application (see figure 4).

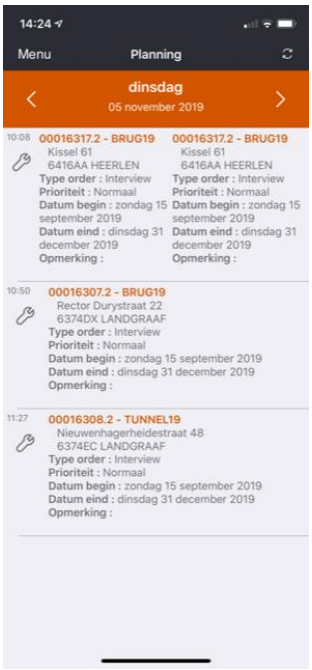


Figure 3: Addresses on Link2Mobile after synchronization

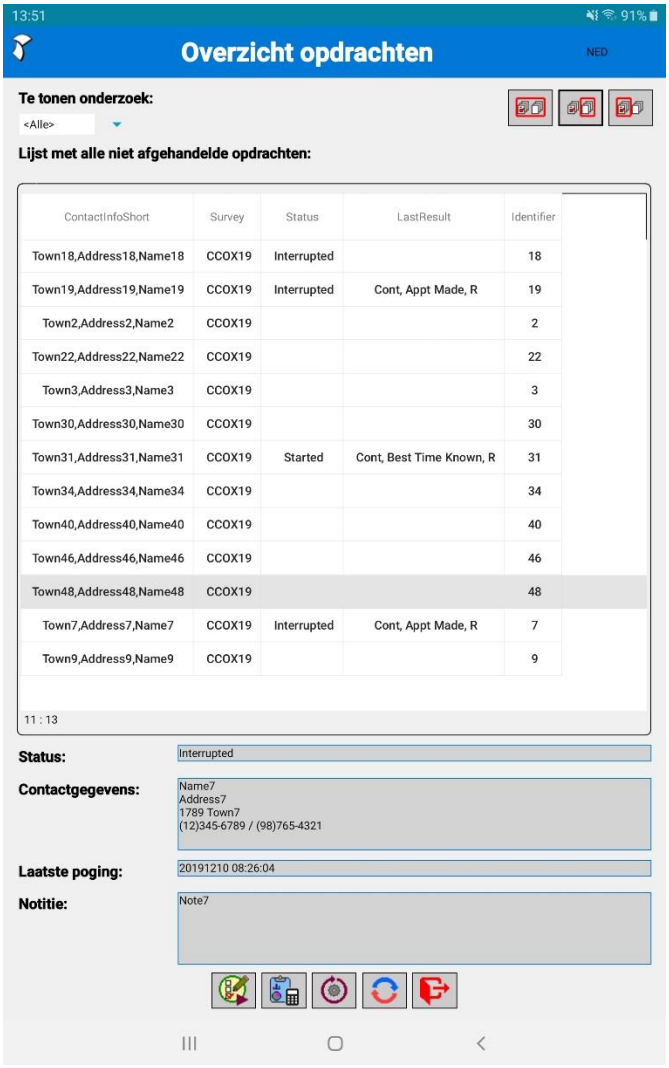


Figure 4: Overview of Addresses in CMA after synchronization

4. Process description: interviewing process

After these synchronizations to local devices the actual interviewing process can commence. As stated above (Business Case) Statistics Netherlands has chosen to use Link2Mobile to alleviate the interviewer's efforts in timekeeping and expenses. Once the interviewer has selected a specific address in his list, he can start measuring his travel time and distance by clicking (see figure 5): through GPS the road travelled is being kept by the system. After arrival at the desired address the system asks whether the kept number of kilometers is correct. By clicking on 'work' the worktime on the interview-task starts. Again, after clicking once more on 'work' the worktime on this task ends (see figure 6). At the end of an interview-task (regardless of the outcome) the task can be closed and reported to the central system by choosing the appropriate reason code (see figure 7)

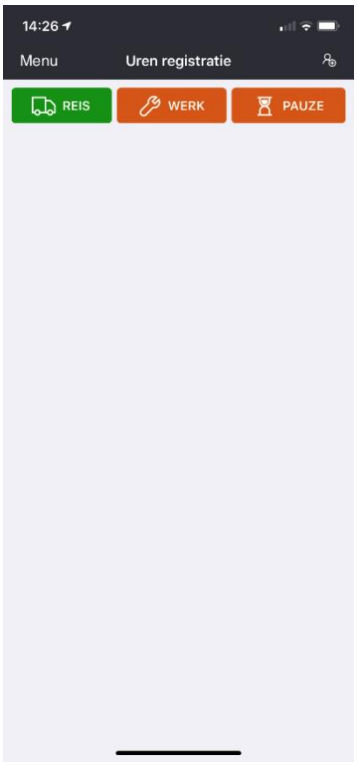


Figure 5: travel time keeping

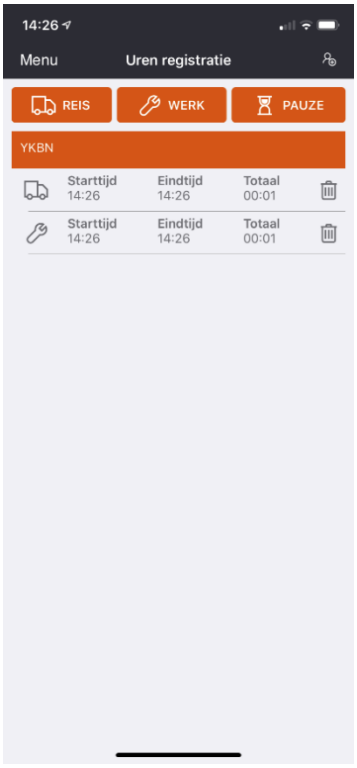


Figure 6: result time keeping

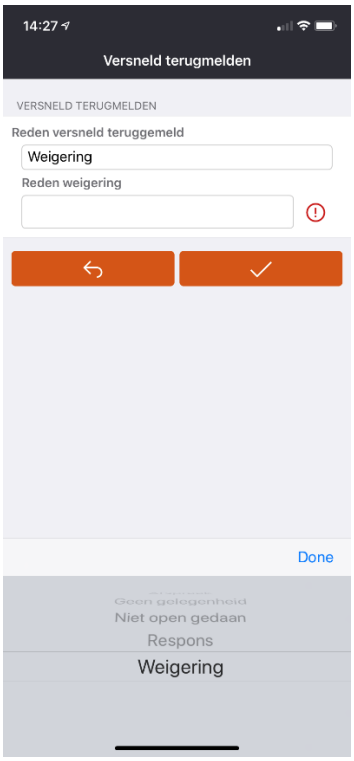


Figure 7: end result visit

Apart from the amount of time saved by using this functionality (and not having to repeat all administration at home at night), the additional advantage is that the interviewer can do all of this on his mobile phone, without having to open laptop or tablet. These bulkier devices are only needed once the respondent agrees to conducting the interview at this moment. In this case the interviewer starts (or opens) his interview device, opens CMA, selects the appropriate address (see figure 4) and clicks on the 'start interview'button (first on the left). After this, CMA opens the selected questionnaire (figure 8)

Figure 8: First page of consumer survey

5. Process description: synchronization of end results

Both Link2Mobile and CMA have standard (built in) functionality to synchronize the outcome of a visit to respectively LinkOffice and the Blaise Server Farm. From there the other standard components in the Phoenix landscape are notified (in the same way as it is done for CAWI and CATI channels).

One challenge remains: what happens to the assignments within CMA if a non-response happens at the door? In this case the interviewer hasn't even opened his CMA device and handles the non-response only on Link2Mobile (see figure 7).

CMA has a possibility to handle this: it is similar to the process of re-assigning addresses to another interviewer in case of illness of the original interviewer. In the central CMA database an address can be given an 'transfer required' status. Records with this status will be erased from the local device of the interviewer fallen ill. The only dependency in this process is that the first interviewer has to trigger a synchronization action. In case of illness the record will be transferred to another interviewer, in case of a non-response within Link2Mobile the record will receive an endstatus. Both endstatus and 'transfer required' status will be set through an API update on the central database.

6. Conclusion

Conceptually the combination of Blaise and CMA on the one hand and Link2 on the other serves the business needs as well as the architectural wishes of Statistics Netherlands. The CAPI channel will be implemented through the combination of two COTS products, with a minimum of custom made components. I am sure that we can communicate a successful implementation on the next IBUC.

DIM – Device Instrument Manager

Max Malhotra, University of Michigan Survey Research Center

1. Abstract

Using Blaise 5 for interviewer-administrated surveys pose unique technical challenges, some of these technical challenges include running offline distributed surveys. To address these challenges we developed an in-house application called Device Instrument Manager (DIM).

We required a mechanism that was robust where an interviewer could synchronize their laptop with a central repository and receive the components that are pertinent to them and be confident they could conduct offline interviews in a seamless manner and once their tasks were completed and they had internet access they can synchronize their data back to the main repository.

This paper will discuss two major components of the DIM and some of the subcommands.

2. Introduction to DIM

Prior to the development of DIM, we did not have a mechanism in place that could be coupled with an in house sample management system to provide a succinct user experience and to address the delivery and subsequent data collection which worked for both single-mode or mixed-mode environments.

This paper will discuss two major components of the DIM and some of the subcommands this application consists of such as.

1. Blaise Sync:
    a. Check Version
        i. Download Instruments
        ii. Perform Data Model Migrations
    b. Upload Cases
    c. Download Cases
2. Run Survey
    a. Run Pre App
    b. Start Survey
    c. Run Post App
    d. Pull Values

This paper will delve into how and why the above commands were required and will talk briefly about each command's implementation strategy, as well as go into our future plans for utilizing the write and download interceptors more extensively. We will also touch on some of the trials and tribulations we faced along the way until we got to a successful outcome.

There are also other subcommands that we will discuss in this paper that complement and help to facilitate a smooth flow and experience, those subcommands are not cupped with major commands such as Blaise Sync and Run Survey.

3. Background

Prior to the creation of DIM, we evaluated a number of design options that would best fit our organizational long term needs to determine what direction the design and development process will adhere too. Some of the Implementation Options and Top Concerns we had were:

1) Leverage native Blaise upload/download services.
    a) CBS is willing to provide instruction, but if SRO's needs do not align with the design of Blaise's services, CBS may not be willing and will not be able to customize their services to SRO's exact needs on SRO's timeline.
    b) Does this option push SRO toward long-term reliance on a single vendor?
2) Leverage native Microsoft Sync Framework (SQL Server synchronization utilities).
    1) The Sync Framework is a robust toolkit, but extensive technical design and development are required to tailor the tools to SRO's needs, which may prevent us from achieving full functionality on our timeline and within our requested budget.
3) Create SRO Blaise Sync Service from custom components, extending approach used with Blaise 4 and Blaise 5 with SurveyTrak.
    1) These technologies may not be the most efficient way to implement mixed-mode data transfer.
    2) The estimated effort may prevent us from achieving full functionality on our timeline and within our requested budget.

During the evaluation period we kept the following project scope in mind:

1) Start the survey on the server before downloading it.
2) Download new instruments to laptop.
3) Download a new case.
4) Resume survey on laptop after download.
5) Start survey on laptop after download.
6) Upload case data for a single case.
7) Upload case data for multiple cases.
8) Resume survey on the server after upload.
9) Download an existing instrument.
10) Download an existing case.
11) Delete case from the laptop (end of protocol).
12) Delete instrument from the laptop (end of project).

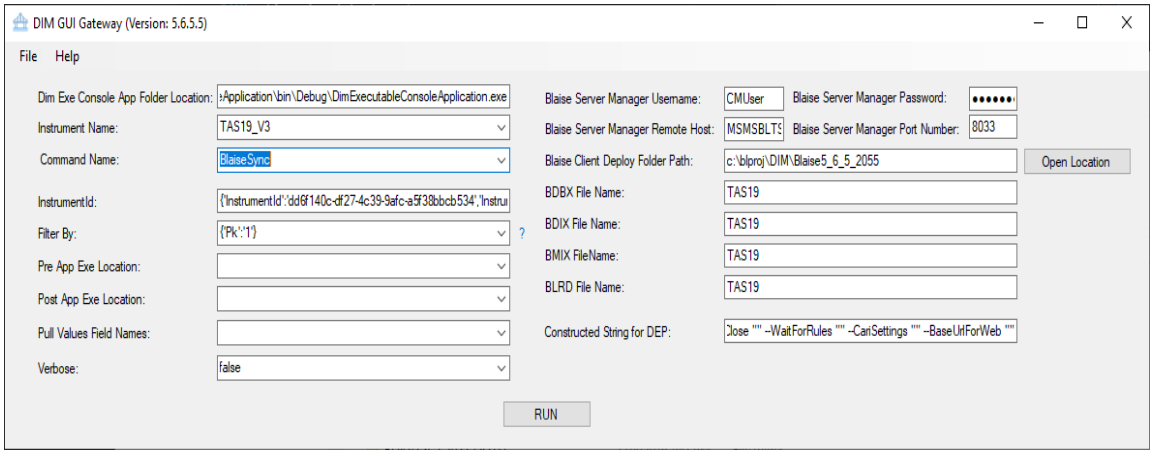
After we carefully evaluated each option it was decided that option 1 (Leverage native Blaise upload/download services) met our needs and timeline best.

At this point, DIM also leveraged the Start Kit Project and the Dep App and utilized them as a baseline during the development process.

4. Overview

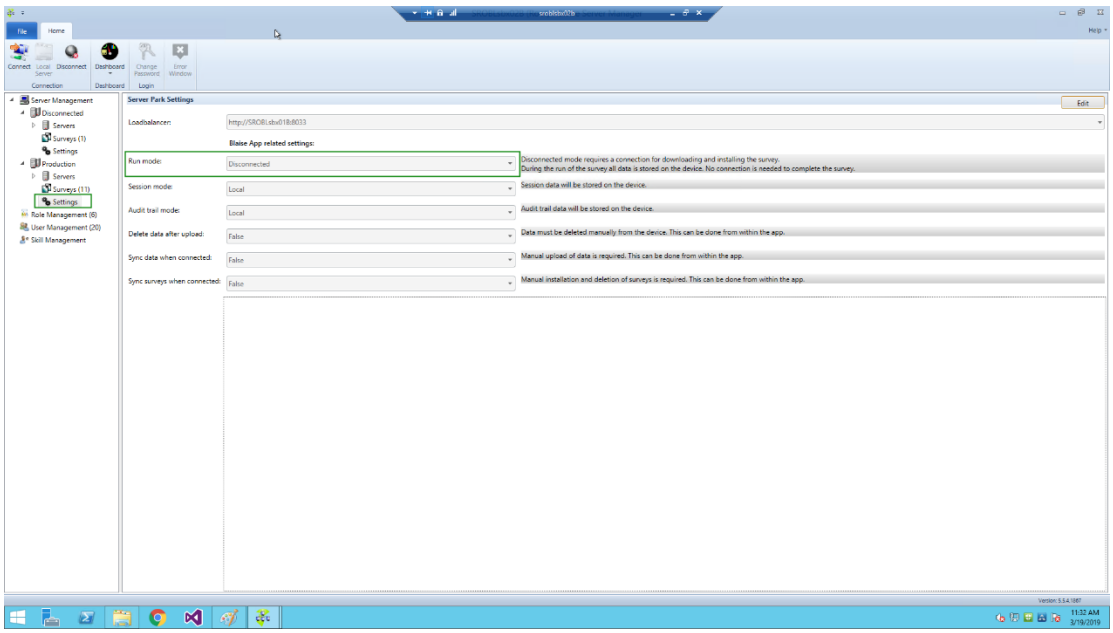
DIM is a C# .NET program that is the underlying core of how offline single-mode and mixed-mode cases are delivered to interviewers in our organization. DIM can be called completely independently for standalone testing purposes by utilizing a command-line interface and supplying the appropriate parameters for the command being executed.

During our development phase of the process, we took this a step further and developed a supplemental application called the DIM GUI Interface to serve as a bridge to help facilitate rapid testing and project integration while the in house sample management system, Michigan Sample Management System (MSMS) was being developed and downstream phases which allowed for a coupling between MSMS and DIM were being planned for implementation with DIM. During this phase the DIM GUI Application provided a Graphical User Interface for the DIM Executable Console Application this greatly aided in the development process and also helped to isolate any downstream bugs to either the DIM application (with the underlying Application Programming Interface (APIs) being utilized) or to the calling components of MSMS, this way we know where to allocate our resources to quickly address the issues being discovered.

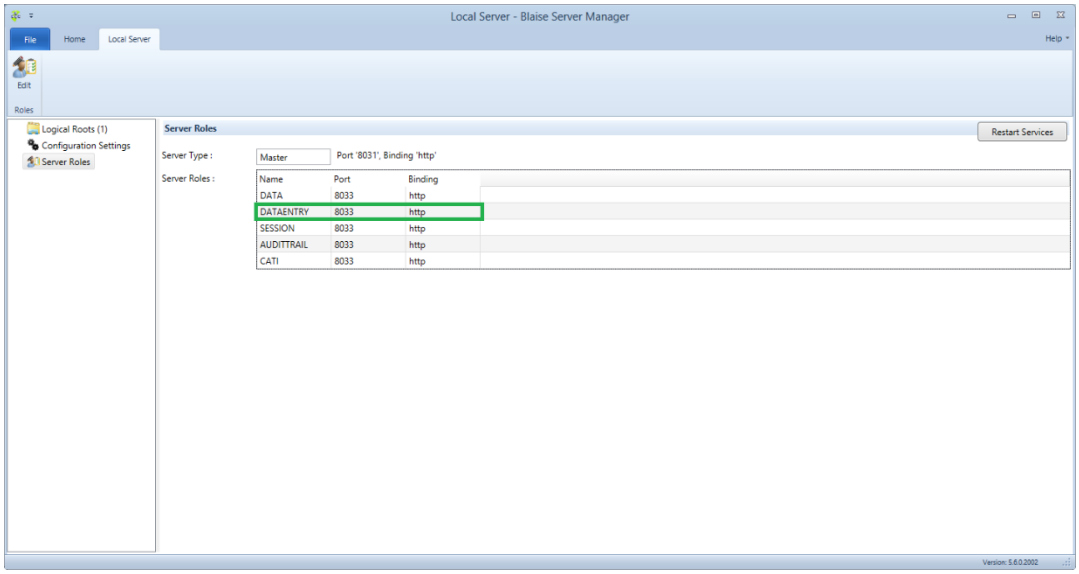


5. Prerequisites

- Run Mode on the Server Park in question MUST be set to Disconnected for DIM application to work. There is NO harm in setting it to Disconnected VS Client Server as Disconnected simply creates more files such as app.bpkg that DIM can utilize.



- On the **01** (Web server) > Right-click Blaise Server Manager > Run as Administrator > Click cancel on the prompt to log in > Click on Logical Servers > Click on Server Roles > Verify **Data Entry** role is present if not Edit and add one. Without this you cannot log in to the remote host of ***01 server.



- .NET Framework 4.7.2 or higher needs to be installed on the client machine
- Ideally, the setup project (Data Model) should be built in the version being tested and MATCH the NuGet **packages the** version of DIM is utilizing.
- Check that the latest version of SerialNumber.txt is present in the root directory of the DIM application otherwise DIM command calls such as Remove Cases Command that initiates the API to call datalink.Delete() will fail due to "Operation is only allowed with a valid license."
- Make sure that you have unchecked the Prefer 32-bit checkbox (for Both Debug and Release Configuration) at the project's Build tab (In Visual Studios).
    - Otherwise, you will get an SQL lite exception and will not be able to use AnyCpu and then forced to use x64.

6. DIM Commands – Inputs and Outputs

This section describes what comprises each of the individual DIM commands and how to initiate the command and what a typical JSON output result will look like for that command.

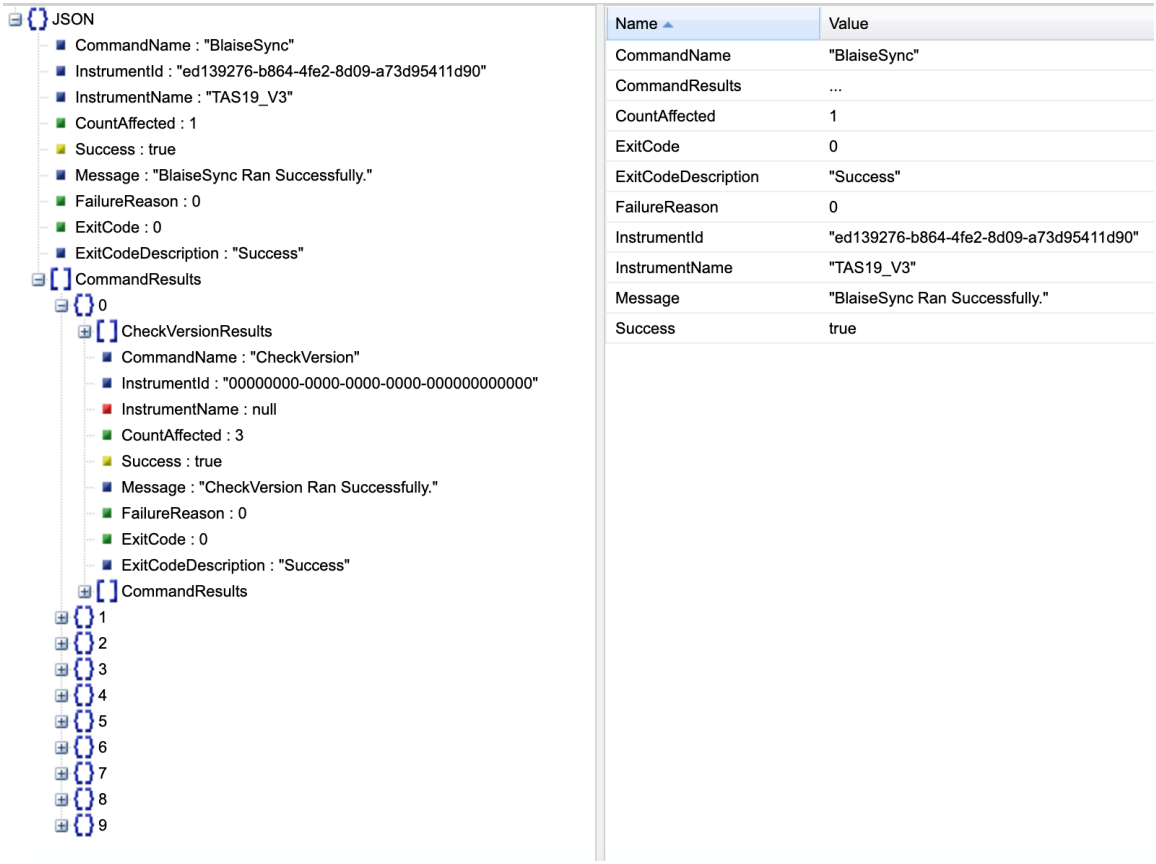
- **Blaise Sync Flow**

    Blaise Sync
    Check Version
        string **InstrumentId**
        string **InstrumentName**

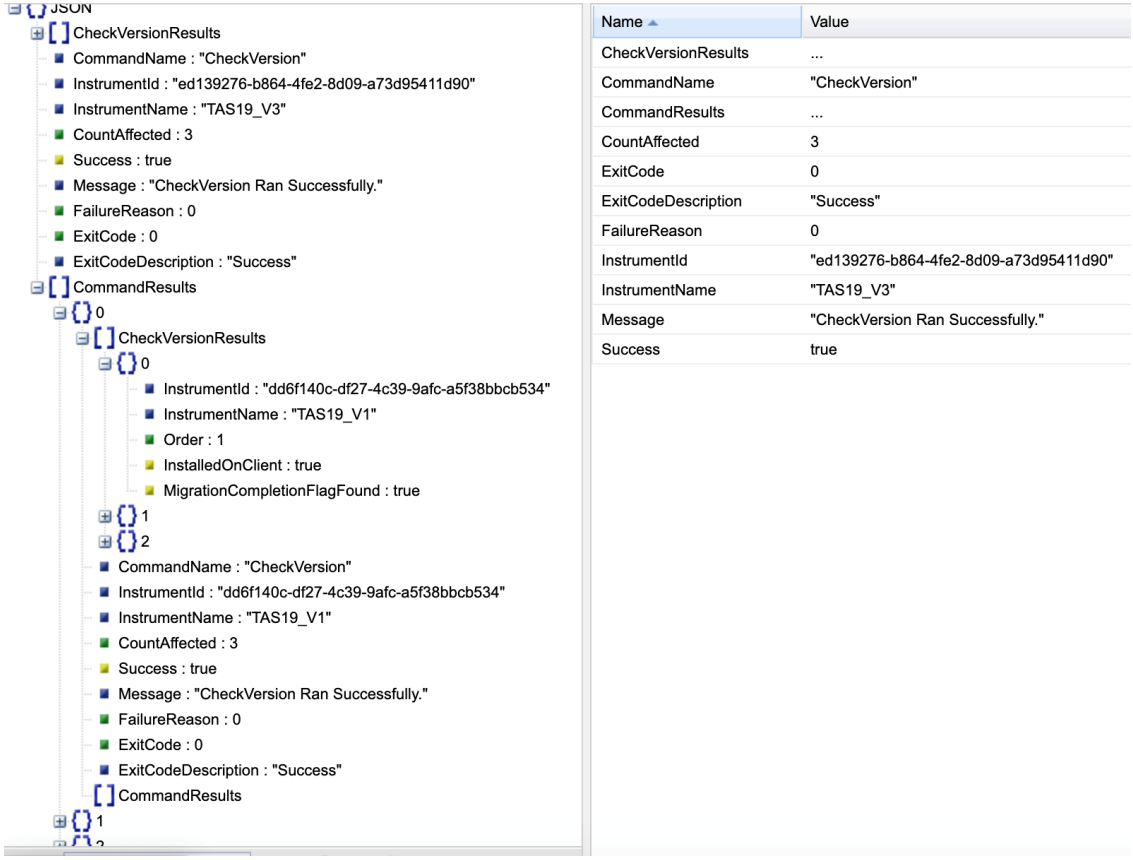
int **Order**
bool **InstalledOnClient**
bool? **MigrationCompletionFlagFound** (is a nullable bool)

Upgrade Instrument - If needed based on Check Version
Download Instrument - Download All instruments requested first.
Data Migration - If a **MigrationCompleted.txt** file does not exist for that DM
UploadloadCases - On the LAST instrument - Need to Loop through
DownloadCases - On the LAST Instrument - Need to Loop through

- **Blaise Sync Command**
  - **Command Prompt**:
  - .\DimExecutableConsoleApplication.exe" -c "BlaiseSync" -i "{'InstrumentId':'dd6f140c-df27-4c39-9afc-a5f38bbcb534','InstrumentName':'TAS19_V1','Order':'1'},{'InstrumentId':'a9eef537-7109-4a0f-bad4-a59ae5ea35b8','InstrumentName':'TAS19_V2','Order':'2'},{'InstrumentId':'ed139276-b864-4fe2-8d09-a73d95411d90','InstrumentName':'TAS19_V3','Order':'3'}" -f "{'Pk':'1'}" -r "MSMSBLTSTNXT02.isr.umich.edu" -u "CMUser" -p "XXXXXXXXXX" -n "8033" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19"
  - **OUTPUT:**



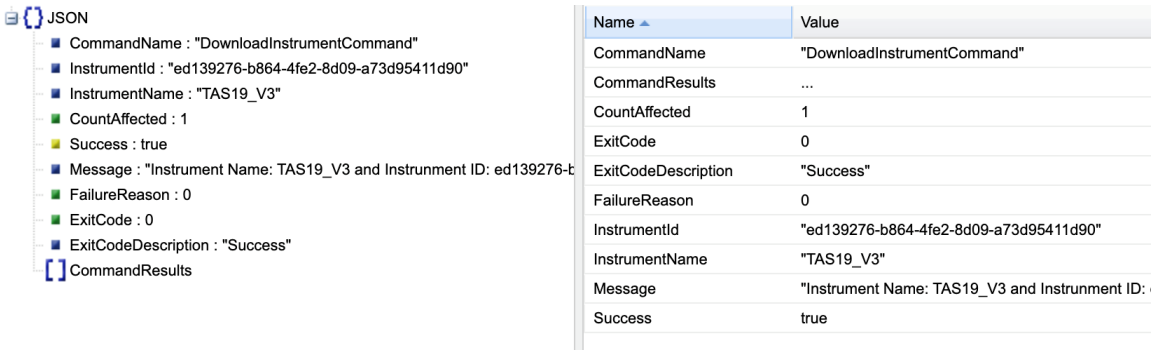
- **CheckVersion Command:**
  - **Command Prompt**:
    - .\DimExecutableConsoleApplication. -c "CheckVersion" -i "{'InstrumentId':'dd6f140c-df27-4c39-9afc-a5f38bbcb534','InstrumentName':'TAS19_V1','Order':'1'},{'InstrumentId':'a9eef537-7109-4a0f-bad4-a59ae5ea35b8','InstrumentName':'TAS19_V2','Order':'2'},{'InstrumentId':'ed139276-b864-4fe2-8d09-a73d95411d90','InstrumentName':'TAS19_V3','Order':'3'}" -f "{'Pk':'1'}" -v "false" -r "MSMSBLTSTNXT02.isr.umich.edu" -u "CMUser" -p "XXXXXXXXXX" -n "8033" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19"
  - **OUTPUT:**



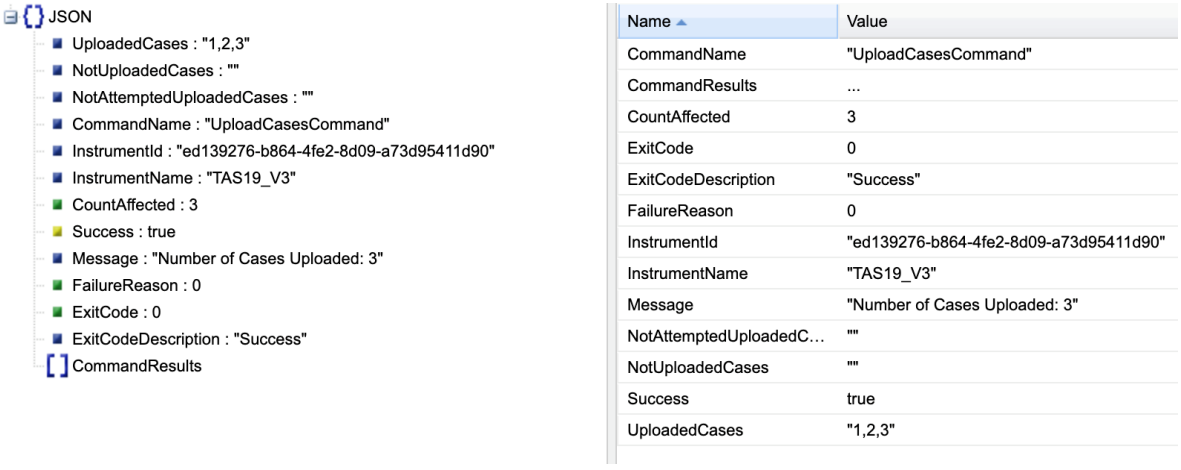
  - **Return Object**

    string **InstrumentId**
    string **InstrumentName**
    int **Order**
    bool **InstalledOnClient**
    bool? **MigrationCompletionFlagFound**

- **Download Instrument Command:**
    - **Command Prompt**:
        - .\DimExecutableConsoleApplication.exe -c "DownloadInstrumentCommand" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -r "MSMSBLTSTNXT02.isr.umich.edu" -u "CMUser" -p "XXXXXXXXXX" -n "8033" -d "c:\blproj\DIM\Blaise5_6_5_2055"
    - **OUTPUT:**



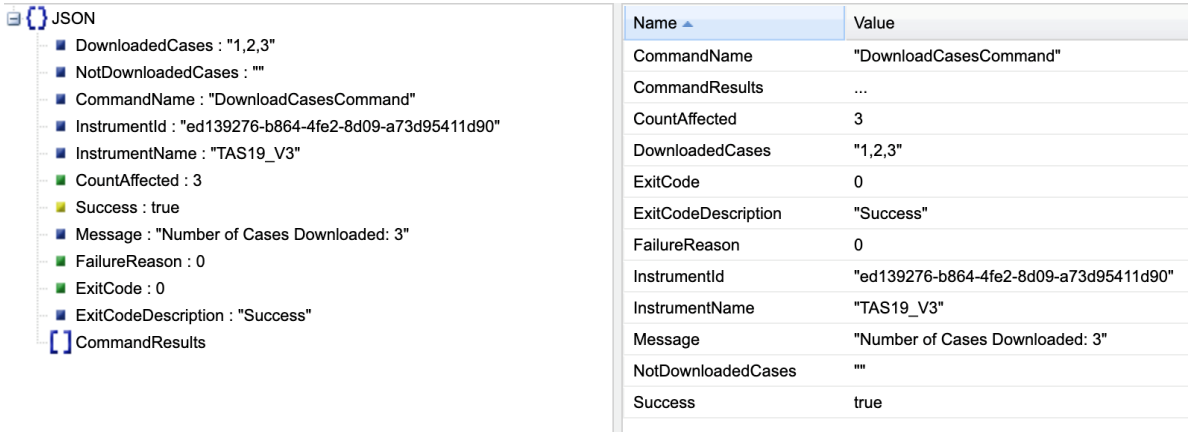
- **Upload Cases Command**
    - **Command Prompt**:
        - .\DimExecutableConsoleApplication.exe -c "UploadCasesCommand" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -f "{'Pk':'1'},{'Pk':'2'},{'Pk':'3'}" -r "MSMSBLTSTNXT02.isr.umich.edu" -u "CMUser" -p "XXXXXXXXXX" -n "8033" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19"
    - **OUTPUT:**



- **Download Cases Command**
    - **Command Prompt**:

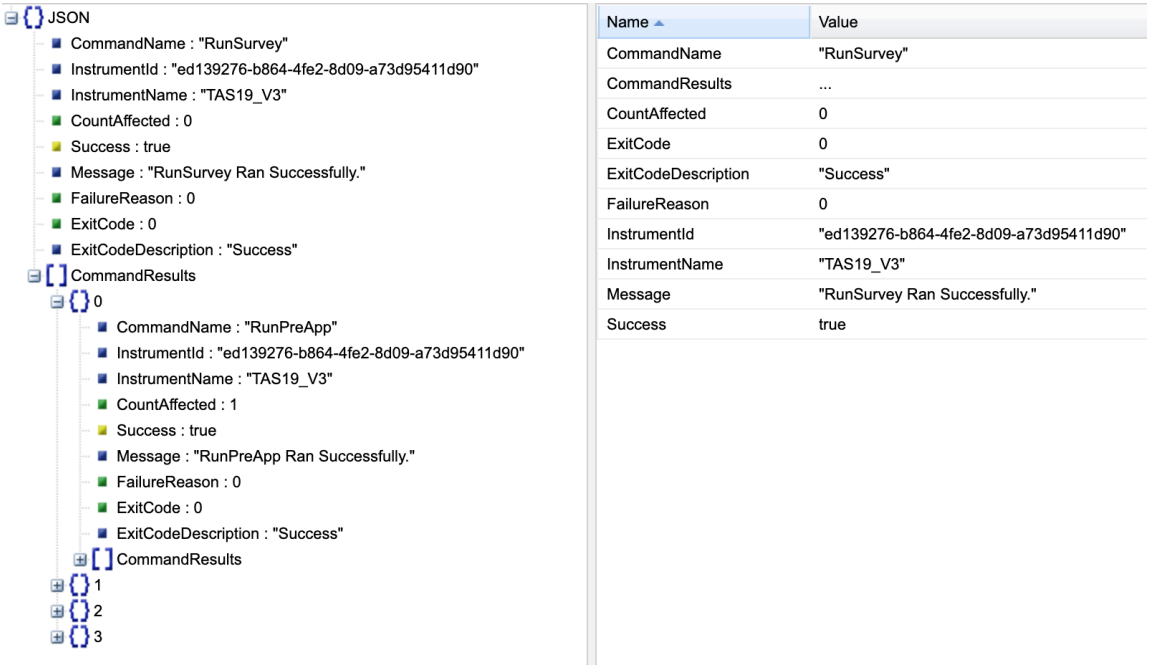
- .\DimExecutableConsoleApplication.exe -c "DownloadCasesCommand" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -f "{'Pk':'1'},{'Pk':'2'},{'Pk':'3'}" -r "MSMSBLTSTNXT02.isr.umich.edu" -u "CMUser" -p "XXXXXXXXXX" -n "8033" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19"
    - Note: Download Cases Command requires that -f (FilterArgument) is present to execute.

- **OUTPUT:**

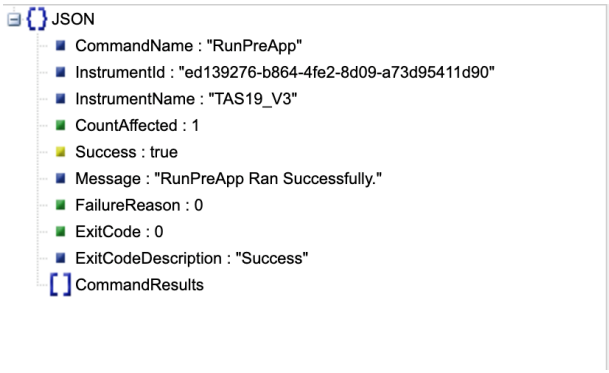


- **Run Survey Command**
    - **NOTE:**
        - Commands run in this sequence:
            - RunPreApp
                - optional pram of --PreAppExeLocation "C:\Windows\System32\notepad.exe", you do not have to pass it in the below command at all **or** you can pass an --PreAppExeLocation "" if you want to skip it).
            - StartSurveyCommand
            - RunPostApp
                - optional pram of --PostAppExeLocation "C:\Windows\System32\notepad.exe", you do not have to pass it in the below command at all **or** you can pass an --PostAppExeLocation "" if you want to skip it).
            - PullValues
                - If you want to skip this command you can pass an empty string such as: --PullValuesFieldNames "" if that is done then the command will still return a 0 value and return as a success since it presumes you meant to pass nothing and skip it.
    - **Command Prompt**:

- .\DimExecutableConsoleApplication.exe" -c "RunSurvey" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -f "{'Pk':'1'}" --PreAppExeLocation "C:\Windows\System32\notepad.exe" --PostAppExeLocation "C:\Windows\System32\notepad.exe" --PullValuesFieldNames "{'Fn':'Field1'}" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19" --LayoutSetGroup "Interviewing" --LayoutSet "Large" --DataEntrySettings "StrictInterviewing" --Language "" --Fields "xTimeGate=1" --AssignMode "Always" --StartParallel "" --RunMode "ThickClient" --InitialWindowState "Maximized" --EnableResize "False" --EnableClose "False" --WaitForRules "" --CariSettings ""
- **OUTPUT:**

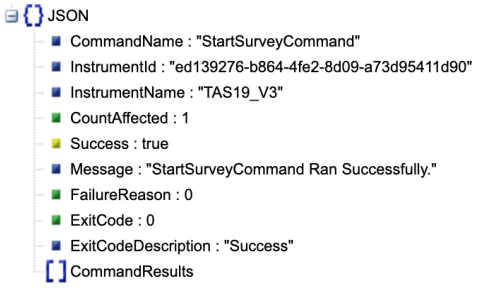


- **RunPreApp Command**
  - **Command Prompt**:
    - .\DimExecutableConsoleApplication.exe -c "RunPreApp" -i "ed139276-b864-4fe2-8d09-a73d95411d90" --PreAppExeLocation "C:\Windows\System32\notepad.exe" -d "c:\blproj\DIM\Blaise5_6_5_2055"
  - **OUTPUT:**



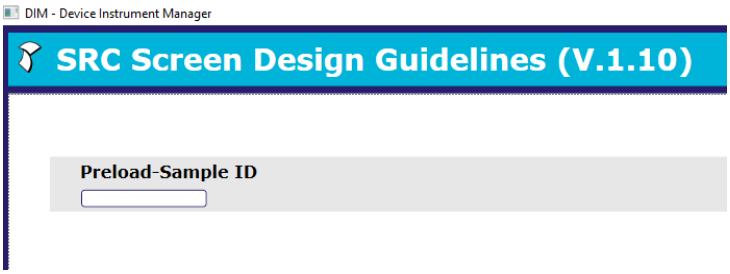
Name	Value
CommandName	"RunPreApp"
CommandResults	...
CountAffected	1
ExitCode	0
ExitCodeDescription	"Success"
FailureReason	0
InstrumentId	"ed139276-b864-4fe2-8d09-a73d95411d90"
InstrumentName	"TAS19_V3"
Message	"RunPreApp Ran Successfully."
Success	true

- **Start Survey Command**
  - **Command Prompt**:
    - .\DimExecutableConsoleApplication.exe -c "StartSurveyCommand" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -f "{'Pk':'1'}" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19" --LayoutSetGroup "Interviewing" --LayoutSet "Large" --DataEntrySettings "StrictInterviewing" --Language "" --Fields "xTimeGate=1" --AssignMode "Always" --StartParallel "" --RunMode "ThickClient" --InitialWindowState "Maximized" --EnableResize "False" --EnableClose "False" --WaitForRules "" --CariSettings ""
  - **OUTPUT:**

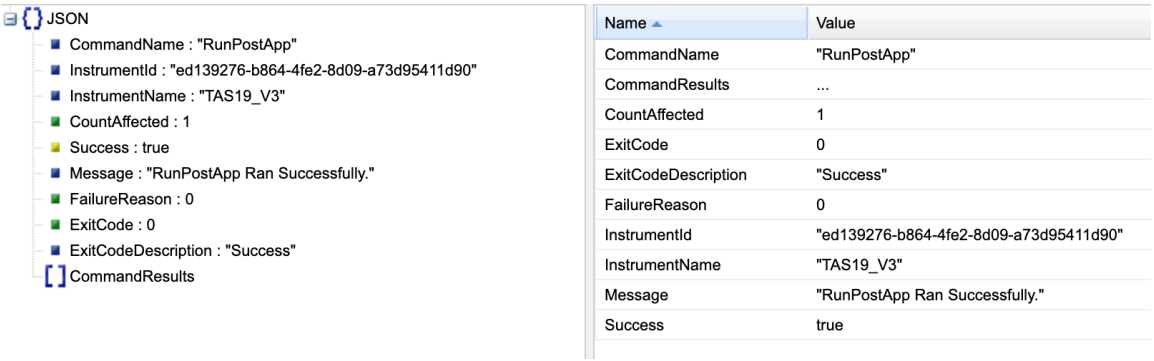


Name	Value
CommandName	"StartSurveyCommand"
CommandResults	...
CountAffected	1
ExitCode	0
ExitCodeDescription	"Success"
FailureReason	0
InstrumentId	"ed139276-b864-4fe2-8d09-a73d95411d90"
InstrumentName	"TAS19_V3"
Message	"StartSurveyCommand Ran Successfully."
Success	true

- If no value is supplied generally a generic Blaise page will come up which will ask you for your key value. (This is not advisable)
  - Example of Command: .\DimExecutableConsoleApplication.exe -c "StartSurveyCommand" -i "ce0c7770-c3a1-44c7-b804-7ed0e11d37a7" -f ""

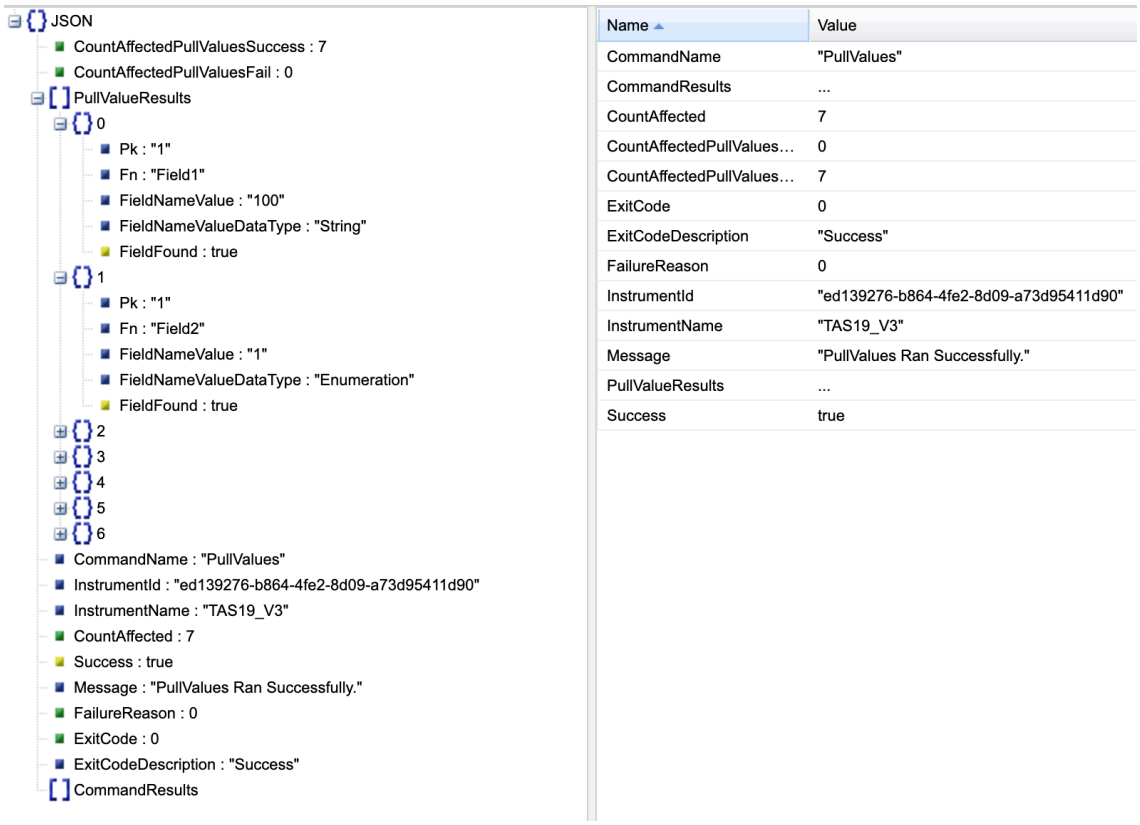


- **RunPostApp Command**
  - **Command Prompt**:
    - .\DimExecutableConsoleApplication.exe -c "RunPostApp" -i "ed139276-b864-4fe2-8d09-a73d95411d90" --PostAppExeLocation "C:\Windows\System32\notepad.exe" -d "c:\blproj\DIM\Blaise5_6_5_2055
  - **OUTPUT:**

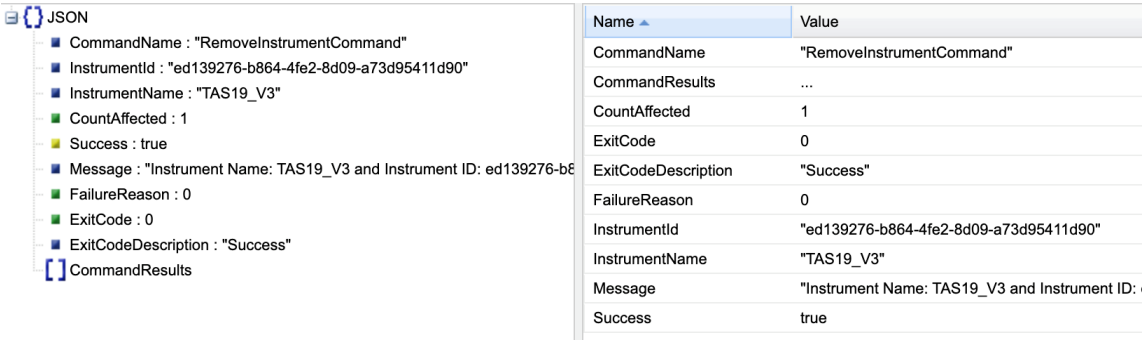


Name	Value
CommandName	"RunPostApp"
CommandResults	...
CountAffected	1
ExitCode	0
ExitCodeDescription	"Success"
FailureReason	0
InstrumentId	"ed139276-b864-4fe2-8d09-a73d95411d90"
InstrumentName	"TAS19_V3"
Message	"RunPostApp Ran Successfully."
Success	true

- **Pull Values Command**
  - **Command Prompt**:
    - .\DimExecutableConsoleApplication.exe" -c "PullValues" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -f "{'Pk':'1'}" --PullValuesFieldNames "{'Fn':'Field1'},{'Fn':'Field2'},{'Fn':'Field3'},{'Fn':'Field4'},{'Fn':'Field5'},{'Fn':'Field6'},{'Fn':'Field7'}" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19"
  - **OUTPUT:**

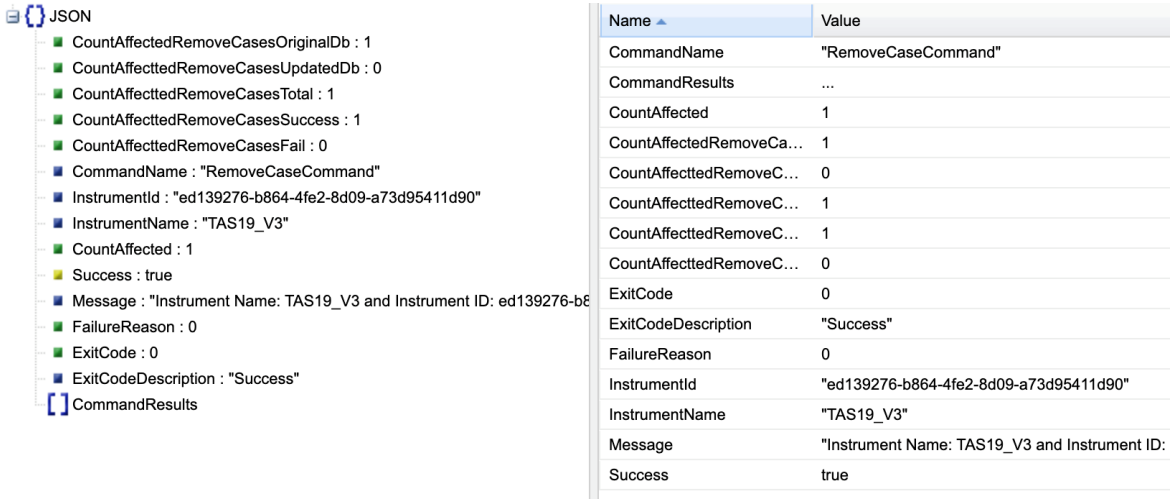


- **Remove Instrument Command:**
  - **Command Prompt**:
    - .\DimExecutableConsoleApplication. -c "RemoveInstrumentCommand" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19"
  - **OUTPUT:**



- **Remove Case Command:**
  - **Command Prompt**:

- .\DimExecutableConsoleApplication. -c "RemoveCaseCommand" -i "ed139276-b864-4fe2-8d09-a73d95411d90" -f "{'Pk':'1'}" -d "c:\blproj\DIM\Blaise5_6_5_2055" --InstrumentName "TAS19_V3" --BdbxFileName "TAS19" --BdixFileName "TAS19" --BmixFileName "TAS19" --BlrdFileName "TAS19"
  - **OUTPUT:**



7. DIM Commands Explained

7.1 Check Version

In the check version command, DIM examines what versions are currently installed on the interviewer laptop machine and compares it to what versions are being requested from the server.

Then check version downloads any instruments in sequence (future versions) that are not currently present on the interviewer laptop.

Finally check version command checks what is the current versions of the Data Model (DM) are installed on the interviewer's laptop, then migrates the data from the current version to each of the new future versions installed on the laptop in the sequence provided a previous migration had not taken place.

7.2 Upload Cases

Uploads the requested cases from the interviewer laptop to the server. When it makes it to the server the Write Interceptor gets triggered server-side and performs the necessary logic to based on business rules and ultimately moves the case to the appropriate database.

The audit data for the cases specified is automatically removed upon successful upload.

7.3 Download Cases

Downloads the requested cases from the server to the interviewer laptop. Currently, we are not utilizing the Download Interceptor but has the potential to be called in future versions where business logic will be run prior to the downloaded cases making it to the appropriate database on the interviewer laptop.

7.4 Run Pre App | Run Post App

The Run Pre App and Run Post App can call external applications prior to and after the survey are conducted. This can be used to create applications that perform complex logic outside of the data model to perhaps spawn new lines based on a preview respondent answer or any other logic that is desired and can be programmed.

The Run Pre App and Run Post app expect an application-level Exit Code of 0 to consider it a success otherwise DIM will break out of the application.

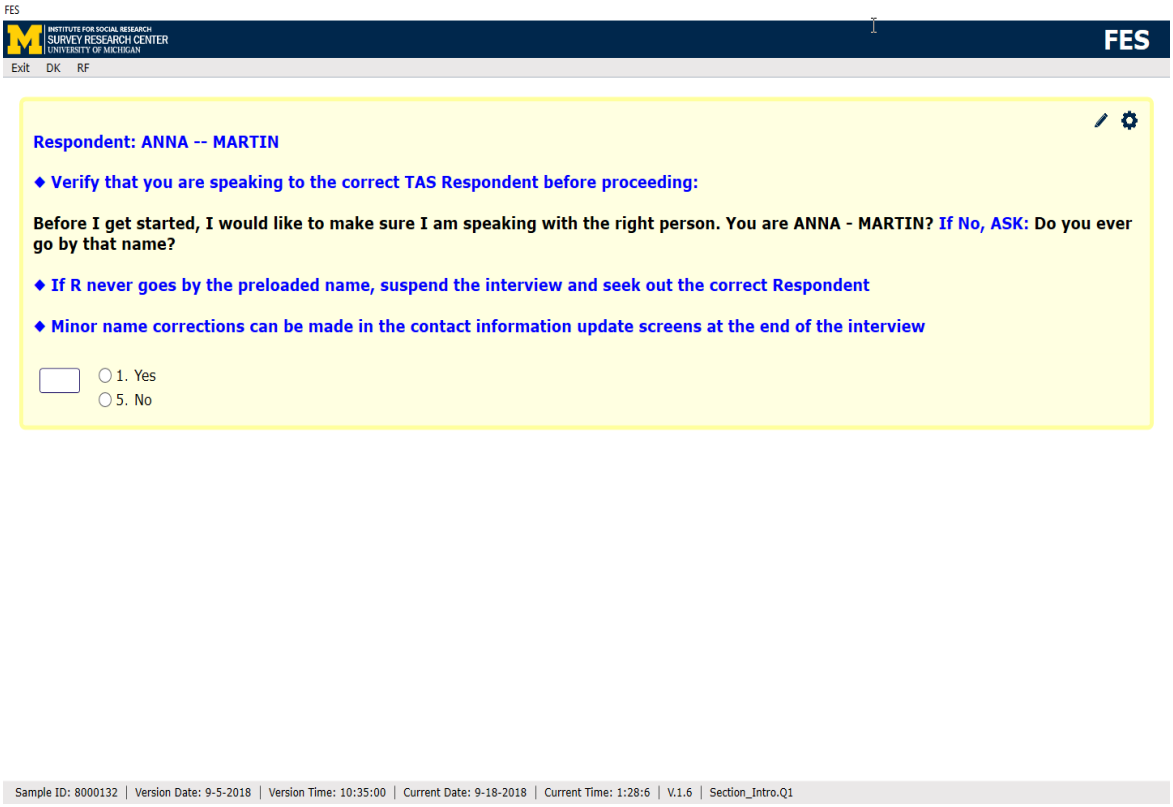
7.5 Start Survey

The Start Survey command launches the custom DEP and launches the instrument.

The custom dep window is designed to check for error handling so if the instrument returns an error internal to the instrument it will return that back to DIM so it can perform the appropriate logic.

The custom dep is also accounting for event handlers such as aborted, completed, and so forth. The custom dep is also set up to handle desired parameters such as always launching maximized and not showing the close button to the interviewer so we can internally control the interviewer experience.

An example of the custom dep can launch an instrument can be seen below.



7.6 Pull Values

The Pull Values command returns back all the values for the fields being requested for a specific SampleId. It will also tell you the field data type and also let you know if that field is present inside the database.

7.7 Download Instrument

The Download Instrument command is capable of downloading one or many instruments in the order supplied.

7.8 Remove Instrument

The Remove Instrument command removes the specific instrument specified and its associated files.

7.9 Remove Cases

The Remove Cases command removes the specific cases specified and associated session data for that case.

8. DIM Application Usage – End Users

8.1 Intended Audience of the Application

The target audience/end-user of this application would be personnel who administer Computer Assisted Telephone Interviews (CATI) and Computer Assisted Personal Interviewing (CAPI). In the case of our organization, this means both in-house -- our Survey Services Lab (SSL), and field personnel, depending on project need, the majority of the use will be for CAPI.

8.2 Michigan Sample Management System (MSMS)

DIM was designed to be able to be coupled within a house sample management systems, in fact, we built it with this in mind and thus it lends itself well to this purpose as each command and subcommand is designed to provide appropriate notifications back in JSON format to the calling system/application.

The calling system/application will take notifications and perform the necessary business logic that is appropriate for the organization. Time/resources and effort are required to make this coupling happen in a seamless manner and have DIM fully integrate with the in-house sample management system.

9. Trials and Tribulations

We have faced many hurdles along the way and continue to work through issues in conjunction and in cooperation with CBS. Issues such as the Write Interceptor not causing a lockout condition where excessive system memory usage was detected, if this issue cannot be resolved quickly, we will have to reexamine our system workflow for when certain conditions are met such as in the case of mixed-mode interviews.

In the past, we discovered issues with how some APIs did not function the way we desired such as the deploy folder always having a static location and we required dynamic capabilities for that, and other parameters and CBS were able to accommodate in future versions.

We also ran in too much more complex issues to diagnose such as when DIM called the Start Survey Command and launched an instrument the majority of the system instrument looked correct however some lookup tables would behave oddly and not give the desired result. After some back and forth CBS was able to resolve an underlying API problem.

We have countless examples of the above items but the major take away is that it took time and effort and a lot of back and forth but with the help of CBS we were able to resolve the issues as they were discovered.

10. Conclusion

The DIM application was developed to provide a mechanism to conduct single and mixed-mode interviews for offline distributed surveys. The goal was to create a back-end application with the flexibility to serve the needs of many projects without increasing a burden on interviewers and Blaise programmers.

DIM plays an integral role in our organization in facilitating Computer Assisted Interviewing (CAI) studies. It is used to conduct interviews and provide for succinct user experience to the interviewers were

the complexity of synchronization, data migration and data transfer are neutralized from the interviewer viewpoint and the necessary tools are provided to the interviewer to succeed in the task of collecting single-mode and mixed-mode surveys seamlessly. Data validity during the synchronization process is crucial to provide assurance of proper data integrity through the entire system and DIM provides this assurance through intensive code reviews and regression testing conducted.

DIM has been built out using an approach that lends itself well for future development and growth wherein each of its commands and subcommands can be quickly coupled together to produce new commands and has the ability to leverage core Blaise components to keep pace with Blaise development wherein each new version of Blaise is linked to a new version of DIM allowing for DIM to leverage changes made inside the Blaise .dlls though the use of NuGet packages.

In closing, DIM serves as an important utility in our organization for Computer-Assisted Personal Interviewing (CAPI) and on our end we are continuing to put the DIM application through its paces by utilizing new data models as they become available and we continue to try to improve upon it as new feature requests are made.

It has been a long road getting to this point due to running into difficulties and due to project scope design changes such as deciding to implement Blaise Sync using the Record Filter approach as oppose to an more automated Blaise sync using a offline CAPI block in the Blaise database as the driving factor, as well as other various hurdles along the way, but what we can say is that the CBS team has provided excellent support every step of the way and working together we were able to overcome numerous challenges and come out with a viable product that is used in production projects.

In the future, we wish to develop additional functionality that allows for DIM to be further streamlined for deployment. We also wish to work closely with the CBS team to help resolve issues we have encountered with the Write Interceptor as that would be a critical path division if we do not come up with a resolution strategy.

11. References

Blaise 5 Help. (2020). ‘http://help.blaise.com’
Blaise 5 Starter Kit Project. (2019).
Blaise 5 Dep App. (2019).
University of Michigan Survey Research Center. (2019). ‘Blaise 5 Integration Documentation’

Data Collection Management System in Statistics Finland

Pyry Keinonen, Joonas Salmi, Heikki Leino and Petri Godenhjelm Statistics Finland

1. Abstract

This paper discusses how Data Collection Management System was developed in Statistics Finland and in which way it is used in practice. Data Collection Management System was developed in 2017 to 2019 and was implemented to production in January 2019. System uses Blaise5 as primary Data Collection System.

Data Collection Management System in Statistics Finland consists of the primary data collection management service and from two smaller sub-services. The main service provides tools for survey, sample and case management and monitoring of data collection. One of the sub-services provides the user interface for case and interview management for interviewers, and one provides the infrastructure and API for web-survey data collection. Together these services provide the necessary functions for a multi-mode data collection process.

Statistics Finland has had a need to develop a data collection management system because the old operating environment has a lot of manual work stages that require effort. These workflows are largely built around the operating models of the Blaise4 production environment. The development of the Data Collection Management System has enabled production of reports on data collection and comprehensive management of multi-mode data collection. Most importantly, it enables online and offline interviewer data collection at the same time as online data collection.

The Data Collection Management System is built to utilize Blaise5 as a data collection tool for online and interview data collection. Therefore, following the Blaise5 Evolution path is crucial because the effects are directly reflected in the definition and implementation of the system development needs.

Future development work may include implementation of the Blaise5 CATI system and enterprise data collection features. In addition, the development of utilizing geographic data and information into the system to allocate cases to interviewers and reduce logistic costs is underway. In the future, the data collection management system will be used for all personal data collection and the Blaise4 production environment is being driven down.

2. Evolution of multi-mode data collection in Statistics Finland

Statistics Finland's data collection system on household surveys has long been Blaise4, which is mainly used by interviewers for field (CAPI) and telephone (CATI) data collection. In addition, the organization has experience with the Blaise IS system in web (CAWI) data collection. The transition from Blaise4 to Blaise5 began in 2015. Since then, the web data collection has been progressively implemented and migrated to the Blaise5 environment. The same year organization started a set of projects to develop a multi-mode data collection system and infrastructure. These projects were completed in January 2019 when Data Collection Management System was successfully implemented in production.

2.1 Early phases of multi-mode data collection

The first steps towards multi-mode data collection in Statistics Finland were taken in 2015 when the survey “Use of information and communications technology by individuals” was created with Blaise5. The pilot data collection was conducted the same year and the first production data collection was

conducted in 2016. In practice the web data collection was collected by using Blaise5 and all other data collection management were provided with the data collection processes built around Blaise4.

Alongside these first steps, Statistics Finland began developing a multi-mode data collection system and infrastructure in 2015 to fully migrate to Blaise5. The system was developed in 2015-2016 and the first pilot was conducted in 2016-2017 but had to be discontinued due to technical problems. The first data collection used in the pilot was “Labor force survey”.

Despite the encountered technical problems in the pilot the “Adult education survey” data collection was carried out by using the multi-mode system in 2017. After finishing the first data collection it was decided that the current development needed a lot of redefining and reconstruction based on the experiences gained and the implementation project was put to a halt. This meant that transition from Blaise4 to Blaise5 was delayed and multi-mode data collection could only be done by combining the existing Blaise4 production processes with standalone Blaise5 web data collection processes.

3. Data Collection Management System

After gaining a lot of experience from the previous multi-mode project, a new project was launched in 2017 under a code name Ruuti. The organization set four aims for the Ruuti system which were reliability, usability, unity and flexibility. These aims were based on the idea that the system should be functional and reliable but also easy to use in both managing the data collection and performing the interviews. One importance was unifying the working tools, processes and methods of data collection including automatization of manual work phases. Also, the possibility to combine multiple data collection modes and software was considered an important feature.

3.1 System architecture

The core part of Ruuti system is Data Collection Management Service called Mixeri. Its primary function is to handle the data collection entity and communicate with linked sub-services. Currently there is two sub-services or channels which are Web Data Collection Service and Field Data Collection Service. Other channels are possible to be added later such as CATI channel. From communication point of view Mixeri is a client and different channels are each one’s servers.

The integration of the Ruuti system into the statistical production process takes place exclusively through the Ruuti system. If necessary, Mixeri and all channels can be placed in different domain areas of the infrastructure.

The channels are independent and each one is dedicated to only one data collection mode. All channels have their own dedicated databases and are responsible for their own special features which Mixeri does not know about. These features are for example the Blaise environmental duplication in Web Data

Collection Service and Offline capabilities in Field Data Collection Service. Blaise integration is done only within the channels.

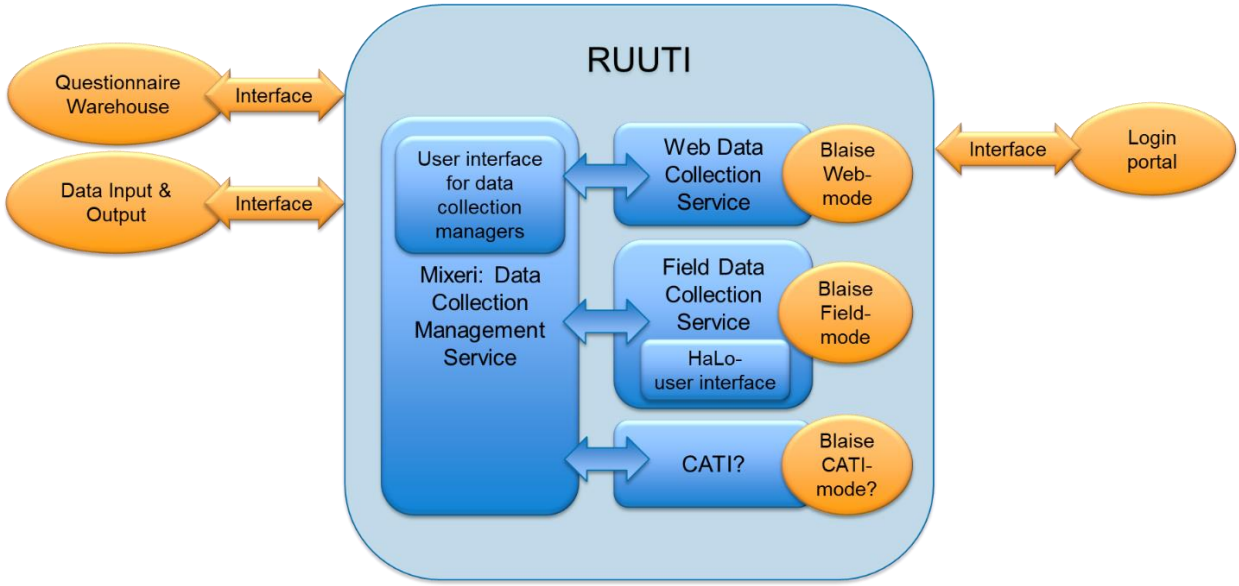


Figure 3: Ruuti Data Collection Management System

3.1.1 Mixeri – Data Collection Management Service

Mixeri is the central part of Ruuti System and it contains the knowledge of all data collections and cases and their status. Mixeri includes data collection establishment and management, case and sample data management including distribution, Blaise package distribution and management, interviewer resource management, and the management of data collection monitoring. It is a tool for data collection managers.

Mixeri communicates and synchronizes information necessary to channels. It enables the distribution of case data from all surveys to all interviewers and handles the multi-mode collected case data automatically. In practice this means that a case can be handled simultaneously in all data collection modes. In addition, a real-time overview of the progress of data collection and the ability to react to the situation during the collection is possible.

The most important user-side features are the possibility to create and monitor data collections and manage it even on individual data collection or individual interviewer resource level. Also, the distribution of cases to interviewers or from one interviewer to another is possible.

Together with Mixeri, Blaise, Web Data Collection Service and Field Survey Management Service provide the essential tools which makes a multi-mode data collection possible.

3.1.2 Blaise

Ruuti system utilizes Blaise5 as primary tool for data collection through Application Programming Interface. Blaise Server installation is used both in dedicated server for web data collection and individual interviewers' computer. Collected field survey data is stored in Blaise Database in interviewer's computer and web survey data is stored into SQL-server. In interviewer's PC the Field Survey Management Service handles the collected field survey data and synchronizes the data to Mixeri when interviewer is online

while Web Data Collection Service synchronizes the collected web survey data straight to the Mixeri's database. In both situations the initial process starts when Blaise Event triggers.

In case of the the survey package is changed in the middle of the ongoing data collection, Mixeri handles the possible data conflict situations. For example, if a new installed Blaise Survey package does not include a variable which is collected with the previously installed package then Mixeri keeps the previously collected variable data but does not pass it to Blaise any longer.

3.1.3 HaLo – Field Data Collection Service

HaLo is a Field Survey Management Service alias Channel and it provides the platform, communication between Mixeri and user interface for interviewers. It is installed locally to interviewer's workstation. HaLo completes the Blaise-integration by utilizing Blaise's API libraries. This integration includes starting the survey interview, case data pre-filling and reading the collected data after the survey session has ended. It also handles the installation of Blaise-packages and launches the Blaise5 survey.

The user interface of HaLo enables the survey and case management tools for interviewers. It works both online and offline state and synchronizes collected data and retrieves any updates between Mixeri and HaLo while interviewer has access to organization's network. The communication between HaLo application and Mixeri is handled by HaLo Server which is a service that passes messages in Field Data Collection channel.

3.1.4 Web Data Collection Service

Web Data Collection Service alias Channel consists of two different services which are WebHost and WebInstance. WebHost is a service which communicates betweet Mixeri and WebInstance and manages to which WebInstance commands are sent.

WebInstance completes the Blaise-integration in Web Data Collection channel by utilizing Blaise's API libraries. This integration includes the same capabilities as does HaLo in Field Data Collection channel. These are starting the survey interview, case data pre-filling and reading the collected data after the survey session has ended. It also handles the installation of Blaise-packages and launches the Blaise5 survey. All the necessary support services for identifying a case in login process is handled by WebInstance.

In Web Data Collection channel there is only one WebHost-installation that communicates with Mixeri. Channel may have more than one Blaise-installation. For each Blaise installation, there is also a WebInstance installation that communicates with WebHost. Currently there is only one Blaise-server-installation in use in Statistics Finland.

3.2 Interface services

Ruuti is utilizing other services through API for certain functionalities such as sample data input, collected survey data output and Login Portal for Web respondents. Also, Blaise survey packages are retrieved from external service. Some of these services such as Login Portal is used also in other Data Collection Systems in Statistics Finland.

3.2.1 Input & output

Samples for each individual data collection is uploaded to Ruuti System from statistical in-premise systems through API. One data collection may have many data collection periods and multiple samples. A

sample always contains the ontology variables required by the system such as basic case information. Some of these variables are also passed to Blaise5 for example to be displayed in interviewer's survey view. Also, other survey-specific pre-fill variables that contain data is passed to Blaise5.
Answer data can be read from Ruuti Interface any time to be used in Statistical Processes. All the actual data processing is handled outside of Ruuti System. For example, the sample and pre-fill data must always be created in Statistical Processes outside of Ruuti System because the system itself produces only raw data.

3.2.2 Login Portal

Statistics Finland uses Suomi.fi e-Identification which is a shared identification service for public administration e-services. The service is in use in the national and municipal e-services, in which a user must identify themselves reliably.

After a successful login either using e-Identification service or using in-house created credentials the respondent is directed through Login Portal to the Web Data Collection Service. The service then launches Blaise5 survey with encrypted primary key in URL-parameter. The surveys actual primary key is decrypted before it is passed to Blaise. This process disables the possibility to share or hijack the case. For example, if the visible and encrypted URL is copied and reused, the decryption process catches this and blocks the entrance to the survey.

3.2.3 Questionnaire Warehouse

Questionnaire Warehouse is a service that enables Blaise Developers to upload Blaise Survey Packages into use. These packages are always built with unique Blaise GUID and named with unique Ruuti-ID. Also, the data model is named with the same Ruuti-ID. This way no problems will occur in case of the need to break Blaise data model and update the survey package to an ongoing data collection. When the Blaise packages are uploaded to Questionnaire Warehouse Service, they are instantly available for Data Collection Manager in Mixeri's user interface.

In case of the the survey package is changed in the middle of the ongoing data collection, Mixeri handles the possible data conflict situations. For example, if a new installed Blaise Survey package does not include a variable which is collected with the previously installed package then Mixeri keeps the previously collected variable data but does not pass it to Blaise any longer. If the Blaise data model changes dramatically then the new package should not be included to a data collection before the next period even though it is possible.

4. System implementation into production

Overall goal of going towards to digital data collection has taken a long step in last two years in Statistics Finland. This means different measures in survey communication starting from contacting respondents and reliable identification to the response process itself with questionnaires and ending to feedback and rewarding. There are four main areas where the change management was and is essential. The whole data collection process has been reorganized, the new data collection system is in production phase, and multi-mode questionnaire development and testing process is taking a new shape as Agile methods are being

incorporated. Also, statistical methods are being developed further to answer the emerging analysis and quality issues.

In multi-mode administered surveys the demands of a flexible work division between interviewers is important and the tools for effective management and monitoring surveys have now been met with the Ruuti System. This development has been important at the transition from Blaise4-centered production to Blaise5-centered production. In our development process field interviewers and CATI interviewers use the same HaLo user interface in organizing their work. The feasibility of Blaise CATI management as part of Ruuti System is still being considered.

4.1 Roadmap of implementation

Ruuti System development started in August 2017. The first major steps were to rewrite a lot of the code written in the previous multi-mode system project and create a functional infrastructure and a functional communication between services. This meant largely the rewrite of Mixeri. The second step was to create HaLo Field Data Collection Service and its user interface. Third step included the creation of user interface and essential features for Mixeri.

In 2018 the first interview pilot survey was carried out and Ruuti System was tested. Later that year the work around Web Data Collection Service was started and it ended at the end of January 2019.

The implementation to production started in January 2019 but there were some compatibility issues with multi-mode surveys usage in the system. These problems were largely due to the inexperience integrating a demanding multi-mode Blaise Survey into the system.

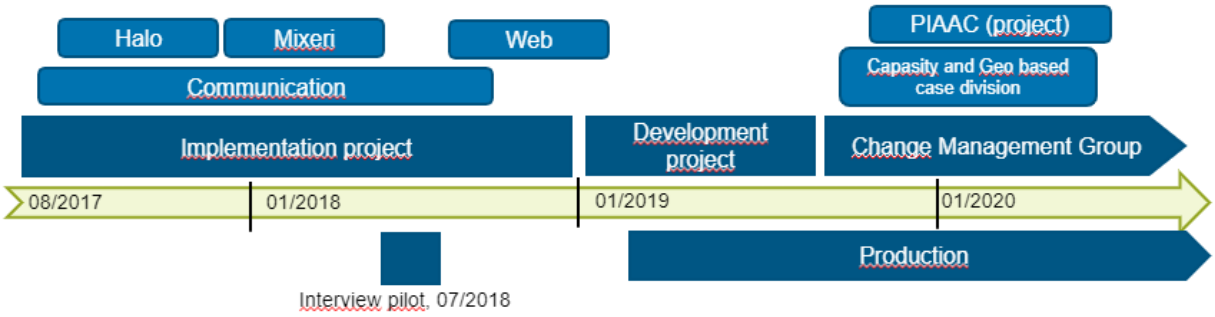


Figure 4: Ruuti System Development roadmap

4.2 Implementation schedule of multi-mode social surveys

Between the implementation years from 2017 to 2019, there have been three multi-mode pilot data collections carried out with Ruuti System. First pilot was carried out in June 2018 and its purpose was to test the HaLo Field Data Collection Service in action. The second pilot was carried out in November 2018 and in that pilot the technical capabilities for carrying out a multi-mode survey in Ruuti System was tested with Consumer survey.

Consumer survey together with Travel Survey were the first surveys to be implemented into production.

In 2019 another pilot was carried out when Labor Force Survey alias LFS was tested in Ruuti System. In this Pilot the formation of household with Blaise5 was tested the first time. In our roadmap for the coming

year there is a major pilot for Survey of Income and Living Conditions alias SILC. In this pilot the household formation will be tested further.

Production wise the Time Use Survey will be carried out in August 2020. SILC and LFS are scheduled to be carried out in 2021 and Household Budget Survey in 2022.

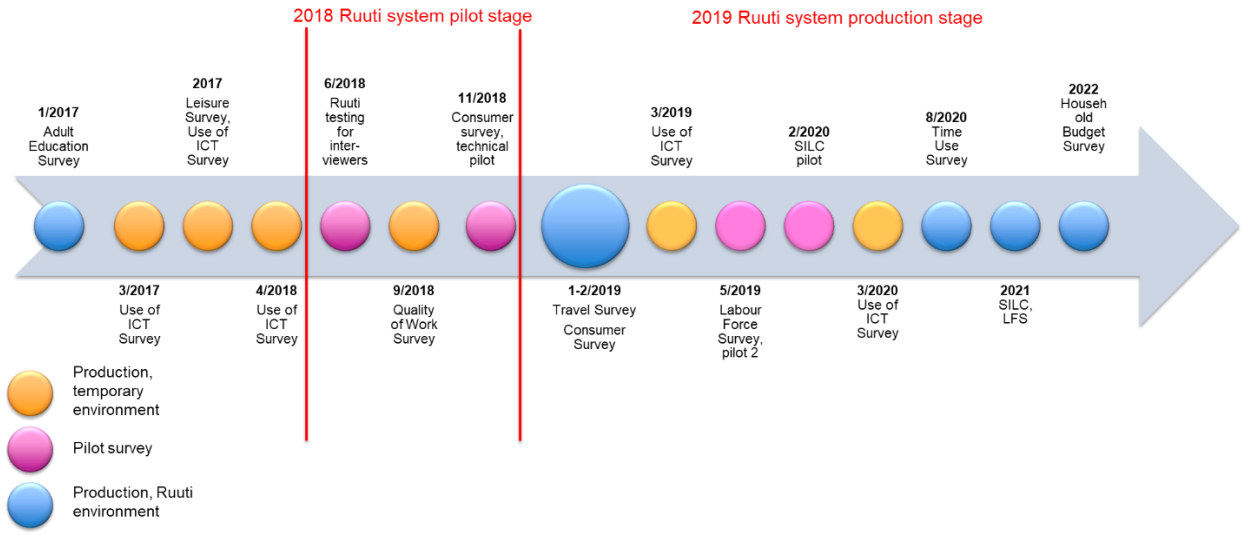


Figure 3: Implementation schedule of multi-mode social surveys 2017-2020

5. Conclusions and future plans

Development of the new data collection management was a much bigger effort than was considered at the start of the work. At same time moving from Blaise4 to Blaise5 created a lot of new requirements for the development process itself but also to the personnel of Statistics Finland with adaptation the new skills and processes. On business line of our data collection unit, the establishment of the dedicated product owner, was one big step to unify and intensify the more focused development process. The broader aims, reliability, usability, unity and flexibility to the Ruuti system are still valid. The time of evaluation and reflection will be relevant after we have moved all our production to Blaise5.

During system implementation some major flaws, bugs and technical issues of Ruuti system have been solved during 2019. In near future there are some new development areas in the roadmap of the system. Field Survey Management Service has got user feedback from interviewers. This will guide our development of more user-friendly user interface for interviewers. Other new important features of the system are the implementation of the Blaise5 CATI mainly because there is a special need to be able to use outside provider for interview services on the time of demand spike. And the possibility to establish and manage enterprise data collections is one perceived need.

We went to the implementation phase of system with MVP (minimum viable product) as defined by product owner. In future the work will continue to follow the principles of agile methods also in the questionnaire design. The new data collection system and Blaise5 together will guide the new generation of our statistical production.

Choréo – The Blaise 5 Multimode Management System

Mark M Pierzchala / Mark M Pierzchala, MMP Survey Services, LLC (MMPSS)

This is the status of the Blaise 5 Multimode Management System, now called **Choréo**[9] as of July 2020.

1. Abstract

The Blaise 5 Choréo will be demonstrated and discussed. The Choréo architecture closely follows the design agreed by the BCLUB Multimode Management Group as demonstrated at IBUC 2018. Choréo was first prototyped in Blaise 4 and an information webinar was given in August 2019. It was ported to Blaise 5 in November 2019 where work has continued.
While the overall architecture follows the BCLUB design, there were many details to work out. In addition, Choréo is designed to fit into an institute's existing infrastructure. For example, most institutes already have a Sample Management System (SMS). Such a system can be used to send mail and email, and to receipt responses when they arrive, among many other tasks. Choréo does not replicate such functionality.
Choréo does provide a Survey Handling System (SHS) with a Survey Handling Database (SHD). The Survey Handling Database keeps track of survey management statuses, counts, and indicators. It does not contain any Personally Identifiable Information (PII). Such confidential data are held in the institute's own Sample Management System.
The Survey Handling System issues instructions for all survey management actions. To do so, it must be able to communicate with an institute's systems. Further, Choréo should operate the way the institute wants. This is done through specification databases. There are databases for (1) survey design parameters, (2) happenings, and (3) actions. It also works with Blaise 5 CATI. The Choréo should be able to produce reports using an institute's own coding system. Further, there are hooks in the system to implement management responsive design.
How all this is elegantly done will be revealed at virtual IBUC 2020.

2. A Brief History of Choréo

The BCLUB Multimode Management Group was formed in October 2017. It was to define functionality for multimode survey management. The group members represented a wide variety of experiences and practices. The final report (Pierzchala, et al, 2018) was released to BCLUB members in September 2018. At the 2018 International Blaise Users Conference, a high-level description was given (Pierzchala 2018).

A webinar for BCLUB members in August 2019 showed a rough prototype programmed in Blaise 4. Lon Hofman of the Blaise Team at Statistics Netherlands ported the system to Blaise 5 in November 2019. A key feature that made this port possible was the extension of Manipula to handle dialogs. Finally, a status update for BCLUB members was posted on BaseCamp at the end of December 2019.

Recently, there was a first release of the CAPI Management App (CMA) as well as improvements to CATI. The progress of CATI, the CMA, and Choréo make it possible to integrate these 3 modules.

Recently, MMPSS reread the original BLCUB report, the 2005 multimode article by de Leeuw, and the 2016 AAPOR report on outcome dispositions. A new AAPOR report released in October 2019 (Olson, et

[9] Previous system names were MMS and M3. Choréo is a new faux word derived from 'choreography' since multimode management must choreograph many cases and outcomes. 'Choréo' is rooted in Greek, and is present in English, French, German, Dutch, and other languages. The 'é' is stressed. The pronunciation is 'core-ray-oh'.

al, 2019) provided an additional excellent review of multimode survey management needs resulting in new Choréo features or hooks for future features. Another valuable resource was a paper that proposed an outcome coding scheme for UK household surveys (Lynn et al, 2001).

2.1 The Near Future

The combined Choréo, CATI, and CMA sub-modules should be ready for testing in late 2020. Testing such a system, with thousands of possible sequences, requires automated testing running on a network.

3. The Goals of Choréo

Choréo has been designed to achieve the following goals:

- Provide a multimode management system in Blaise, that
- Fits easily into an institute's existing infrastructure, where
- The Choréo capability is value-added to the institute's own systems, that
- Is generalized and built around principles that are common to institutes, while
- Allowing an institute to maintain its own survey management coding scheme, that
- Is mainly implemented through an easy specification exercise for the institute, where
- The system is extensible for the institute by specification or modifying source code, which
- Applies to person- and household-level surveys, and at the same time,
- Collects massive case-level survey paradata that enable the institute to implement
    - Its own reports (in addition to some basic Choréo reports)
    - Its own Responsive or Adaptive Survey Design (Groves and Heeringa, 2006), Kreuter (et al, 2013), and Schouten et al (2018).

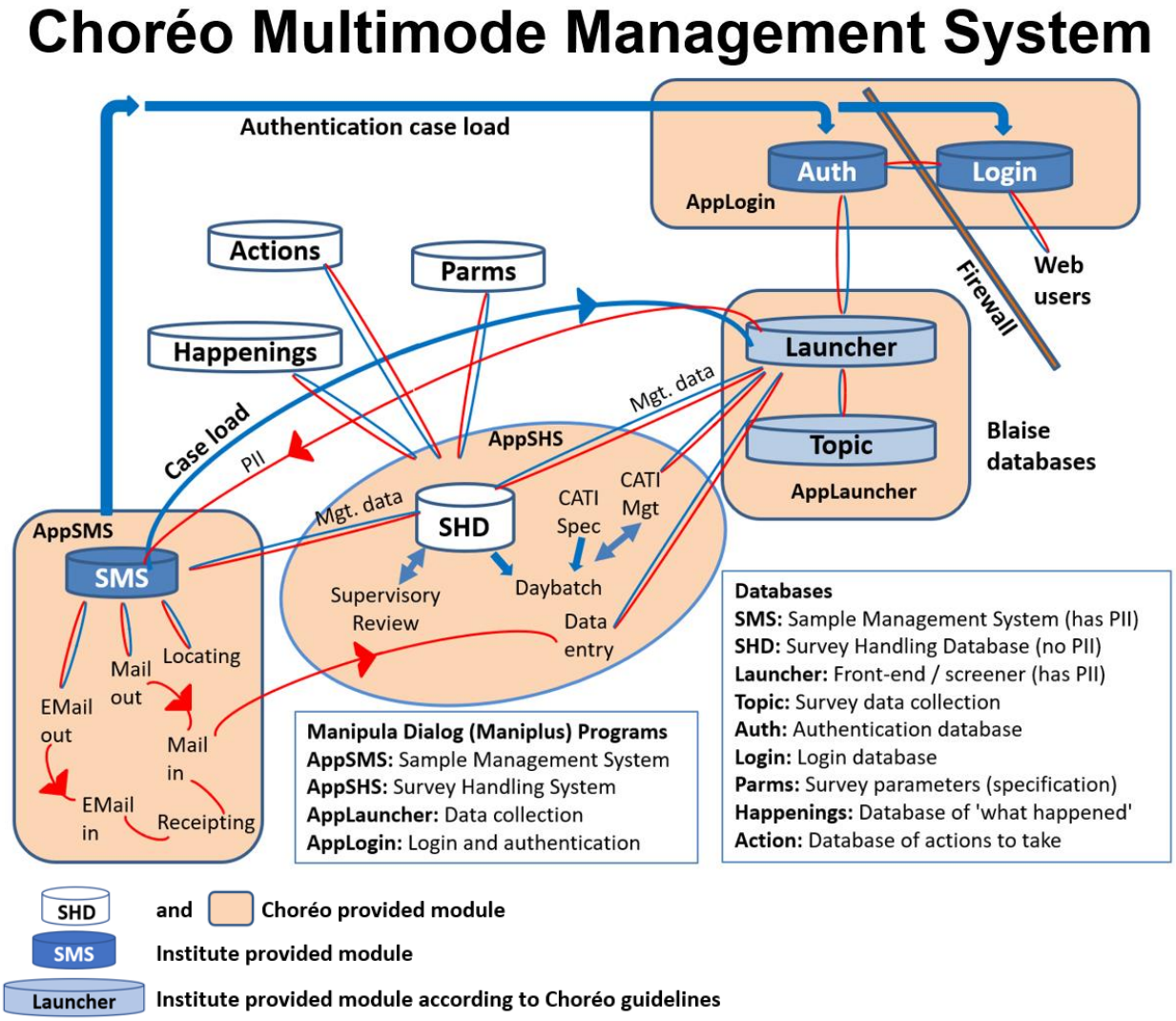
☛ Choréo is based on operational codes and statuses. In both AAPOR (2016) and Lynn et al (2001) these are called temporary codes. Choréo formalizes the handling of operational statuses. They contain enough information to map them to final disposition codes. See below for more information.

4. Choréo High-Level Design

The overall design and principles of Choréo have not changed since the 2018 BCLUB report. Figure 1 shows the overall architecture. Key design principles include:

- The Survey Handling Database (SHD), its external files Happenings, Actions, and Parms, and the 4 Manipula Dialog (Maniplus) App programs make up Choréo. The Launcher and Topic Blaise instruments would be provided by the institute, according to certain Choréo guidelines.
- The SMS, the web infrastructure, and many subsystems are provided by the institute.
- The internal Choréo source code will not refer to any published coding system.
    - Tokens (explained below) are used in the source code to refer to happenings and actions.
- Choréo is built around happenings and actions, and survey design parameters.

Figure 1: The High-Level Design of Choréo



The Survey Handling Database (SHD) is a Blaise provided database as are the external files used by AppSHS. The Launcher and Topic instruments may be Blaise datamodels, but this is not required.
The relationship between the Sample Management System (SMS), the Survey Handling Database (SHD), and the AppSHS is as follows:

- All Personally Identifiable Information (PII) is handled by the SMS.
- No PII is held in Choréo. None!
- Launcher holds selected PII that is necessary to identify the respondent or for an interviewer to conduct the interview.
    - The Launcher can collect or modify PII. This new information is sent directly to the SMS bypassing the Survey Handling Database (SHD) though the SHD knows about this.
- There is constant communication between the SMS, the SHD, and other modules via AppSHS.
- AppSHS commands Choréo including the SMS. When there is an independent SMS action (e.g., a paper questionnaire is receipted), it reports the action to the SHD.

5. Building Choréo around Happenings, Actions, and Survey Parameters

A happening is simply anything that can happen to a case during a survey. A happening occurred in the past. Examples include:

- A busy signal
- The respondent slammed a door in the face of an interviewer
- A web respondent logged in
- An email bounced back

An action is anything that can be done with a case. An action would occur in the future. Examples include possible actions to take after the happenings above:

- Call back the respondent in a few minutes
- Record an adamant refusal
- Rest the case for all outbound contact attempts
- Change to a different email address if available

The Parms database is a survey design database. It is philosophically like the CATI specification module, but its scope is for the entire survey. At some point, either (1) a clear line will be drawn between the CATI specification module and Parms, or (2) they will be combined. For the Actions above, the Parms survey specification or the CATI specification file would be used as follows:

- CATI specification: Follow the callback pattern for busy signals
- Parms: Stop the case
- Parms: Indicates the length of the rest period
- Parms: Switches the email pointer to a different email, if one is available

5.1 Why Happenings and Actions

Possible happenings and actions are common to all institutes. A busy signal may sound different from country to country, but a busy signal is a busy signal. Happenings and actions are operational in nature. Choréo provides a standard and extensible happenings database of around 400 records and an actions database of more than 200 hundred records.

The use of happenings and actions avoids using any published outcome coding system during survey operations. However, the AAPOR (2016) outcome coding system was heavily referenced mainly to determine which attributes of a happening or action would be useful for survey operations.

Records in the happenings and actions databases are identified by tokens (primary key). These are character strings that are used (hopefully minimally) in the Choréo source code. For each record (identified by a token) the institute is free to:

- Give any numeric or alpha-numeric code to the happening or action record
- Give any description to the happening or action record
- Change specific attributes of a happening
- Add a happening or action record that has a unique token

The code and label can be based on the AAPOR or AAPOR-like coding system, or it can be based on a long-standing institute coding system. Choréo does not care.

If a new happening or action record is added, the institute may have to modify one or more Manipula procedures that indicate how to handle the new record (see below).

5.1 Generalization of Choréo with Tokens

Figure 2 shows a few records from the Happenings database specification.

Figure 2: Happenings Specification with Token

Happenings Token	InstCode	MetaDescription
NonC_Busy_Phone	11336	Busy
NonC_SomeOneElse	90337	Contact with someone else
NonC_VMNoIDNoMsg_Phone	11401	Voice mail, SM no ID, no left message

The token *NonC_Busy_Phone* contains 3 pieces of information separated by the underscore: *NonC* indicates this happening is a non-contact, *Busy* (a busy signal) is the happening, and *Phone* is the channel of communication. *InstCode* is an institute-provided alpha-numeric code. This entry enables the institute to connect with its own systems and coding schemes. *MetaDescription* is also provided by the institute. The second entry token, *NonC_SomeOneElse*, applies to all channels of communication or modes because it does not have a third component.

The main point of the happening, e.g., *Busy*, is that it can happen to any survey that uses the phone channel of communication. This allows Choréo to adapt to any institute's survey program without imposing an outcome coding scheme.

Figure 3 shows a tokenized action. The token name uses the same concepts as the Happenings tokens. *ActionCode* and *Action Descriptions* are institute specific while the token is used by Choréo.

Figure 3: Action Specification with Token

Actions Token	ActionCode	Action Descriptions
Follow_Busy_CATI	260	Follow the CATI busy treatment

5.2 Careful Use of Terms

Two of the happenings in Figure 2 use the term *Phone* while the token shown in Figure 3 uses the term *CATI*. This is a deliberate usage of these terms. *Phone* refers to the general use of the phone as a channel of communication while *CATI* refers to the use of the Blaise CATI Management System.

5.3 Attributes of Happenings Tokens

The specification of happenings includes many attributes for each happening. Figures 4, 5, and 6 show additional attributes of happenings specifications.

Figure 4: Happenings Tokens with Attributes (basic information)

Happenings Token	WhoActed	Burden	Cost
NonC_Busy_Phone	NoOne	3	2
NonC_SomeOneElse	NoOne	3	1
NonC_VMNoIDNoMsg_Phone	Interviewer	5	3

Figure 5: Happenings Tokens with Attributes (implications of the happening)

Happenings Token	BurdenReset	Direction	Contact	Identity	Eligibility	ChannelValidity	SPImplication	SampleCategory
NonC_Busy_Phone	No	Outbound	Unknown	No	No	NoInfo	NonContact	UnknownEligibility
NonC_SomeOneElse	No	Outbound	Unknown	No	No	Invalid	NonContact	UnknownEligibility
NonC_VMNoIDNoMsg_Phone	No	Outbound	No	No	No	UnkValidity	NonContact	UnknownEligibility

Figure 6: Happenings Tokens with Attributes (linked actions, the first of several each, there are more →)

Happenings Token	DLA1Token	DLA1Text	Pr1
NonC_Busy_Phone	Follow_Busy_CATI	Follow the CATI busy treatment	Required
NonC_SomeOneElse	SendSupReview	Send case to supervisory review	Required
NonC_VMNoIDNoMsg_Phone	Rest_CATI	Rest case for outbound CATI	Required

The presence of attributes makes the internal Choréo source code easily adaptable to the institute. To the extent possible, Choréo is programmed in terms of attributes.

5.4 Choréo Programming and Adaptability Strategy

Choréo is programmed in Manipula and refers to Blaise databases. There are 4 master programs that start with the acronym App. These are **AppSHS**, AppSMS, AppLogin, and **AppLauncher**. AppSHS and AppLauncher are part of the distribution. AppSMS and AppLogin stand in for the institute's own systems. For now, they allow thorough testing. Each Manipula program uses many small procedures.
Internally applied procedures: These are procedures that impact the inner workings of Choréo. For example, if only AppLauncher and AppSHS are involved, this is totally within Choréo.
Externally applied procedures: These procedures connect Choréo to the institute's own IT infrastructure. For example, if AppSHS must send something to the institute's SMS, then the institute will want to adapt this procedure to govern this interaction. This might be done by converting some Manipula assignment statements to Manipula API statements.
For each happening, every effort is made to program Choréo solely in terms of happenings attributes. While this is the overall goal, there are times when the Choréo source code refers to tokens.
Choréo **programming in terms of attributes:** When this is achieved, this part of the Choréo system is completely parameterized. This means you can change the behaviour of the system by changing the attributes of the happenings record, by adding additional happenings or actions records, or modifying the contents of the Survey Parameters Database.
Choréo **programming in terms of tokens:** When this is necessary, the institute may have to modify one or more of the Manipula procedures. All Choréo Manipula source code is distributed.

5.5 Extensibility of Happenings and Actions

Choréo provides hundreds of specified happenings and actions that apply to almost any institute. However, the institute can add its own happenings and actions tokens along with their attributes. If Choréo can operate on their attributes, then there is no further work for the institute (in theory). On the other hand, if direct reference to tokens is required, then the institute may have to modify source code.

5.6 Successions of Happenings and Actions within an Attempt

For an attempt, there will usually be successions of happenings and actions. Let us use the example of a phone attempt.
Happenings (past tense) include:

- Case was delivered to the interviewer
- Interviewer reviewed the case
- Interviewer placed a call
- A busy signal was recorded (happening NonC_Busy_Phone)

Happenings are duly recorded in the Survey Handling Database (SHD) because they actually occurred. Actions (future tense) for the happening NonC_Busy_Phone are recorded as attributes of the happening record (with identifier NonC_Busy_Phone). Actions to execute will or may include:

- Refer to the CATI management system busy treatment specification (required)
- Call back in a few minutes (if indicated by the CATI Management system)
- Retire the case for the day (if the action above is not applicable)
- If there is a succession of busy signals, refer the case to supervisory review (if the action above is not applicable and if indicated by survey parameters)
- Switch to a new phone number if available (if the action above is not applicable).

The sequence of actions includes required, possible, and/or contingent actions. The requirement status of each action is recorded in the happening record. See Figure 6. Note that Figure 6 shows only 1 of several actions that are possible for each happening.

5.7 Implied Operational Statuses – Derived from a Succession of Attempts

Some cases will have lengthy and complicated attempt histories. Choréo anticipates these situations and provides adaptable procedures to assess the true situation. For example, a succession of No Answer happenings may indicate the respondent is travelling and the case can be put on rest for a while.

6. Survey Parameters

The survey parameters file, *Parms*, is where you design your survey. Figures 7, 9, and 10 give an idea of the many possible survey parameters.

Figure 7: Page 1 of the Parms Specification

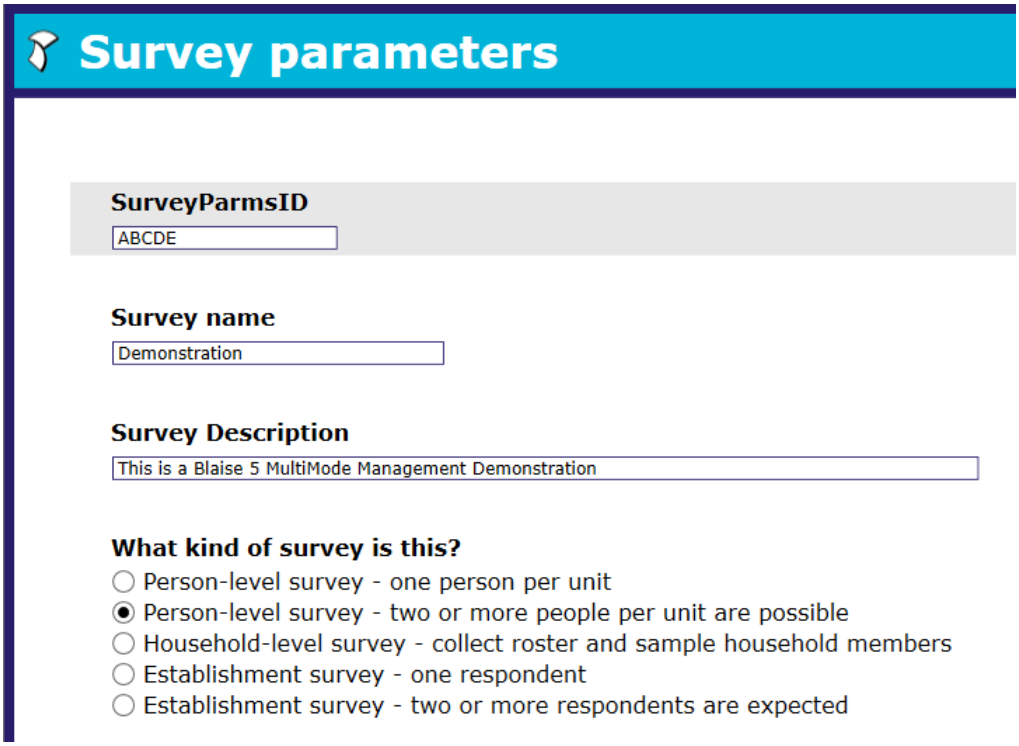


Figure 7 shows that Choréo is designed to handle person-level and two-stage surveys. This is enabled by a two-part ID number. Figure 8 shows a UnitID and a CaseID. Unit 10002 holds 2 cases.

Figure 8: Two-Part ID Structure

UnitID	CaseID
10001	1
10002	1
10002	2

Figure 8 shows the situation, known ahead of time, where 2 people from the same household are selected for a survey. This can happen in a survey of program participants where 2 or more adults from the same household participated in the program. This is operationally important to know. For example, if you interview person 1, then after this first interview you can ask if person 2 is available. If so, then person 1 can hand over the phone to person 2 and it is possible to get the second interview immediately. Alternatively, where you access a household, enumerate the members, then determine which are to be interviewed, Choréo can easily handle the spawning of cases. In a household, each would have the same UnitID but a different CaseID and its own SHD record. This latter scenario also applies to a corporation with subsidiaries (complex corporations can be represented using a cleverly constructed 2-part ID). Longitudinal surveys can be handled similarly. Unit-level reports can be generated by UnitID.
Figures 9 and 10 show that Choréo distinguishes between data collection modes and channels of communication. This distinction is due to de Leeuw (2005). For example, 2 modes shown in Figure 9, paper and CAPI, rely on a physical address that underlies the mail and visit channels of communication (figure 10). Choréo knows if a happening applies to a mode or to a channel. Thus, for example, the paper mode is distinguished from the mail channel.

Figure 9: Modes Specification

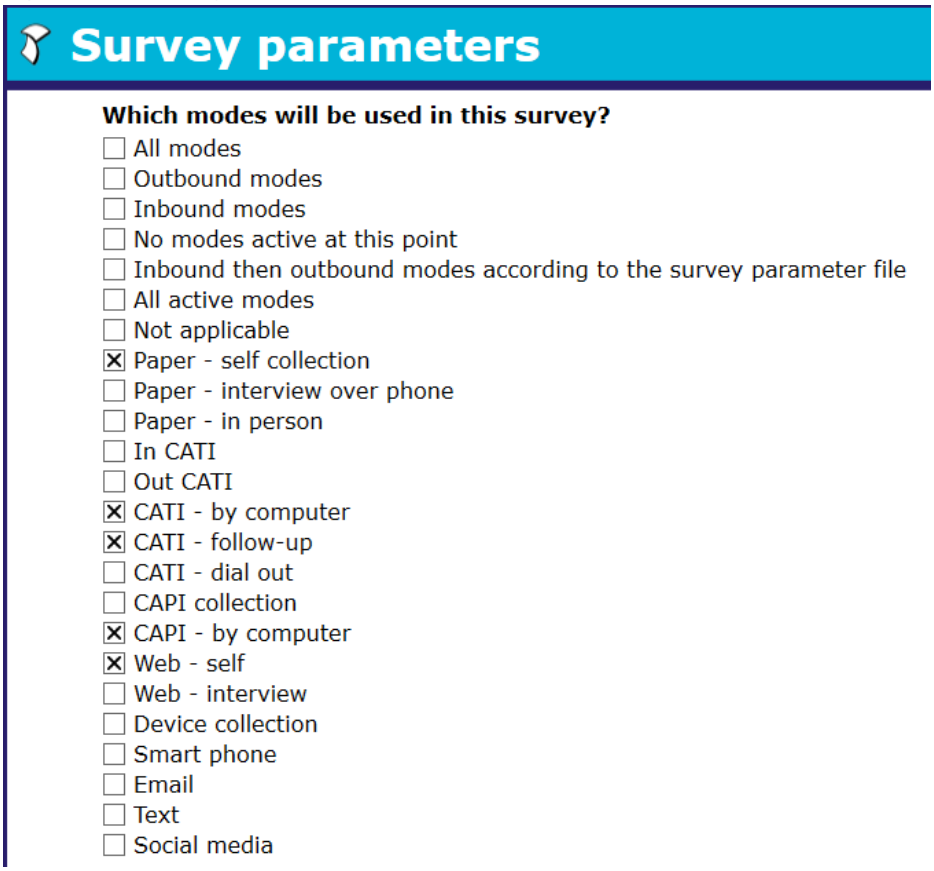
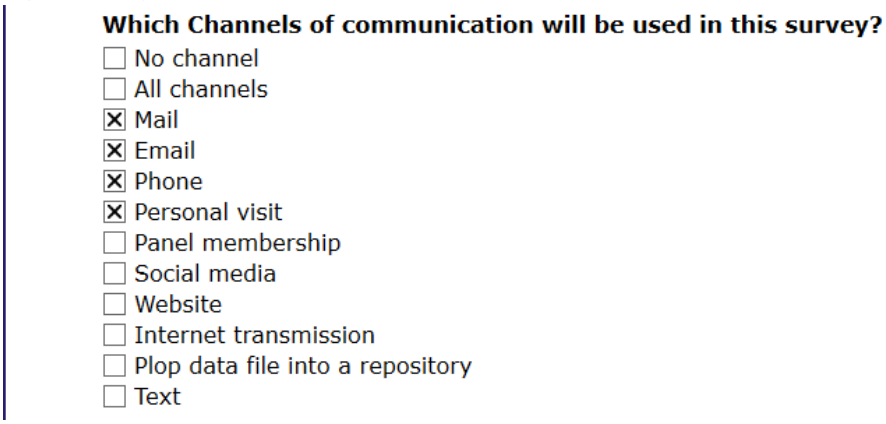


Figure 10: Specification of Channels of Communication



Survey parameters allow many design options including whether the survey is staged (e.g., one mode follows another) or is racehorse (all modes all the time), the length of kinds of rest periods, how many attempts to allow per case, and so on. Operationally, Choréo records happenings, refers to indicated actions, considers sequences of attempts, refers to Parms to fill in details, then acts through a procedure called *HazLo*, Spanish for do it!

7. Operational Codes versus Final Disposition Codes

Operational codes and final disposition codes are highly related, but they are not the same. An operational code, or a sequence of operational codes for a case, must map to a final disposition code as the survey closes. Operational codes are useful when they help survey-conducting staff optimize effort. According to Lynn et al (2001) the mapping to final disposition codes should be objective and as automated as possible. However, there are times when the final code is assigned by survey staff through a manual process. These tend to be cases where there is a complicated history. In Choréo, this history is stored in the SHD. Table 1 below gives some high-level operational status scheme implemented in Choréo.

Table 1: High-Level Operational Status Scheme

Working:	**Appointment:**	**Rest case:**
Engaged	RDefAppoint	RestToRefield
Active	ODefAppoint	ProcessToRefield
NoChange	RAmbAppoint	PendLocating
	OAmbAppoint	
Refusal:		**Supervisory review:**
RSoftRefusal	**Breakoff – Resistance:**	SupReview
RHardRefusal	BreakoffIntro	
OSoftRefusal	BreakoffAfterIntro	**Stop:**
OHardRefusal	BreakoffOther	StopOutbound
AdamantRefusal		StopAll

7.1 Statuses, Counts, and Indicators

Choréo keeps track of several kinds of statuses, counts, and indicators. These values are stored in the SHD. To map to final dispositions, each case tracks (1) whether R is ever identified, (2) whether R is ever contacted, (3) whether R has ever been engaged, (4) whether R is eligible, and (5) completion status. For each happening, it is possible to assign a relative burden value and a cost value. Both use scales from 0 to 100. The concept of burden is rarely if ever defined well. Perhaps it is easier for an institute to say that a Busy signal is no burden to R, a left message has relative burden 5, making an appointment has

relative burden 10, and completing an interview has relative burden 30. These are accumulated. It is possible to reset the overall burden for some happenings. For example, if you have been pursuing a phone number only to find out that it is not connected to R, then Choréo can reset R's burden to zero.
Similarly, relative costs are used. Placing a phone call may have relative cost of 2 while locating a new phone number may have a relative cost of 15. This gets away from the problems of knowing exact cost per happening and tying the system to a currency.
It is possible in the Parms survey specification database to establish burden, and cost limits. When these are reached, the case is stopped or sent to supervisory review.

8. Current State of Affairs for Choréo

Work over the past few years resolved many difficult conceptual issues and details. Discussions after the 2019 webinar clarified how to make Choréo generalizable using tokens and other measures. The emergence of the CMA as well as further development of Choréo and CATI now enable full-scale prototype testing.
Testing of Choréo will be on a network of computers using automated scripts that can randomly execute sequences of attempts for a case. The Survey Handling Database (SHD) holds a wealth of survey management information. Listings and reports will be generated that will allow an analysis of the sequences of events and whether they were handled correctly. An interesting test will be to see how well Choréo maps to the AAPOR (2016) coding scheme and the scheme proposed by Lynn et al (2001).

9. References

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

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

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

Kreuter, Frauke (editor). Improving Surveys with Paradata – Analytic uses of Paradata, © 2013, John Wiley and Sons.

Lynn, P., Beerten, R., Laiho, J., and Martin, J (2001). Recommended Standard Final Outcome Categories and Standard Definitions of Response Rates for Social Surveys, Working Papers of the Institute for Social and Economic Research (ISER), paper 2001-23. Colchester: University of Essex. https://www.iser.essex.ac.uk/research/publications/working-papers/iser/2001-23.pdf

Olson, K., (Chair), Smyth, J. D. (Co-Chair), Horwitz, R., Lesser, V., Marken, S., Mathiowetz, N., McCarthy, J., O'Brien, E., Opsomer, J., Steiger, D., Sterrett, D., Su, J., Suzer-Gurtekin, Z. T., Turakhia, C.,, Wagner, J. (2019). Report of the AAPOR Task Force on Transitions from Telephone Surveys to Self-Administered and Mixed-Mode Surveys, found at: https://www.aapor.org/getattachment/Education-Resources/Reports/Report-of-the-Task-Force-on-Transitions-from-Telephone-Surveys-FULL-REPORT-FINAL.pdf.aspx.

Pierzchala, Mark M., Hart, M., Suresh, R., Rodriguez, G., Filippenko, L., Sjodin, S., Boodhumeah, N., Miceli, C., Degerdal, H., Båshus, T., Haslund, J., Maher, P., Frey, R., Bilhorn-Janssens, A. Abernathy, T., Hofman, L., and Carati, T. (2018). Blaise 5 Multimode Management – A Report by the BCLUB Multimode Group, available to BCLUB members on BaseCamp.

Pierzchala, Mark M. (2018b). Blaise 5 Multimode Management – A Report by the BCLUB Multimode Management Group. Paper presented at the 2018 International Blaise Users Conference, Baltimore, U.S., October 2018. This is a condensed version of BCLUB report. http://blaiseusers.org/2018/papers/4_1.pdf
Schouten, B., Peytchev, A., and Wagner, J. Adaptive Survey Design, © 2017 Chapman & Hall.

Blaise 5 in SHARE

Maurice Martens, CentERdata

1. Abstract

The Survey of Health, Ageing and Retirement in Europe (SHARE) is a multidisciplinary and cross-national panel database of micro data on health, socio-economic status and social and family networks of about 140,000 individuals aged 50 or older (around 380,000 interviews). SHARE covers 27 European countries and Israel in 40 locales and is currently fielding its ninth wave. The questionnaire is fielded in CAPI mode.

CentERdata scripts the questionnaire in Blaise and has developed several tools that connect to these questionnaires. For latest wave the questionnaire was migrated from Blaise version 4.8 to version 5.4.

This migration was not only a change within the Blaise environment but also forced an adaptation of all tools we developed that interfaced with Blaise in our architecture. This step meant a new case management system was developed, new interfaces with our Translation Management Tool (TMT) and a new design for the Data Delivery backend were created.

This paper discusses these solutions and will also address some positive and some negative findings. The Blaise 5 development environment, full Unicode compatibility, the introduction of roles were greatly appreciated. The limited functionality for triggering events, keyboard navigation and control over lookup tables is something that should be improved on. We will discuss the workarounds we implemented to get these features in.

2. Introduction

The Survey of Health, Aging and Retirement in Europe (SHARE)) is a multidisciplinary and cross-national panel database of micro data on health, socio-economic status and social and family networks of about 140,000 individuals aged 50 or older (around 380,000 interviews). SHARE covers 27 European countries and Israel in 40 locales and is currently preparing its ninth wave. The questionnaire is fielded in CAPI mode and is fully ex-ante harmonized, this means that the questionnaire is centrally programmed and its underlying data model is identical for all countries. This allows for a quick turnaround and easy access to the data during fieldwork.

CentERdata is an independent, non-profit research institute located on the campus of Tilburg University (NL). The institute is specialized in (online) data collection, data dissemination, survey methodological research, tailor-made software solutions, model building, and socio-economic (policy) research. CentERdata hosts a representative household panel, the Longitudinal Internet Studies for the Social Sciences (LISS) Panel, with excellent possibilities for representative surveys and (controlled) experiments. In addition, CentERdata has expertise in state-of-the art econometric and data science models and techniques and collaborates closely with (academic) researchers on national and international projects. CentERdata also offers support in the field of valorization and project management, for example to support companies to arrive at data-driven and evidence-based decision making.

CentERdata scripts the SHARE questionnaire in Blaise and has developed several tools that connect to these questionnaires. For the eight wave of SHARE, the questionnaire was migrated from Blaise version 4.8 to version 5.4. This migration forced us to adapt many of the tools we developed over the years that interfaced with Blaise in our architecture. This paper will discuss the software architecture as used in the

SHARE study in wave eight. The SHARE study is ex-ante-harmonized. We develop a single source questionnaire and define a data model that is used in all participating countries. Only the translations differ per language. An overview of the SHARE dataflow depicted in the figure below.

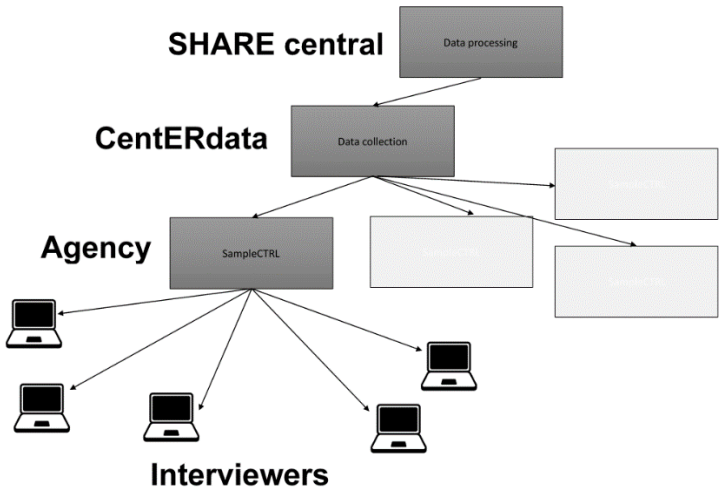


Figure 5: Architecture

We send out interviewers with laptops to households. There are two types of households: panel households; they have done at least one interview with a person in the household in a previous wave, or refreshers; new households that participate for the first time. On the laptop a Blaise questionnaire is installed, together with CaseCTRL, a program that can be used by the interviewer to manage the local subsample. It registers contacts and contact attempts, it has a build in agenda, and runs a household grid. When the interviewer has a connection to the internet, they can synchronize their data from CaseCTRL with an installation of SampleCTRL. Sample control is an application that runs at a server at a local agency that is hired to coordinate the fieldwork in a country. This agency can monitor the individual interviewers, can assign and (re-) distribute subsamples. They are requested to synchronise with a central server every two weeks. In this synchronization step the data is anonymized and send over to a Data Collection server. This server is hosted by CentERdata. In this Data collection environment, the data is joined together and some basic validation is done on the data. When the data passes the validation it is pushed to the SHARE central server in Munich, Germany where during fieldwork the data is processed into fieldwork progress reports and ultimately disseminated, cleaned and published.

The SHARE development workflow starts with a document that describes the questionnaire definition for the new wave. CentERdata will review this and builds a first version of the questionnaire in Blaise, this is tested and if needed adaptations in questionnaire definition or in the programmed source tool are done. Several of these cycles are done until everybody is happy with the programmed source questionnaire, and we freeze the development of the source questionnaire. Now the questionnaire is uploaded to the Translation Management Tool (TMT). The TMT is an online tool that coordinates the translation process. It is can be configured to support various translation processes, including the TRAP-D process. Up to wave 7 of SHARE, the TMT used the Blaise API to load in a compiled Blaise source questionnaire, detected changes in the source version from an earlier version it already had stored, and flagged those in the translation environment. The translatable elements are shown in context, together with texts used in the other tools, like the Sample Management System and are translated by professional translators. We will import the translated text in a Blaise questionnaire and several iterations start, where a version is tested, translations are adapted and new tools are generated. During the country specific testing phase, it can happen we find country or language specific requirements that force an adaptation in the source

questionnaire. If the issue is deemed problematic enough this might force an adaptation to the original definition, which restarts this loop from start, for the involved items. Ultimately tools are generated that can be used to do fieldwork with. Share has three such development cycles per wave. One for a pretest, where new items are tested, next for the field rehearsal, which aims at generating a final instrument to be used in the main fieldwork phase, and last the version for the main fieldwork phase, in general only bug fixes and possibly some questions removals.

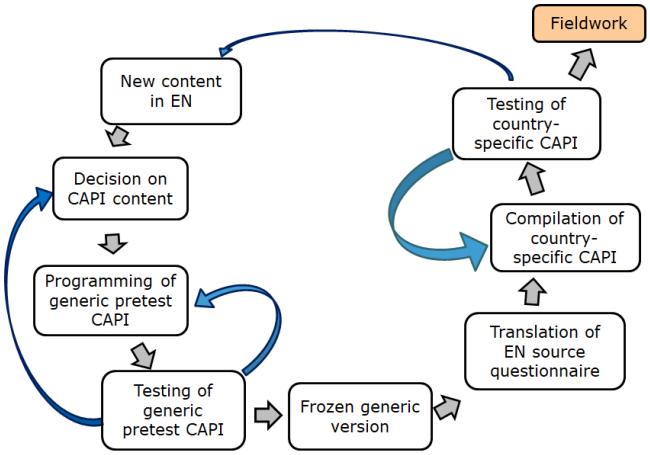


Figure 6: Questionnaire development process

We intended to keep this development workflow for the wave eight instrument, but hoped to reduce the number of cycles due to bugs. Especially on versions, we generated for languages with Non-Latin scripts, often issues were reported with non-readable text, especially when fills are used.

The SHARE tools up to wave 7 used the TMT to load in texts into the CAPI and the SMS. The CAPI is in this context the Blaise questionnaire which is called via the DEP. It is started by the Sample Management System (SMS), this talks to a tool called Sample Distributer (SD), which transfers to the Data Delivery System (DDS). Finally, the data is ported to the Data Portals, which has three views on it: Internal, External and Questasy (see figure 3).

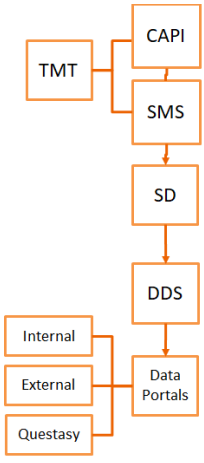


Figure 7: Previous waves tools

3. Migration

Many of these tools used the Blaise API or interfaced with the questionnaire's database, and might have to be adapted. We first draw up a migration plan to set up SHARE Blaise 5 CAPI, so we could test out how feasible a migration would be:

1. Import wave6/wave7 questionnaire; several items from the wave 6 and wave 7 questionnaires reoccurred in the wave 8 questionnaire. We imported them in Blaise 5.
2. Update to wave8 questionnaire; in the Blaise 5 environment, we adapted the imported questionnaire to the wave 8 definition.
3. Design default interface; we understood the questionnaire layout as used under previous waves, with the split screen, was no longer available, we had to build an interface that still supported several 'old' features, like keyboard navigation.
4. Implement non default features; in SHARE there are some questions that need counters, show wordlists or need more complex lookup tables, in phase 4 these were addressed
5. Export to TMT; the export to the TMT needed to be redefined, we decided to review this once we knew which of the features would be possible or when it would be clear if there were new concepts in the questionnaire.
6. Import from TMT; in earlier waves we copy pasted the translated texts in the source code; ideally there would be better processes we could use this time to generate translated questionnaires.

The import of the previous wave's questionnaire worked without any problems. And scripting the questionnaire worked perfectly. The new environment was useful. It sped up development time, the background parsing, helped finding problems immediately. Since the interviewers were used to using the split screen, which had the navigation paths displayed on the bottom of the screen, we tried our best to mimic this in the Blaise 5 environment ourselves. This failed; we tried several ideas but concluded it was not feasible. Another feature the interviewers liked were the icons that showed when a 'DontKnow', 'Refusal' or 'Remark' was attached to a question. We added these icons to a field pane, and determined their visibility on the status of the field.

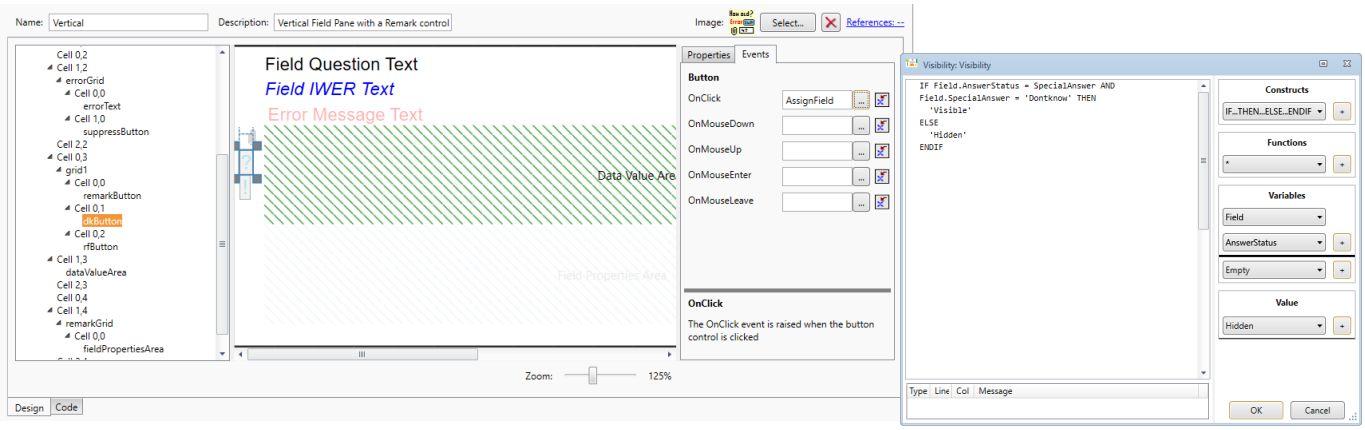


Figure 8: DK and RF

Several non-default features needed to be explored:

- Jobcoder; SHARE invested a lot of effort in setting up large databases of already classified job titles; in previous waves we wrote external apps to show these, so we could have full control over the

algorithms and behavior. We hoped in Blaise 5 the way lookups behaved would be improved, to allow for alternative algorithms, and we hoped to show a list that also accepts whatever is typed in, so we could in go detect that there was no match, and still collect the response. This was unfortunately not possible, we ended up using the lookup as they are available in Blaise 5 with trigram search, but still hope this will be further improved.

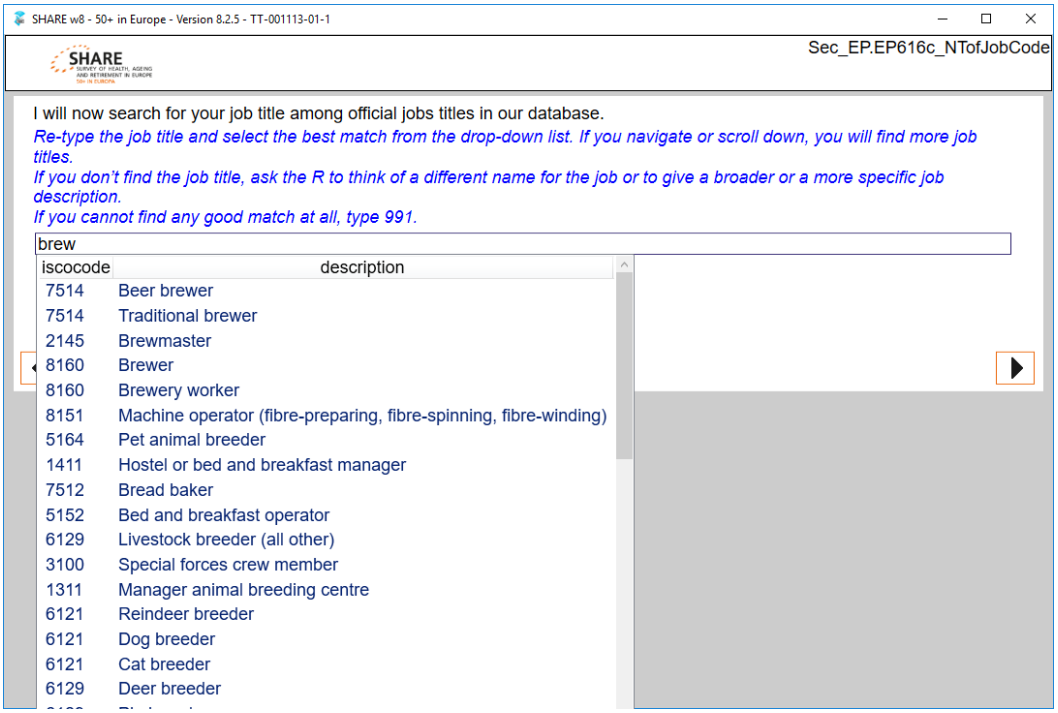


Figure 9: Lookup table

- Wordlists/Counter; In SHARE, there are some cognitive measures, like a timed word recall, questions where a timer and a stopwatch are shown. In previous waves videos were used to display these. Unfortunately, at the time we developed the wave eight questionnaire videos were not yet supported in Blaise 5. We invested quite some time in using the timer, somehow feeding an array of texts to the timer that would then change the text of a label at each tick. This however did not work, changing labels at runtime via an action is not possible, and this also makes it impossible to develop a feature like a stopwatch. We ended up using animated GIF files, and disable navigation, this solution was not ideal since the GIF restart when enter is pressed. If only the Text property of labels could be changed by clicking a button or by a tick from the timer, we would have developed this differently.

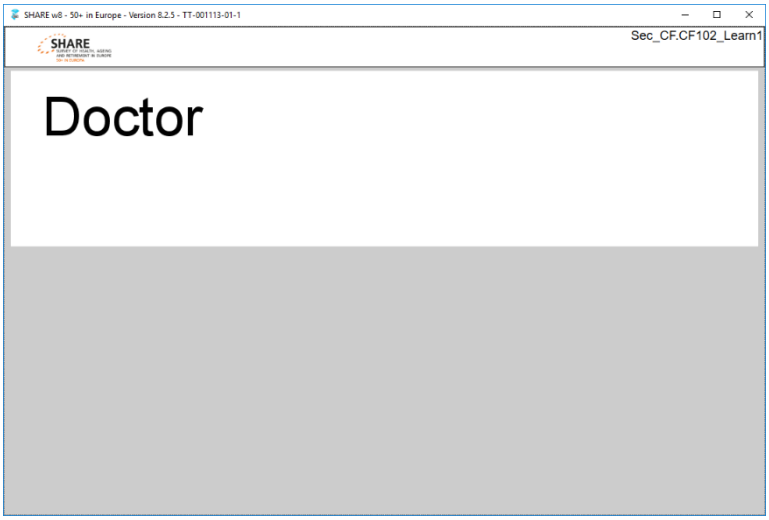


Figure 10: Disabled navigation while word recall list is playing

- Keyboard navigation; a key functionality, the complete questionnaire is in CAPI mode, the interviewers work quicker if they do not need to point and click. In version 5.4, this was not yet implemented. To get this working we introduced enumerationTextBoxes and setTextBoxes, which got the focus when a field pane is activated, one could type in the responses and the attached checkboxes or radio buttons would be checked. In addition, when the interviewer checks the radio buttons or checkboxes, the textboxes will show the value, like in earlier waves.

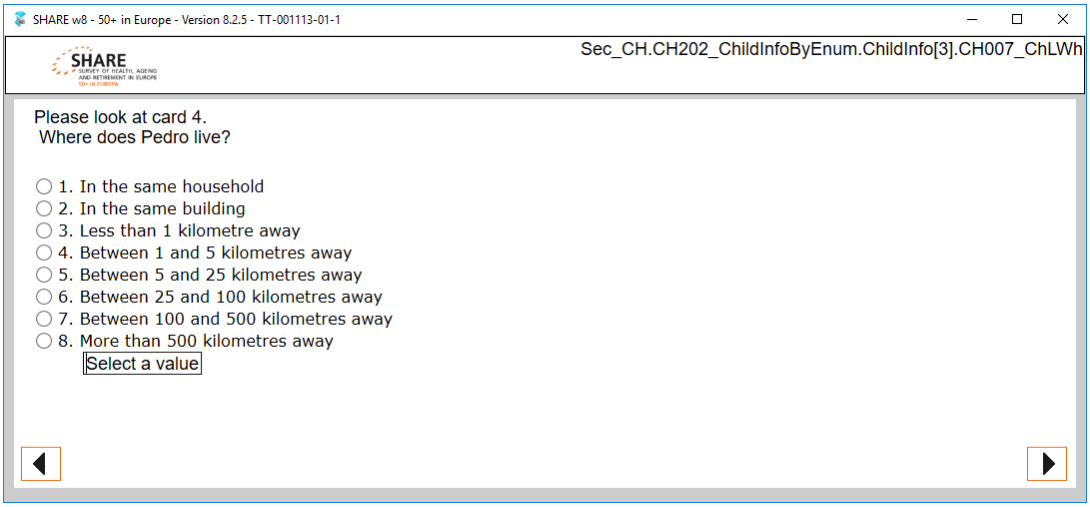


Figure 11: When nothing is entered, the setTextBox shows the message ‘Select a value’

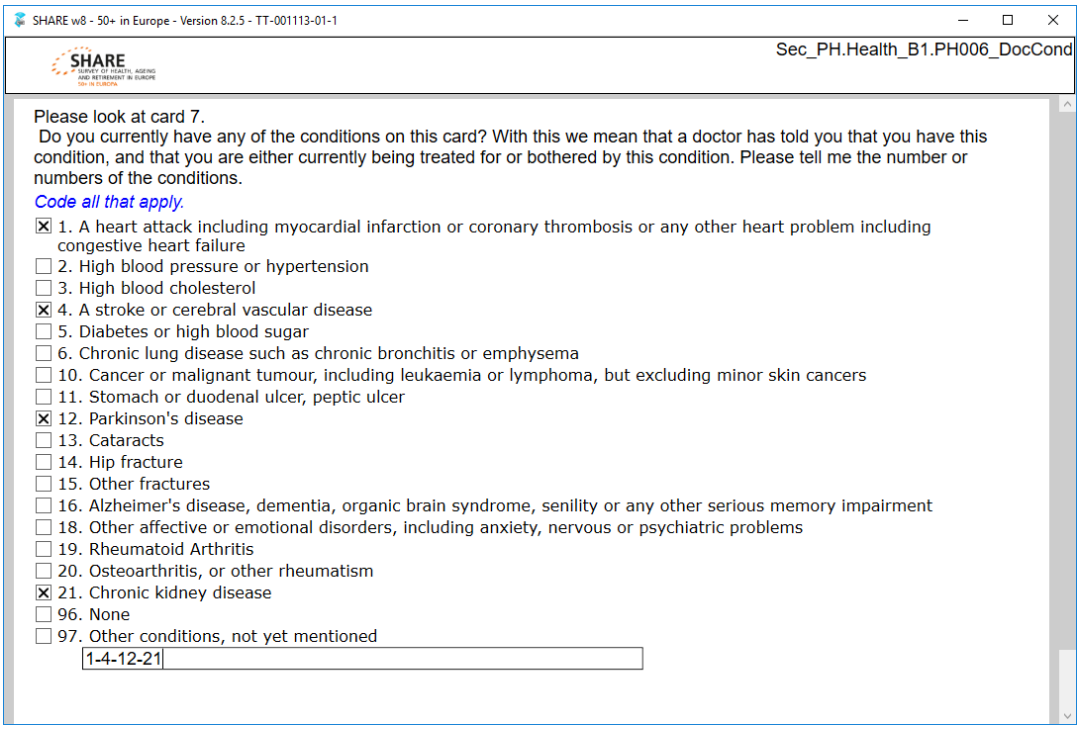


Figure 12: Keyboard navigation to select multiple response options

- Textroles; We like the text roles features very much, in earlier waves of SHARE this was already done implicitly, interviewer instructions and questions text were already defined as separate translatable items and during import we used layout we gave them different fonts and colors,.Now we can easily set this up.

- Unicode; Under Blaise 4 CentERdata developed several hacks to support the SHARE questionnaire in non-Western scripts like Arab, Hebrew or Russian. These hacks cost us frustration, and needed thorough checking, especially for fills that were cut off at 256 characters without any error message when fed through procedures. We really appreciate the full support of Unicode in Blaise 5.

- Images; Wave 8 of SHARE had some questions that used images on show cards, since the use of images was so straightforward, we also show them in the responses the interviewer sees.

- Child grid; since the split screen as common in Blaise CAPI questionnaires in Blaise 4, is not implemented in Blaise 5, it was not convenient to present a full overview of all children in a household, or at least not one compatible with the previous questionnaire routing and field. We chose to rephrase the questions, and develop a child overview on screen as a separate area in the field pane.

- We match children that were preloaded from a previous wave or from the responses of the partner, and children we may have detected in the Social Network section, and possibly add children that are not already mentioned. In this interface, the list of children is shown on the right, and will change to green and will add a check mark when a child is confirmed, if one of the children is mentioned twice or should not be in the list, they are colored red and a cross is added.

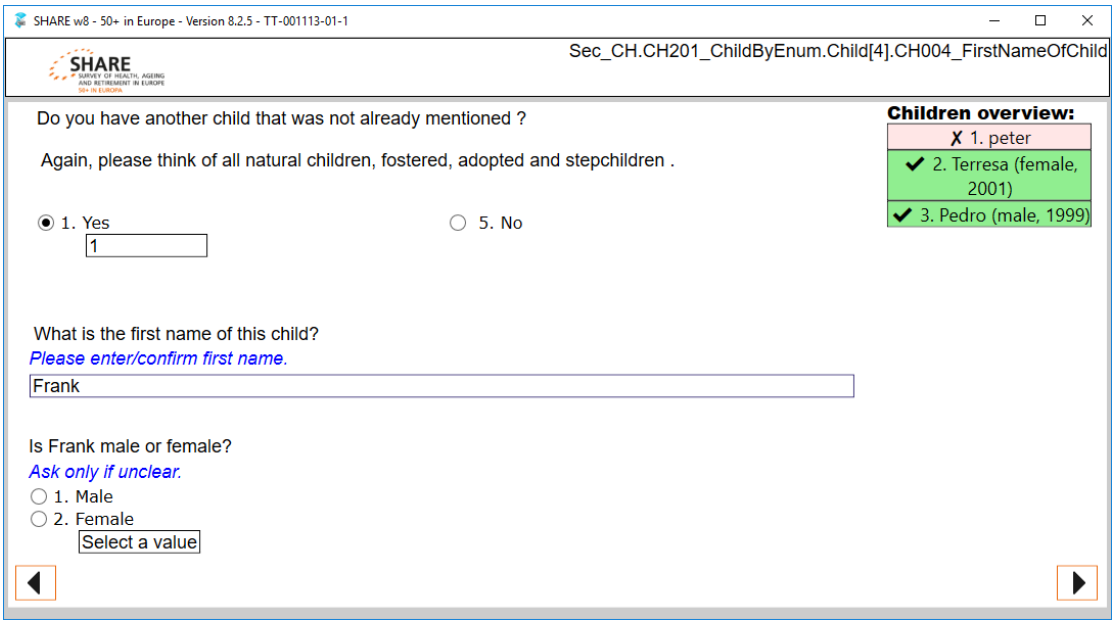


Figure 13: Children overview

The keystrokes are different from before. We gained better insight in loading times on question level. This created quite the discussion, since the definition of interview duration is depending on this, do we measure the time a question is shown on screen, or also include the time between when the response is submitted and the next question is shown, and to which question belongs that time, maybe not relevant for most surveys, but for a survey that may take on average an hour, this definition can be quite relevant.

In earlier waves, we would use the API to walk through a compiled questionnaire, and determined the questionnaire structure from that, for example to import into the translation environment. Since Blaise 5 exports XML definition we appreciate that, no need to call the API, simply parse XML made our systems more stable, also paper version interface, Stata scripts, various excel files with metadata overviews.

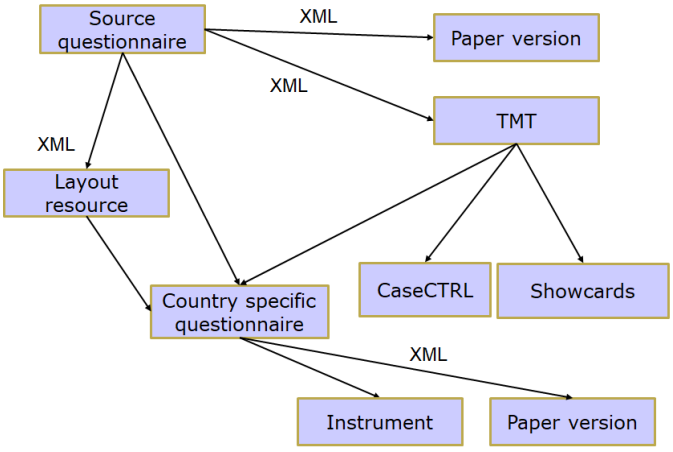


Figure 14: Use of Metadata XML

An extra feature we hoped for was to support the use of tablets for the SHARE system. Since Blaise 5 promised to support this, we knew that our SMS system would become a bottleneck in any future support. To overcome this a new SMS was developed, under the name CASE CTRL, this light weight program, can run on any device, it has a web interface and can link to many questionnaire engines. Linked to this we also replaced the SD in our systems and replaced it with Sample CTRL. This resulted in the tools described in figure 11.

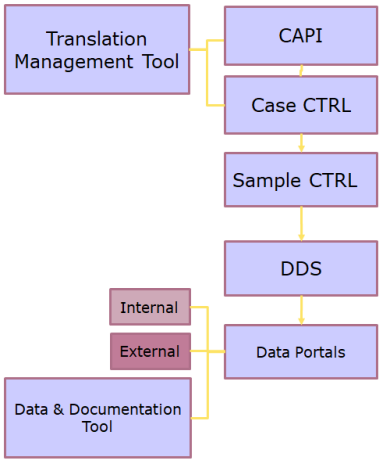


Figure 15: SHARE wave 8 tools

This resulted in the following changes for the SHARE wave 8 architecture:

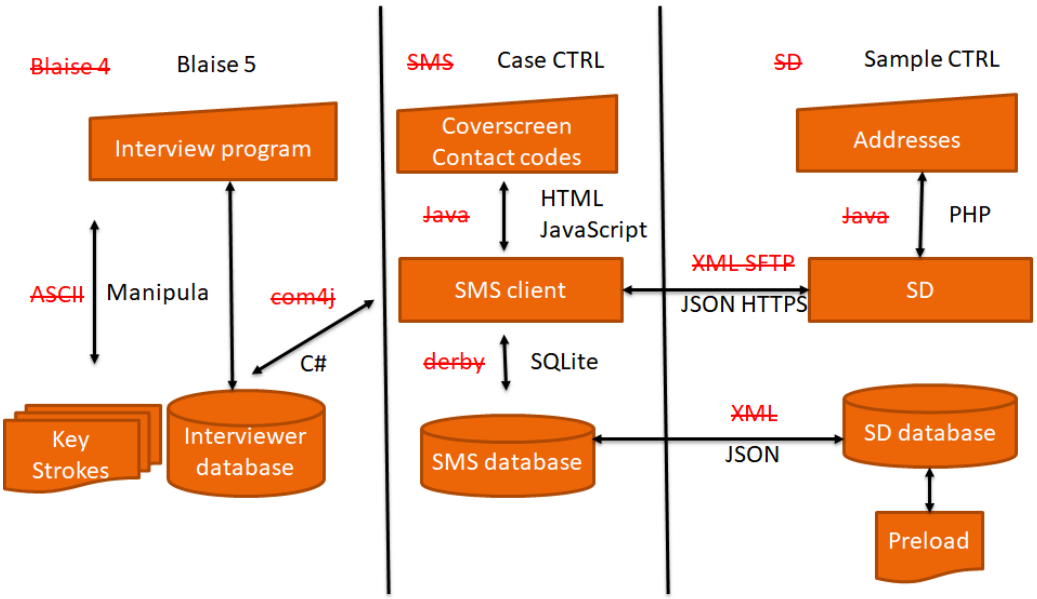


Figure 16: Changes for wave 8

4. Conclusion

The Blaise 5 environment is greatly improved compared to the Blaise 4 system. Its programming interface works much faster, which is convenient when developing a complex, long questionnaire like the SHARE questionnaire. The separation of layout and questionnaire definition is very useful. The introduction of text roles and the compatibility with Unicode solved many problems we had before.

There are some limitations on what we could do, which sometimes feels like they could be made possible if things were a bit more open. The burden of creating complete apps for some features is sometimes a bit overdone when you are very close to a solution within the Blaise environment. On the other hand, maybe we targeted the problems from a wrong angle.

The metadata XML export really gave us a solid reusable definition we used in various exports in pdf, html, excel and Stata script format. It would be perfect if from this definition a questionnaire could be build. This would allow us to generate instruments as an automatic integral part of the translation cycle.

Towards a modern mixed-mode Labour Force Survey

Trond Båshus, Statistics Norway

1. Abstract

The Labour Force Survey (LFS) at Statistics Norway is currently a CATI-only survey running in Blaise 4.8. Statistics Norway will phase out this version of Blaise in 2021 at the latest, which makes it necessary to convert the survey to Blaise 5. New EU regulations harmonizing European social statistics will also lead to changes in the LFS questionnaire. These two changes represent an opportunity to rewrite the questionnaire to support mixed mode data collection, but also to simplify the administration of the survey which currently is quite labour intensive. A pilot for a mixed-mode LFS survey was conducted in 2018, and the valuable lessons learned will be an important input for the development work.

This paper describes briefly the current LFS survey and the mixed mode LFS-pilot but will mainly concentrate on the changes planned for the new questionnaire. The goals for the new questionnaire are: 1. Offer a secure CAWI mode to the respondents, 2. A simpler and better structured Blaise questionnaire which should be easier to maintain, 3. More efficient data collection and simpler administration of the survey. We will also discuss changes to our case management system to facilitate mixed mode surveys, and other additions such as two-factor authentication for CAWI-surveys.

2. Introduction

Statistics Norway is at last on the verge to complete the transition to Blaise 5 from Blaise 4.8. We started using Blaise 5 for CAWI as far back as 2014 and made the first steps towards completing the move in 2018, when we started experimenting with CATI in Blaise 5. (Båshus 2016, Haslund 2018) Blaise 4.8 is now expected to be phased out during 2021. In this paper we will look closer on how LFS has been a driving force behind this development, despite being a survey many are reluctant making changes to.

3. Brief history of the LFS at Statistics Norway

The LFS has an almost 50-year history at Statistics Norway, and it has been carried out using a variety of modes through that time. The choice of modes has been dictated by technological possibilities, but also social developments, cost considerations and organizational issues has played a part. In 1972, when it was first fielded, the LFS was a uni-mode, or more precisely a PAPI, survey. The interviewer got lists of households and paper questionnaires in the mail and visited the respondents in their homes. As time went by and telephone coverage got better, the survey evolved into a mixed-mode survey: PAPI and PATI. If the respondents lived too far for the interviewers to visit and a telephone interview wasn't an option, the questionnaire could be mailed to the respondent, in effect making the survey a three mode survey: PAPI, PATI and PASI.

A major change came with the introduction of computer assisted interviewing (and Blaise) in 1996, which in essence worked as a computerized version of PAPI. A few years later, in 2000, a call centre was established. Most one-person households were routed to the call centres, while the rest of the LFS sample was sent to interviewers around the country. (Gravem 2011)

The next significant change in the LFS data collection happened in 2011/2012. A new case management system had been planned and built, and it was decided to move all interviewing to the Blaise CATI environment, with most interviewers working from the same central database. A system for navigating between the members of the same household was developed, at the same time as each person in a

household was treated individually in the CATI system. This is still how Statistics Norway collects data for the LFS. (Båshus 2012)

4. Current LFS questionnaire

Our current LFS questionnaire has been running since 2012 without any significant changes. The main form of interviewing is CATI interviewing from a central database, but offline interviewing is also possible, although the offline option hasn't been used for several years. Since every person in a household is a separate case in the database, only one household member at a time is normally active in the database. Appointments can be made for individuals in the household, and if there are several appointments within the same household, they will all be active in the daybatch. In case of offline interviewing the whole household would normally be sent to the same interviewer, and the household locked for online CATI-interviewing.

5. Current production system for social surveys

The data collection production system for social surveys at Statistics Norway is a combination of an in-house developed case management system which handles projects, samples, interviewers and payments. The system has good integration with Blaise 4.8, but the integration with Blaise 5 is still very basic. The consequence of this situation is that mixed-mode surveys must run in two different system: CATI on Blaise 4.8 and CAWI on Blaise 5. The reason for this is a combination of previously missing CATI functionality in Blaise 5 and lack of developer resources to update the case management system to work with Blaise 5. Working with mixed-mode surveys is therefore quite labour intensive since one must handle two questionnaires and two databases, which then have to be merged when the data collection period is over. (Båshus 2016)

6. Exploratory efforts

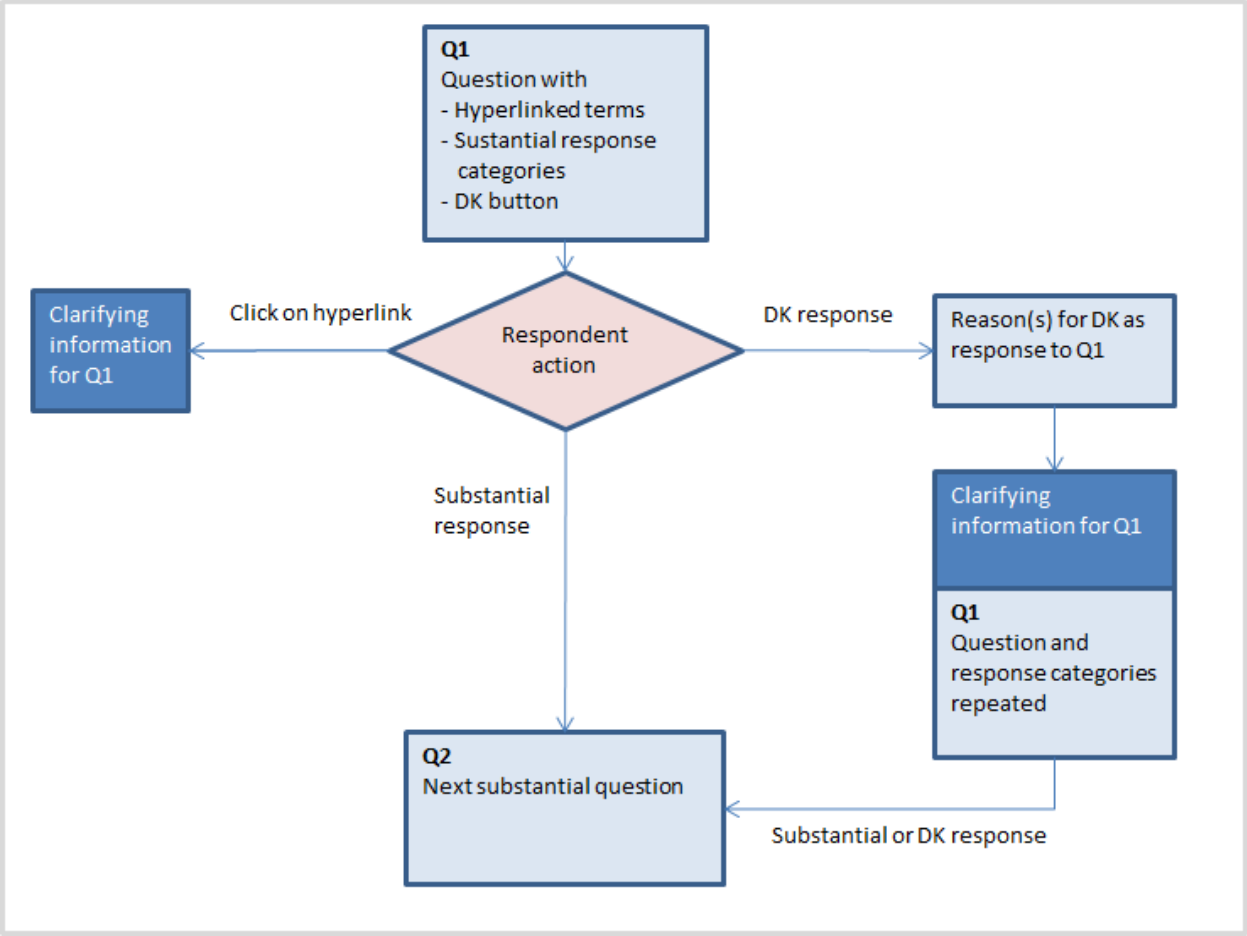
Statistics Norway has considered options for how to move as many surveys as possible to a mixed-mode platform for many years, and many surveys are offered on both CATI and CAWI, although the nature of the production system makes this less than optimal. Also, the nature of the LFS encourages a conservative attitude to changes in data collection. Despite this, we have undertaken several exploratory efforts, particularly under the umbrella of projects within the European Statistical System (ESS).

6.1 ESSnet DCSS

One such effort was an LFS CAWI questionnaire developed in 2012 and user tested piloted in 2013, as a part of the ESSnet project *Data Collection for Social Surveys using Multiple Methods* (DCSS). The aim was twofold. Firstly, to explore technical challenges concerning CAWI-surveys adapted to smart phones, and secondly to find alternatives to the interviewer-respondent dialogue in a CAWI questionnaire.

The survey was developed with Blaise 4.8, using the C-Moto stylesheet for mobile phones developed by CentERdata.

The survey contained hyperlinks on difficult terms which opened a second window with a definition and an explanation and contained follow-up sequences if the respondent answered "Don't know" on selected questions.



A random sample of 1500 aged 18-65 was drawn, and 121 people responded to the survey. The questionnaire dialogue experiments showed that respondents are reluctant to use both the hyperlinks and the follow-up to "Don't know", but that the hyperlinks were most successful. One conclusion was that additional work had to be carried out to make questions clearer and less ambiguous. (Gravem 2013)

6.2 Mixed-mode pilot (2018/2019)

A large scale mixed-mode pilot of the LFS was completed in 2018/2019. The pilot was conducted in part with support from the European Commission. The purpose of the survey was to:

- Create and test a mixed mode version of the LFS
- Improve and explore best practice for mixed-mode data collection
- Identify necessary improvements for the case management system
- Explore strategies for adaptive-responsive data collection

6.2.1 Questionnaire

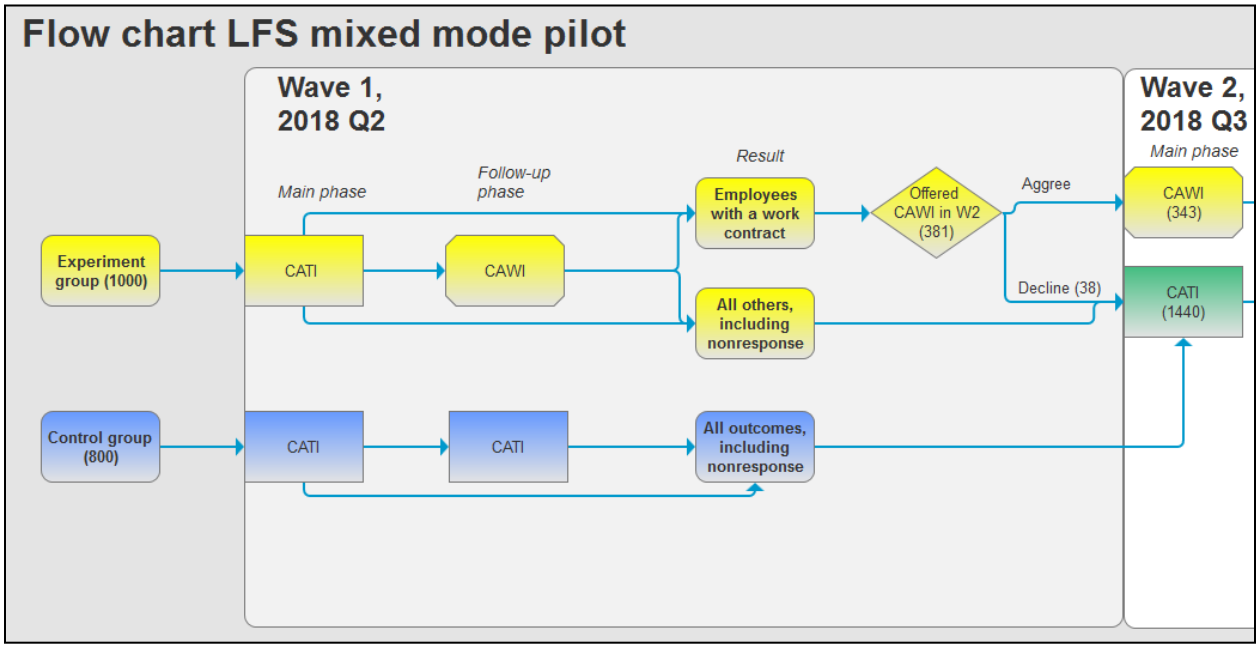
It was decided to convert the current Blaise 4.8 questionnaire to Blaise 5, instead of reusing the questionnaire made for the ESSnet DCSS project described above. The reason for this was that LFS questionnaire in production is more complete, and that a comparison with the ordinary LFS would be easier if the questionnaires for the pilot and the ordinary LFS were kept (relatively) similar. The conversion tools supplied with Blaise 5 were used, and only minor changes were necessary to get the questionnaire to compile and run. Several questions which could be troublesome in a CAWI context were

reformulated and the questionnaire was adapted to two modes, CATI and CAWI, and three layouts: One layout for CATI and two for CAWI. The CAWI layouts were a large layout for desktop computers and a small layout suitable for mobile phones. Since the pilot was to run for four rounds, the questionnaire was incrementally improved each round. Input from an interaction designer in cooperation with survey methodologists proved especially useful to update the look, feel and usability of the questionnaire.

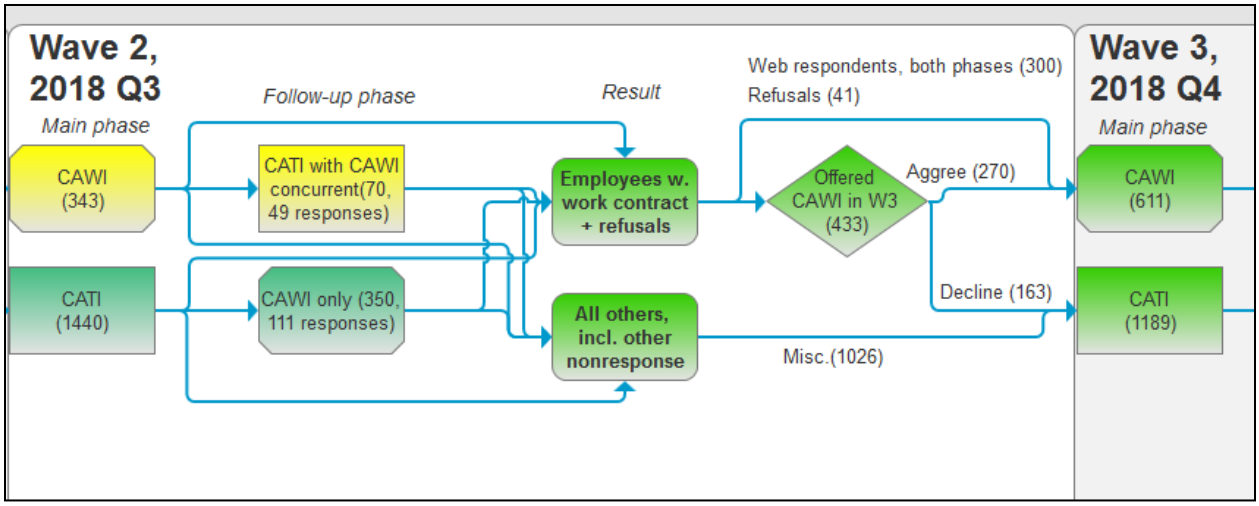
In addition to adapting the questionnaire to Blaise 5, it was also necessary to build an improvised production system to prepare data from previous waves and to prepare register information for the first rounds. This was necessary because it would have required to much work to adapt the ordinary production system for the pilot.

6.2.2 Waves, modes and mode switching

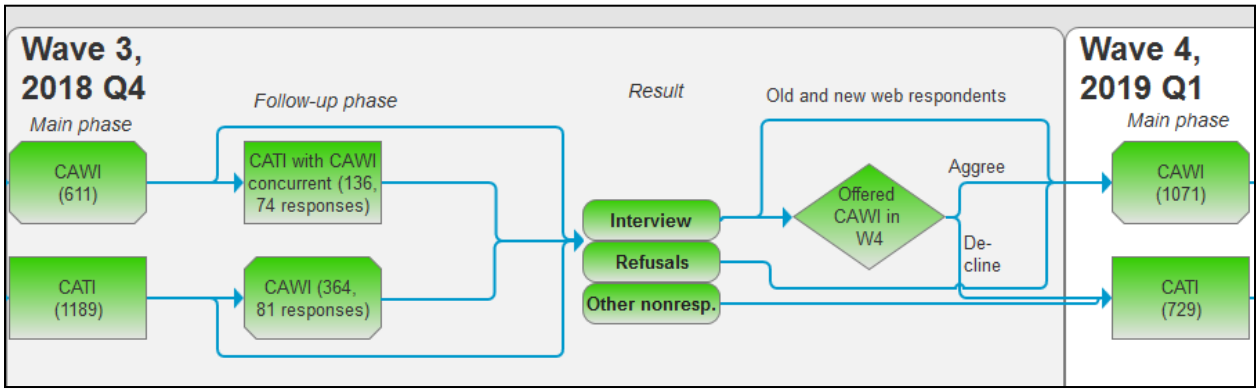
The sample was divided in two groups, an experiment group of 1000 and a control group of 800. For the first wave it was decided to offer CATI as the first mode for both groups, but the experiment group was offered CAWI in the follow-up phase. Employees with a work contract in the experiment group in the first round were offered CAWI mode for second round. The design of the survey was such that the number of respondents being offered CAWI as the initial mode got larger for each subsequent round.



Figur 1Phase plan for first and second wave



Figur 2Phase plan for second and third wave



Figur 3Phase plan for third and fourth wave

6.3 Findings

The main findings of the pilot are that the potential cost savings offered by a mixed-mode CATI/CAWI survey, compared to a single-mode CATI, are considerable. The response rates as well are comparable to single-mode CATI, bearing in mind that the LFS is compulsory in Norway. 53% of the respondens in the pilot would prefer to answer the questionnaire on CAWI, rather than CATI (preferred by 29%).

Another finding is that the transition to mixed-mode data collection requires investments in important infrastructure such as the case management system. Despite the potentially large cost savings in moving to mixed-mode, as documented in the pilot, it can be surprisingly difficult to get funding and resources to do the necessary upgrades to core systems. (Gravem et al. 2019)

7. New questionnaire

The development work for a new LFS questionnaire is now well underway, and the main reason that this is happening in 2020 is the *Framework regulation for the production of european statistics on persons and households* or *Integrated European Social Statistics* – IESS. This new regulation requires significant changes to both the questionnaire and the production system behind it, changes which would be difficult to implement just by modifying the current Blaise 4.8 LFS questionnaire. Also, it is an opportunity to

move forward to a more up to date mixed-mode data collection, building on our own experiments and pilots surveys, and other institutions' experiences.

The case management system at Statistics Norway, Sivadm, has been in severe need of an update to support mixed-mode surveys and Blaise 5 for many years, but a general lack of IT-resources, and not least lack of will to give Sivadm priority, has led to a situation where Sivadm does not support our data collection processes in a way we want. A situation that has forced us to use considerable resources to improvise temporary solutions. Luckily, the new regulation affects several important social surveys, not only LFS, but also EU-SILC and others. The most pressing issues and missing functionality in Sivadm are now being addressed, and we expect the system to be ready for production this fall.

7.1 Standardization

We have used Blaise 5 for CAWI since about 2014/2015 and have during the last couple of years tried to harmonize and standardise the look and feel of our CAWI-surveys. A common resource database is used in most of our surveys, and we have also common templates (consisting of a blax and several incx files). At the moment we have a simple uni-mode template supporting only CAWI, and another mixed-mode template supporting CATI, CAWI, and at some point in the not too distant future, CAPI. Standardized templates are necessary for the integration with the case management system to work.

Typically, most mixed mode survey will be made with three different layouts: two for CAWI and one for CATI. The CAWI layouts are specifically made for either smartphones or desktop browsers.

Work on a layout suitable for interviewers was started when we first tested CATI in Blaise 5 for the survey *Governing and Experiencing Citizenship in Multicultural Scandinavia* (GOVCIT), this work was again taken further in the LFS pilot which we conducted in 2018/2019. (Haslund 2018) The interviewers haven't been entirely happy with the usability of the interface, especially when in comes to navigation within the questionnaire, and we have therefore made an effort to make it easier to use a keyboard. In most places, it is now possible to navigate just by using a keyboard or more specifically the numeric keypad. Among other improvements is a Click-to-call system, so that the interviewers don't have to enter the telephone numbers manually any more. This has been a long-standing item on our wish list.

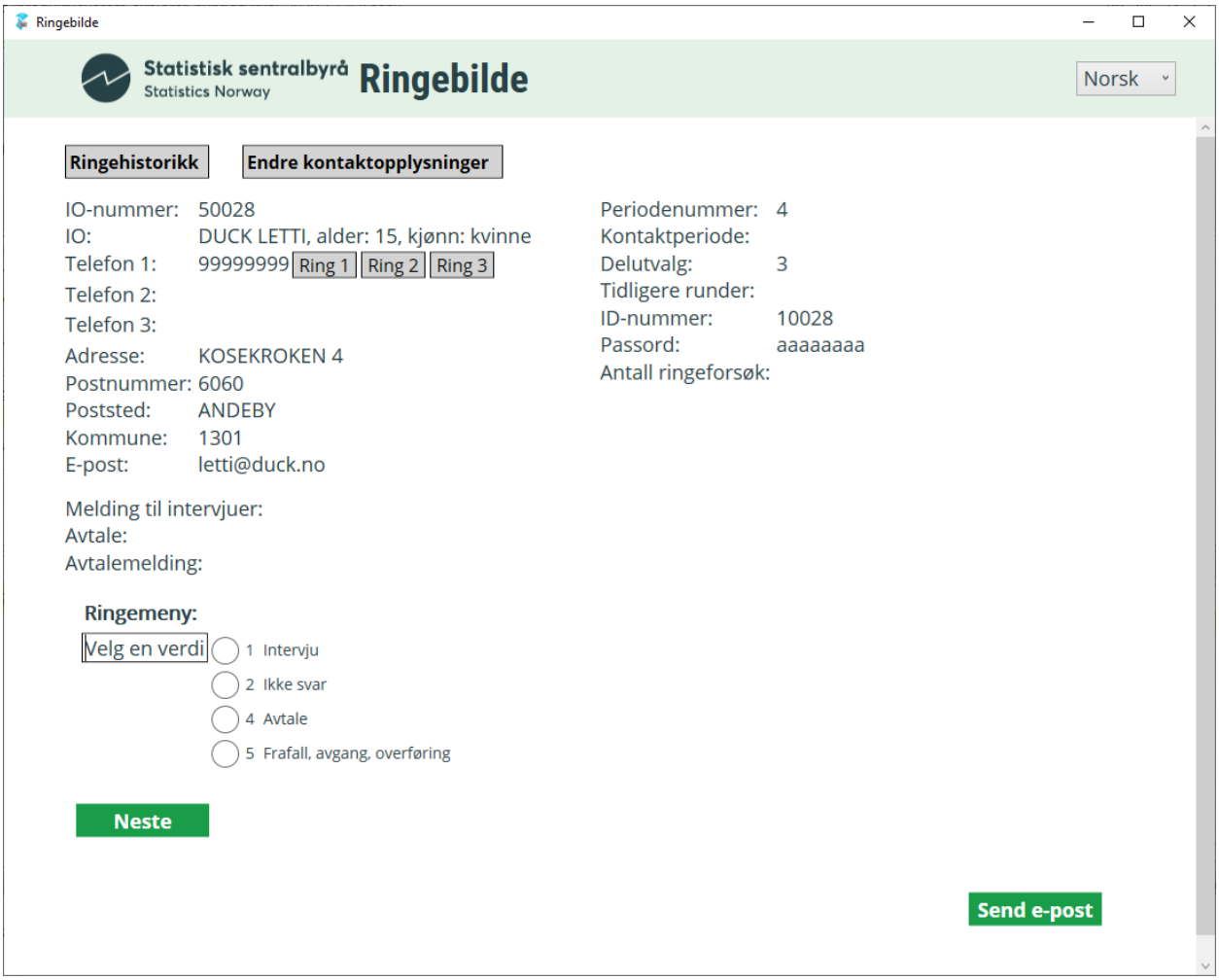


Figure 17Dial questionnaire with Click-to-call buttons

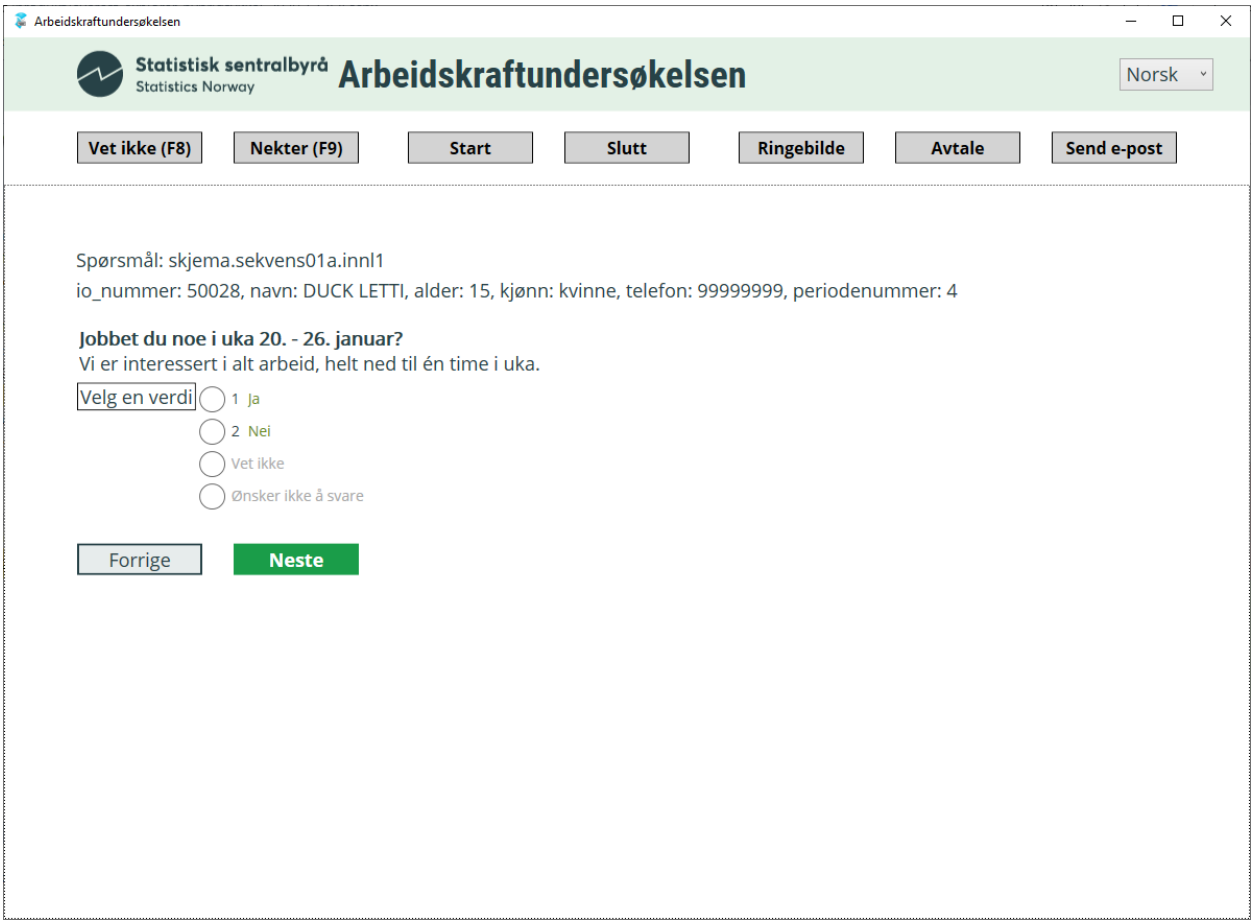


Figure 18Example of a CATI questionnaire layout

The current LFS Blaise 4.8 questionnaire is quite old. Much of the code was written in 2006, but it also contains code going as far back as 1996. The questionnaire is rather complicated, and it does not help readability that many Blaise developers, with sometimes significantly different coding styles, have been working on it. One of the aims with the new Blaise questionnaire is to make better organized and more readable code, which also is better commented and documented.

Even though the questionnaire will be ready for mixed-mode in January 2021, it will be used a CATI only questionnaire, possibly for the first year. The questionnaire will be flexible enough for parallel or sequential mode designs. E.g. CAWI first with follow up on CATI, CATI first with follow up on CAWI, or both at the same time if that is desired.

The sample will change from a family sample to a person sample, but we will interview the persons belonging to the respondents' household in one of the waves. The Blaise questionnaire will not contain a household roster, and the household will instead be established on the basis of register information.
The questionnaire will initially only be available in one language, but because of legal obligations, it must be made available in both written standards of the Norwegian language: *bokmål* and *nynorsk* when it starts on CAWI. English will also be an option, and possibly Polish. We anticipate that the new translation features in Blaise 5 will offload some of the work related to translation (copy and pasting of text) from the Blaise programmers.

As mentioned before, the case management system, Sivadm, is being updated to handle CATI in Blaise 5. The most crucial problem has been updating contact information such as addresses and telephone numbers between the two systems. Another improvement will be to make data about appointments available in the case management system. This will for instance make it possible to send out automatic reminders about hard appointments on SMS or e-mail, but it will also a serve as useful indicator of the future response potential in a survey.

A separate Blaise questionnaire is currently prepared quarterly, requiring us to prepare and install a new LFS Blaise questionnaire four times per year. From 2020 we hope to reduce this to one per year, easing the overall workload. Subsequent cases and waves can easily be uploaded as needed during the year. Data contained within the sample and external files will ensure that the questionnaire work properly.

7.2 Future improvements

For the use of CAWI on the LFS to be a viable option, the security is very important. Especially because the LFS will contain much information from registers and previous waves, which the respondent is asked to confirm or update. Two-factor authentication is therefore essential. The pilot conducted in 2018/19 used a form of secure login, and this worked well with very high response rates, at least in combination with a survey which is mandatory.

More advanced and automated use of audit trail and call data, for error detection and as an input to adaptive-responsive design are under development. Currently we are creating modules to parse unstructured paradata and call data from Blaise into Python Pandas Dataframe. DataFrame is a 2-dimensional labeled data structure with columns. Further, we will visualize those data with Plotly Dash.

Data delivery from the current LFS is completely automated and is in the form of a flat file which is exported from the Blaise database every night. This file is then imported into another database. In the future we will instead write the data directly to a database, removing an unnecessary step in the data delivery process.

8. Conclusion

The LFS data collection processes generally are conservative when it comes to changes, characterized by stability for usually a decade or more. Usually only small adjustments are made (and permitted). At the same time, it has also been a driving force behind experimentation and exploration of new methodologies and technologies, which in turn has benefited other surveys. When the new LFS survey is in place from 2020, we can perhaps expect the cycle to repeat itself with a long period of stability, while we can continue to prepare for and adapt to a changing technological and cultural landscape.

9. References

Båshus, Trond. *A system allowing sequential interviewing of household members, where the household members are individual cases, within the Blaise CATI-framework.* IBUC, London, 2012.

Båshus, Trond. *Blaise 5 at Statistics Norway*. IBUC, The Hague, 2016.

Gravem, Dag F. *Towards an integrated mixed-mode case management system for the Norwegian LFS.* Paper presented at the Workshop on Labour Force Survey Methodology, Wiesbaden 12-13 May 2011

Gravem, Dag F. *Error prevention through interviewer emulation? Introducing questionnaire dialogues in the Norwegian LFS questionnaire.* Paper presented at ESRA conference, July 16-19 2013, Ljubljana, Slovenia.

Gravem, Dag F. et. al. *Report from Statistics Norway's adaptive/responsive mixed-mode LFS pilot.* Cooperation on Multi-Mode Data Collection (MMDC). Quality Improvements for the Labour Force Survey. European commission. Publication forthcoming, 2019.

Haslund, Jan et. al. *Experiences with Blaise 5 CATI and multimode at Statistics Norway.* IBUC, Baltimore, 2018.

Using Field Properties in Blaise 5

Charles Less and Peter Kilpatrick, United States Department of Agriculture – NASS

1. Abstract

By declaring Field Properties of remarks, original values, and overlay values in our Blaise 5 instruments, we are able to store supplemental Field level values and use those stored values for multiple purposes that include quality control and data analysis. Being a statistical agency, we are keen to assist our analysts with the necessary information and tools to ensure that our data are accurate. Using Field Properties effectively allows us additional layout options and area to store data without adding additional programming burdens for our developers of individual projects.

2. Introduction

Blaise 5 (B5) introduced a new datamodel section named FIELDPROPERTIES. This section allows developers to more fully utilize and customize auxiliary information. Field properties – when used in combination with the Blaise Resource Database (BLRD), manipula, and/or third party software (Visual Studio) – provide valuable supplemental information to enumerators, analysts, and researchers. Our paper provides insight and description of how we have begun using field properties at NASS and what our plans are for the future.

A note about our setup. At NASS we do not use the default data storage of a BDBX (SQLite infrastructure). Our agency has required that we use a centralized database for storing all our Blaise surveys in the same database in the same format and we were told we had to use MySQL. At the time this decision was made (ca. 2009), Statistics Netherlands (SN) worked with our lead developers in order for us to deploy a database infrastructure that uses MySQL tables in a format SN refers to as Generic In-Depth. In Blaise 4 (B4), we had a BOI file that connected to a MySQL generic in-depth set of Blaise_* tables. In B5, we have a BDIX file that connects to a MySQL generic in-depth set of Blaise_* tables.

3. Remark: Default Field Property

Our introduction to field properties came when reviewing the reference manual on the subject while trying to get remarks to work as they did in B4. SN has outlined clearly how Remarks work in B5. By following the documentation provided, we were able to implement a more or less standard Remark/Comment section. We found that after adding the Remark field property (Image 1) to our instruments we were able to record and view remarks similar to how we did in B4. Adding this Remark field property activated the Remarks functionality for Field Templates (Image 2, 3) and also activated a new MySQL table named Blaise_FieldProperty (Image 4). This Blaise_FieldProperty table is similar to the Blaise_Remark table in B4, but it led us to ideas and inspiration to do more with the new FieldProperties section.

Image 1: Default Remark field property as described by SN.

```
DATAMODEL LNDVAL200000  INTERVIEWER "Enumerated Land Values Survey"
                        FIELDINTERVIEWER "Field Enumerated Land Values Survey"
                        EDITOR "Interactive Edit for Land Values Survey"
                        SELF "Self-Administered Land Values Survey"
                        DATAENTRY "Data Entry Interface for Land Values Survey"

SETTINGS
  ATTRIBUTES = DONTKNOW, REFUSAL, NOEMPTY
  ROLES = EditMask, Watermark, Help, SectionIndexName, IndexCheckMark, InterviewerNotes, Speedkey

FIELDPROPERTIES
  Remark: open
  OriginalValue: string
  OverlayValue: string
```

Image 2: Vertical field template of the BLRD showing how the Remark field property is referenced.

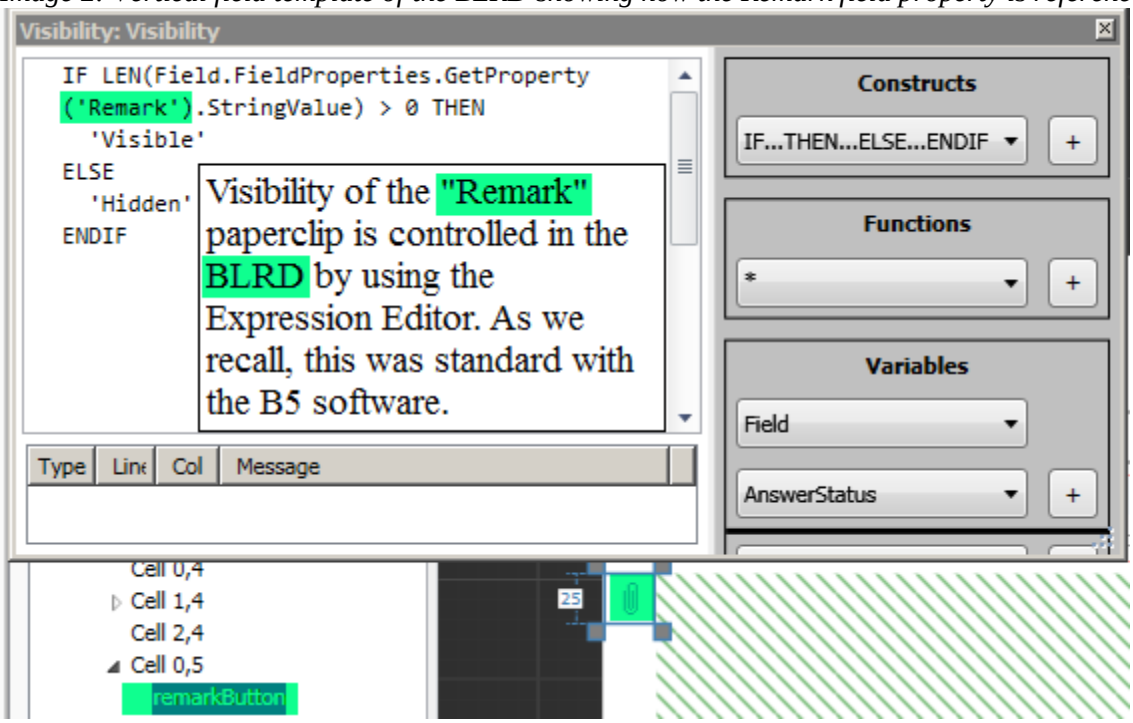


Image 3:The remark paperclip is visible for our users and the remark area is visible if necessary.

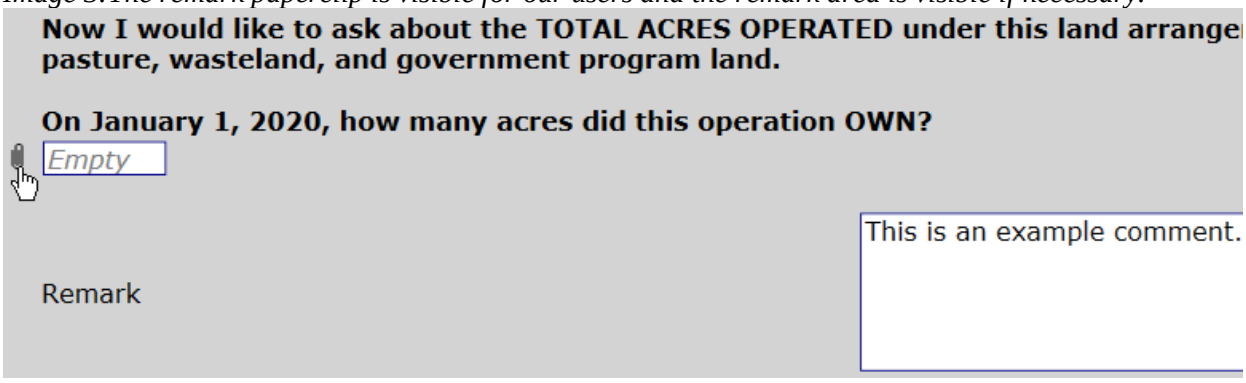


Image 4: View of MySQL table Blaise_FieldProperty.

FORMID	DMKEY	FIELDID	PROPERTYNAME	INTEGERDATA	TEXTDATA
32026	11	16.4	Remark	NULL	inaccessible
32054	11	21.14	Remark	NULL	Hello
32365	11	21.14	Remark	NULL	I am leaving a comment to run the I
32385	11	13.8	Remark	NULL	CAWI: This is a comment
32788	11	17.13	Remark	NULL	Test comment KILPPE
33058	11	21.0	Remark	NULL	CAWI: Mail returned. Inaccessible

4. Development of OriginalValue and OverlayValue Field Properties

The straightforward Blaise_FieldProperty table in MySQL gave us ideas we wanted to follow up on. We set about devising a field property method for recording the original values of a record when the data is first collected. The original values of a record are of particular use at NASS because we have a large editing/cleaning component in between our data collection and summary and there are desires to understand what values are changing after being collected and why.

We also created a similar field property for storing a second copy of the data in the event a record is recorded via two different methods. For example, some respondents will complete a survey over the phone and will also mail back a questionnaire on the same day. We store the StrictCATI data in the actual fields, and put the data from the mailed in questionnaire in our OverlayValue field property.

The two new field properties added to the FIELDPROPERTIES section (Image 1) were OriginalValue and OverlayValue. We chose to use STRING as our type for these field properties as all numeric and date types can be string, but string cannot be numeric or date. This has not been a problem yet, but long OPEN fields could be a problem for us at a point in the future. Most of the data we are interested in is either INTEGER or REAL. Image 5 shows a record that has both OriginalValue and OverlayValue stored in the MySQL Blaise_FieldProperty table. Note the difference in Image 4 and Image 5. STRING field properties are stored in the StringData column, while OPEN field properties are stored in the TextData column.

Image 5: Field_Property Table - In this example, we have a record that has OriginalValue differing from OverlayValue meaning that the respondent two different values at different times.

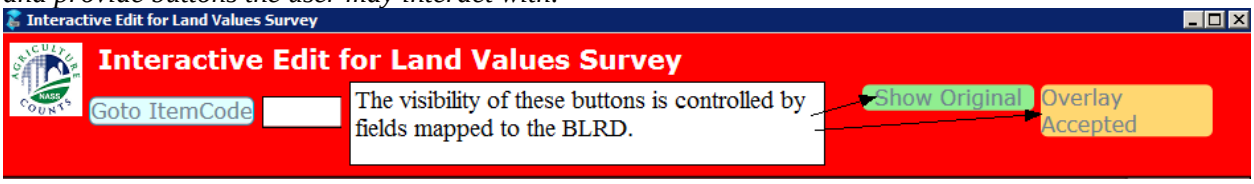
FormId	DMKey	FieldId	PropertyName	IntegerData	StringData	TextData
89140	118	21.21	OriginalValue	NULL	250000	NULL
89140	118	21.21	OverlayValue	NULL	200	NULL
89140	118	21.4	OriginalValue	NULL	20000	NULL
89140	118	21.4	OverlayValue	NULL	1	NULL

In addition to the new field properties we also added two new fields to our instruments – Managmnt.OriginalValueSwitch and Managmnt.OverlayValueSwitch (Image 6). These fields are mapped to the BLRD so that we can show a button for users to interact with (Image 7). The fields are also used to limit reassigning to these field properties after they have already been assigned.

Image 6: New fields added for our instruments to fully utilize the new field properties.

```
OriginalValueSwitch "Original Value has been set?<NEWLINE>
            EMPTY, 0 = OriginalValue was not set<NEWLINE>
                   1 = OriginalValue was set<NEWLINE>
                   2 = Something was done with OriginalValue" : 0..9 {B5 5/2019}
OverlayValueSwitch "Overlay value information:<NEWLINE>
            EMPTY, 0 = Record does not have an overlay <NEWLINE>
                   1 = Record has an overlay present<NEWLINE>
                   2 = Overlaid record was accepted" : 0..9 {B5 5/2019}
```

Image 7: Managmnt.OriginalValueSwitch and Managmnt.OverlayValueSwitch are mapped to the BLRD and provide buttons the user may interact with.

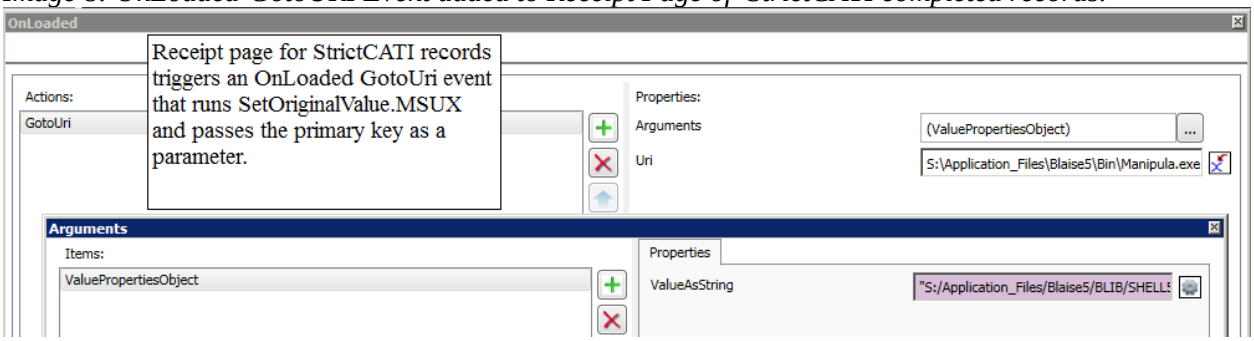


5. Populating OriginalValue and OverlayValue with Manipula

To populate the OriginalValue and OverlayValue, we experimented with a variety ideas but have settled on using Manipula. OriginalValue is set by two manipulas currently: SetOriginalValue.MANX and B5ReadIn.MANX. OverlayValue is set by B5ReadIn.MANX.

For each appropriate record, SetOriginalValue.MANX populates the OriginalValue for every field that is not empty. When a StrictCATI interview event reaches the Receipt Page, an OnLoaded-GotoURI event is triggered which runs SetOriginalValue.MSUX for the primary key of the record that was just completed. Image 8 shows the GotoURI event in the BLRD.

Image 8: OnLoaded-GotoURI Event added to Receipt Page of StrictCATI completed records.



SetOriginalValue.MANX is a straightforward manipula that filters and sets the OriginalValue field property for the one record that was just completed in StrictCATI mode. Image 9 shows the Manipulate section and Image 10 shows the structure of OV.INCX file where the OriginalValue field property is set. OV.INCX is automatically generated at the beginning of the survey (Survey Setup) by using a modified version of SN's ListAllFields.MANX. The exact syntax for assigning a field property via manipula is –

FieldName.FieldPropertyName := 'This string will end up in the FieldProperty'

Image 9: SetOriginalValue manipula 1) filters the record that was StrictCATI completed, 2) verifies that the record needs OriginalValue set, 3) uses OV.INCX to set OverlayValue field property for every non empty field, 4) sets the Managmnt.OriginalValueSwitch t

```
PROLOGUE
  aConfigline := 'LFINFO.KeyValue IN (\''+PARAMETER(1) + '\')'

MANIPULATE
  SurveyData.SETRECORDFILTER (aConfigLine)
    REPEAT
      SurveyData.READNEXT
      IF SurveyData.RESULTOK THEN
        IF Managmnt.ProcessSwitch <> Deflt AND SurveyData.Managmnt.OriginalValueSwitch <> 1 THEN
          INCLUDE "Surveys\LNDVAL\LNDVAL200000\OV.INCX"

          DISPLAY('aConfigLine: '+ aConfigLine + '<NEWLINE>
                   KeyValue: ' + LFInfo.KeyValue,WAIT)
          SurveyData.Managmnt.OriginalValueSwitch := 1
          SurveyData.CHECKRULES('EDITOR') {This to get Integral Check Early 9/20/2019}
          SurveyData.Write
        ENDIF
      ELSE
        DISPLAY('IOResultCode: '+ STR(SurveyData.IOResultCode) + '<NEWLINE>Did not ReadNext: ' + aConfigLine,WAIT)
      ENDIF
    UNTIL SurveyData.LastRecord OR SurveyData.IOResultCode > 0
```

Image 10: OV.INCX has logic to set every field in the instrument's OverlayValue field property when the field is not empty. OV.INCX was automatically generated during survey setup by using a modified version of SN's ListAllFields.MANX.

```
IF Sec10LandVal.RentFrom = REFUSAL THEN
  Sec10LandVal.RentFrom.OriginalValue := 'Refusal'
ELSEIF Sec10LandVal.RentFrom = DONTKNOW THEN
  Sec10LandVal.RentFrom.OriginalValue := 'Don\'t Know'
ELSEIF Sec10LandVal.RentFrom <> EMPTY THEN
  Sec10LandVal.RentFrom.OriginalValue := STR(Sec10LandVal.RentFrom)
ENDIF
IF Sec10LandVal.Rent_to = REFUSAL THEN
  Sec10LandVal.Rent_to.OriginalValue := 'Refusal'
ELSEIF Sec10LandVal.Rent_to = DONTKNOW THEN
  Sec10LandVal.Rent_to.OriginalValue := 'Don\'t Know'
ELSEIF Sec10LandVal.Rent_to <> EMPTY THEN
  Sec10LandVal.Rent_to.OriginalValue := STR(Sec10LandVal.Rent_to)
ENDIF
```

For interviews that are mailed back to us, we have B5ReadIn.MANX for reading the data from an ASCII file in to our instrument. Using indicators that have determined whether or not a record needs OriginalValue and/or OverlayValue assigned, the PutValue statement is adjusted accordingly. Image 11 shows how a field property is set using PutValue in our B5ReadIn Manipula.

Image 11: B5ReadIn.MANX was made generic with VAR().All field and field property assignments are done with PutValue.

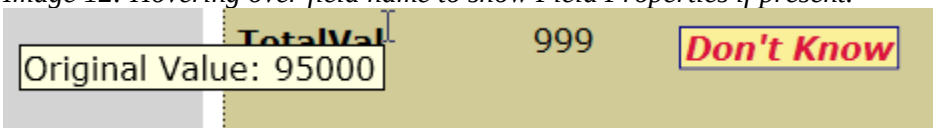
```
IF piNoOriginalValue <> 1 THEN
  SurveyData.PUTVALUE(piICFieldName,'OriginalValue',piCell_Value)
ENDIF
IF piIsOverlay <> 1 THEN
  SurveyData.PUTVALUE(piICFieldName,piCell_Value)
ELSEIF piIsOverlay = 1 THEN
  SurveyData.PUTVALUE(piICFieldName,'OverlayValue',piCell_Value)
ENDIF
```

It's worth noting that we spent time trying to use the BLRD to assign OriginalValue using Events like OnValueChanged, and OnLeave when it was missing, but we did not get the consistency we wanted when compared to using Manipula at the end of an interview or during ReadIn.

6. Uses for Blaise Field Properties

We have displayed our new field properties in two ways thus far – 1) from the BLRD while users are editing, and 2) running a manipula that reads a virtual table (a MySQL view) using QueryFile to write an XML output that is converted to HTML. In the first method, while our editors are cleaning a record, they are given a visual indicator about whether a record has original and/or overlay value data (Image 7). When one or both of these buttons are visible, the user can hover over the field name in order to see OriginalValue and/or OverlayValue data. Image 12 shows how the hover appears for users. This hover event was set up fairly easily by editing InfoPane Field Template of the .BLRD.

Image 12: Hovering over field name to show Field Properties if present.

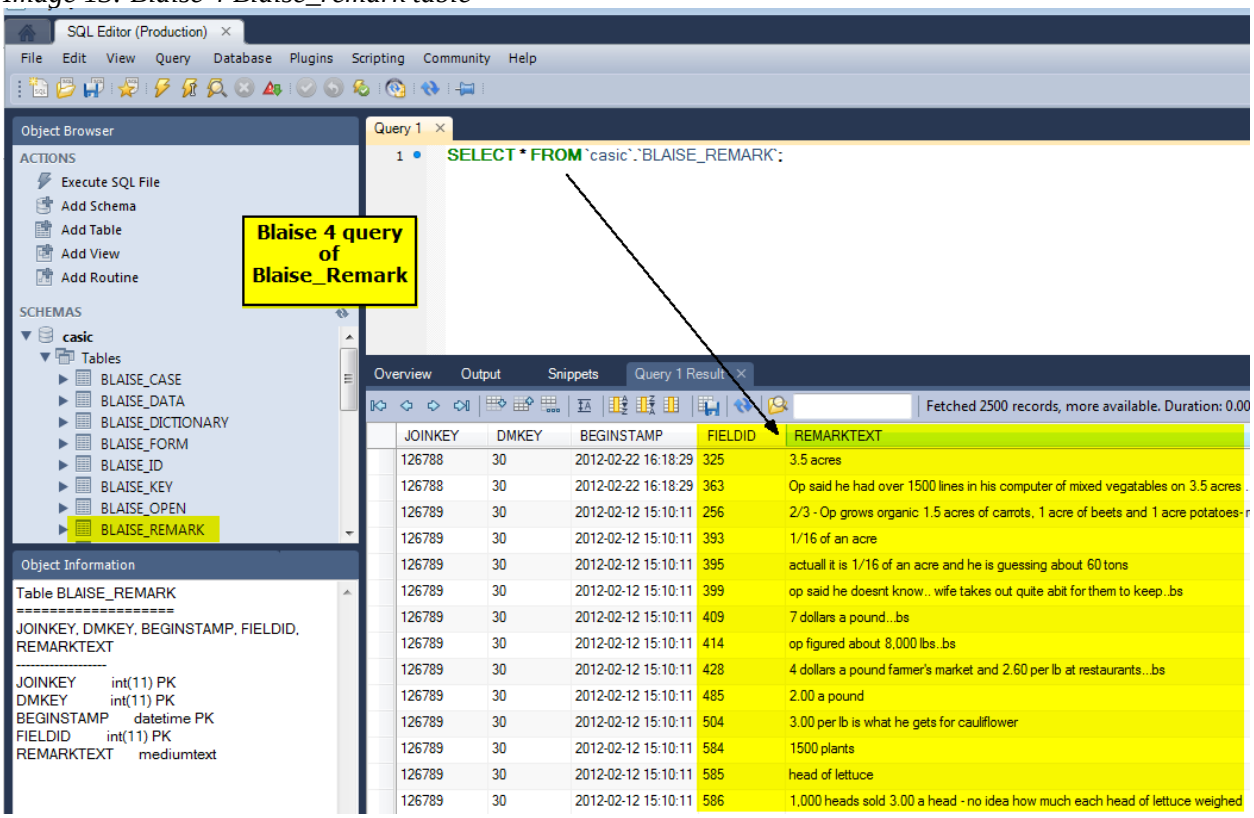


In the long term we plan on developing a manipula that runs from an OnClick button event from the edit that will "Restore Original" and "Accept Overlay" but we have not developed these features fully.

6.1 Blaise 4 GenInDepth Data Storage

At NASS, we utilize the Generic In-Depth data storage type to store our responses. One of the important utilities was the Blaise_remark table which would explicitly hold remarks for a Blaise field. During the time of B4, we would access this information through the use of manipulas in order to display the remarks for a record or to pass this information on to another system. The manipulas seemed effective enough for users to see remarks displayed within our editing system. We did not look too closely at possible ways to utilize the remarks through the Blaise_remark table. What follows is small query of the Blaise_remark table (see Image 13):

Image 13: Blaise 4 Blaise_remark table

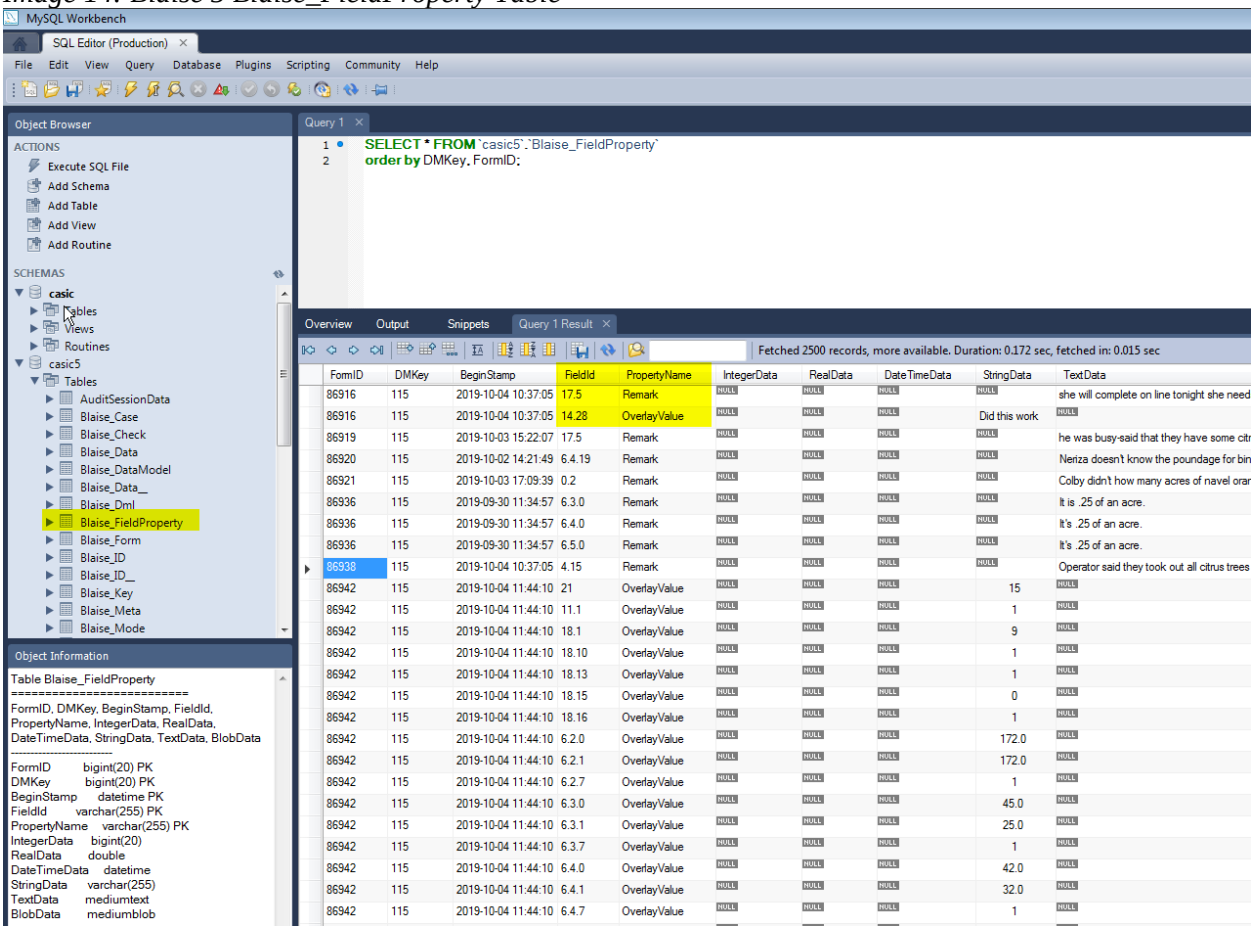


Other issues came into being as B4 gained more prominence in NASS data collection/editing activities. At NASS, we also heavily rely on Blaise as our editing tool for our analysts. As a result of this heavy editing presence, we were often asked to provide more information about the existence of previous responses to Blaise fields and/or original responses to our Blaise fields in our agricultural surveys. And when we tried to use the versioning of Blaise 4, this was not possible because the system became weighted down with keeping the many iterations of editing. Keep in mind, versioning was an issue several years ago when memory was a problem for us. Our DB administrators would wave us off from using versioning.

6.2 Enter Blaise 5 - A New Hope

With the use of B5, we still use Generic In-depth datatype storage. During the early days of comparing B4 MySQL tables to B5 MySQL tables, we noticed some definite differences in tables, column names, and data types. We had to adjust our thinking with new tables and this was a catalyst for us to see what could be gained from these new tables. After realizing that the Blaise_Remark was replaced by the Blaise_FieldProperty (Image 14), we understood that it was designed to be used for more than just Remarks and we set to work to expand on the concept.

Image 14: Blaise 5 Blaise_FieldProperty Table



6.3 The Rise of the Virtual Table

In an effort to facilitate rapid understanding of what is happening for an individual record's response, we have create MySQL views. Our users want a display of our specific NASS KeyValue for a respondent along with corresponding data they understand. Outside of the database, the FormID and dmkey mean very little to users. What follows is a view we created that is a NASS friendly version of the Blaise_FieldProperty table (Image 15). Please note that this view is almost the same as the Blaise_FieldProperty from Image 5, but we have included in this newly created virtual table, the descriptive information that our analysts need to make a quick connection to a NASS record, such as our specific KeyValue from Blaise_Case, as well as the FieldTag and the DescriptionText from Blaise_ID.

Image 15: A MySQL view created by merging Blaise_FieldProperty with Blaise_Case to get KeyValue and Blaise_ID to get Field Names

```
1  SELECT * FROM `casic5`.`viewFieldProperty`
2  where dmkey=118 and KeyValue = '9300789260 1 1'
3  and FieldTag in(999,410);
```

erview | Output | Snippets | Status_Report_Current.sql Result | AvgNumCalls.sql Result | Query 3 Result | SQL File 2 Result | Bla

Fetched 4 records. Duration: 0.000 sec, fetched in: 0.000 sec

KeyValue	FormID	DMKey	FieldTag	DescriptionText	ItemName	PropertyName	StringData	TextData
9300789260 1 1	89140	118	999	MLDBDOOR	Sec10LandVal.TotalVal	OriginalValue	250000	NULL
9300789260 1 1	89140	118	999	MLDBDOOR	Sec10LandVal.TotalVal	OverlayValue	200	NULL
9300789260 1 1	89140	118	410	CLAND410	Sec10LandVal.CroplandVal	OriginalValue	20000	NULL
9300789260 1 1	89140	118	410	CLAND410	Sec10LandVal.CroplandVal	OverlayValue	1	NULL

We have used this view in Manipula that accesses it via the QUERYFILE statement. We are able to access the MySQL view, get the information we need, and output it to XML. This has been done to great effect in order to create Original Value reports for users in the event they need to review in depth how the record originally came in to the edit. Image 16 below shows an HTML report created by a third party software to read the Blaise XML and format for web. This report provides our users with information they understand in a format and terminology they can read.

Image 16: View of Original Value data in a web browser.

ID	ItemCode	MasterVarname	OriginalValue	BlaiseField
Search ID	Search ItemCode	Search MasterVarname	Search OriginalValue	Search BlaiseField
4300128150 1 1	518	CLAND518	3	Sec10LandVal.Change
4300128150 1 1	900	CLANDTOT	126	Sec10LandVal.Operate
4300128150 1 1	901	CLANDOWN	26	Sec10LandVal.Owned
4300128150 1 1	513	CLANDPAS	20	Sec10LandVal.Pasture
4300128150 1 1	413	CLAND413	15000	Sec10LandVal.PastVal
4300128150 1 1	902	CLANDRFM	100	Sec10LandVal.RentFrom
4300128150 1 1			1	Sec10LandVal.Confirm_

7. Conclusion

With B5's new FIELDPROPERTIES section, we have new capabilities for storing, manipulating, and displaying supplemental information. Our future plans include developing a Previously Reported Data field property for carrying information from an earlier survey. We can access and display these field properties via the BLRD (our hover example), via Manipula (our HTML example), and we also have an effective new way to display these properties that can be accessed outside of Blaise via a MySQL view.

8. Further Reading

FieldProperties Section – http://help.blaise.com/Blaise.html?ref_fieldproperties.htm
PutValue – http://help.blaise.com/Blaise.html?ref_putvalue.htm
VAR() – http://help.blaise.com/Blaise.html?ref_uses.htm
Map Fields to BLRD – http://help.blaise.com/Blaise.html?ldmapfields.htm
QueryFile – http://help.blaise.com/Blaise.html?ref_queryfile.htm

The views expressed in this paper are those of the authors and not necessarily those of USDA-NASS.

Field Properties Values: A Tool to Identify and Adjust Missing Data from 'Relational' Extraction

Mohammad Mushtaq and April Beaulé, University of Michigan

1. Abstract

The Panel Study of Income Dynamics (PSID) has been using Blaise since 2003 and "ASCII-Relational" data export option to output data into SAS files. During the early stage of Blaise 5 development, the PSID applications development team has used reverse engineering to transform data from "wide" to "relational" format for "All Stars" pilot. Later, PSID staff worked closely with Blaise 5 development team to create tool to export Blaise 5 data into "relational" format, tested and provided feedback to Stats Netherlands during the development phase. The tool is now integrated with Blaise 5 Control Center and being used by the PSID in two surveys.

In a mixed-mode survey, it's important to identify the source and type of missing data values. The web instrument does not allow "Don't Know" and "Refused" whereas such values are allowed in CATI interviews. In web, the values could be missing due to "On-Route" (but not answered by the web respondent) vs. "Off-Route" (not presented to web respondent). Since the data from mixed-mode survey is delivered in a combined Blaise 5 data file, therefore, it's important to harmonize missing data values across both sources. This is done by using Field Properties Values file.

It was observed from pilot studies that Field Properties Values from "wide" are larger than that of "relational" extraction.

In this paper, we examine Field Properties Values from "wide" and "relational" Blaise 5 data extraction, identify missing observation and/or values from "relational" files. Develop methodology to complete the "relational" files where question was on-route but not answered by the respondent of web survey. This distinction is important for data processing team of the PSID and will be used to write explanation for missing values in the public release files. Also feedback to Statistics Netherlands about Blaise 5 "relational" export and possible improvement.

2. Introduction

The Panel Study of Income Dynamics (PSID) is a nationally representative longitudinal study of approximately 9,600 U.S. families. Since 2003, the PSID has used Blaise as its main software for its data collection. Due to the size and complexity of the instrument, PSID staff work with tables extracted in looped blocks or relational tables rather than a single flat file.

As the PSID moves from interviewer administered CATI data collection to self- administered web data collection, one of our significant challenges is to correctly document missing data values. PSID staff worked closely with the Statistics Netherlands team to create and refine the export option to generate relational tables. This extraction tool option is now integrated into the Blaise Control Center and is successfully being used in two PSID Blaise 5 surveys.

As a by-product of the reverse engineering method that used the fps file, we were able to identify and help debug one of our production instruments by observing the difference in size between the fps values from wide versus relational data extraction. This paper will discuss how the fps file was used to map with survey data from relational tables and how we were able to compare the fps file using both methods in order to correct the production instrument early in the production cycle.

3. Background

The PSID is a long running longitudinal study that began in 1968. From 1968-1992 the survey was collected using a paper and pencil instrument. Between 1992 and 1993, the survey transitioned to CAI and was originally programmed in SurveyCraft. The most recent major transition for the survey was when the survey was re-programmed in Blaise for the 2003 wave. For all versions of the survey from 1968-2019 the questionnaire has been interviewer administered. Once converted to CAI, the vast majority of the PSID interviews have been collected via decentralized CATI. Our next transition using Blaise 5 is in many ways, even more significant than the transition from paper to CAI in that our goal is to have a self-administered instrument. The move to a self-administered instrument requires not only significant changes to questionnaire wording and screen design but also to how we record missing data values.

In the PSID we collect information about all family members however, we collect this information from only one respondent. Our household roster has up to 24 loops. Although allowing for 24 individuals may appear excessive on the surface, due to the generational nature of the PSID, this number of roster rows is necessary. The PSID is a survey of related families, once a branch of the family breaks off and moves away, we begin interviewing independent families on their own survey line. However, the fluidity of families are complex and over time some generations of families may drift back together and share the same housing unit again. In these situations we continue to interview these distinct families as separate units; a concept we have termed “Two FU’s in a HU” (Two Family Units in one Housing Unit). In order to capture situations where we have multiple families sharing the same Housing Unit, we have the interviewer list the other family members in related families and continuously remind the respondent throughout the interview to exclude these other family members from their responses to avoid double counting. In some waves of the PSID, we have two, and in rare cases three or more families sharing the same physical dwelling. Due to the nature of the PSID and the relatedness of our families and how they may be grouped, we require the 24 loops to list all of these individuals in the main roster.

The roster itself is then the anchor table that ties individuals to all other sections. We ask about jobs, marriages, children and so forth for each of our family members and each person may have several iterations of each. The design of the instrument requires a series of carefully crafted nested loops. This design feature of the PSID requires us to extract data in a relational format. In a wide extraction, we end up with numerous blank variable positions and an unwieldly amount of columns to handle. Understanding that our sample contains a fluctuating number of family members, and in turn a fluctuating number of their jobs, their children, and their marriages etc. requires that we process relational tables in these content domains.

In previous versions of Blaise 4 and earlier we extracted data in relational blocks using an extraction tool built by University of Michigan programmers based on Manipula. Block extraction has allowed us to handle the data editing, coding and release much more effectively and efficiently. For the PSID to continue its processing methods, relational extraction was not optional but rather, a necessity. Therefore, even though a relational extraction tool wasn’t readily available at the early development stage of Blaise 5, the PSID programming staff was able to output data via the wide-extraction method and use information from that process to reverse engineer the wide table format into a series of smaller relational tables at the block level.

As an added complexity, in the web self-interviewing mode we also need to capture precisely the values of missing data. In the self-administered instrument, almost all fields are optional, meaning that they do not require a response from the respondent. Unlike CATI instruments where special answers like “Don’t Know” or “Refused” can be invoked on almost any question, the web versions do not have special

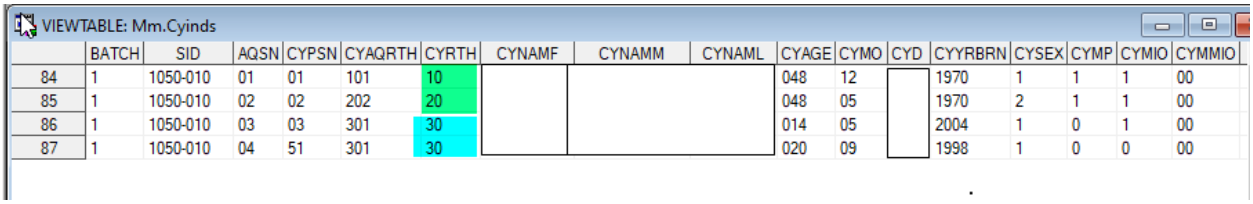
answers available to the respondent. In CATI interviewer mode each question on route requires a response including special answers as necessary. In contrast, the web self-administered mode allows an empty response on almost all screens and special answer options are not provided.

The prevailing reasoning given for the avoidance special answers for the respondent in a self-administration mode, is the belief that if they appear onscreen, the respondent may be more tempted to use them more readily in lieu of providing a valid response option. Given this significant difference between modes, the fps file becomes even more critical in determining the precise value of the missing information for each variable. For documentation purposes, we need to know if this question was on-route and simply not answered or whether it was never on-route. Knowing the difference between these two types of missing values is imperative for documentation and release.

4. Need for Relational Extraction for Coding/Processing and Release

The processing of the PSID takes many months using a team of eight experienced editors. Families are organized on the roster in order to prioritize the focal persons in the survey – they are the Reference Person and their Spouse/Partner. There is a set of rules for determining those two prominent people *(Figure A: CYRTH=10 and 20)* and many more questions are asked of these two individuals than OFUMs (Other Family Unit Members*) (Figure A: CYRTH = 30).* As editors comb through all the interviews in a related clan, they must determine the correct configuration of each family. During field work some individuals are listed more than once in different interviews and editors must decide where each person is located in each family. In order to maintain data integrity, a person may only appear once (in one single family) in any given wave. Due to the nature of this editing task, the PSID staff must manage many edits to the roster- moving people to different positions, adding or deleting people and so forth. In a long format, these types of transformations are straightforward and easy where rows are inserted, deleted or key values for individuals are updated as necessary. In a wide format this editing task would be extremely challenging and cumbersome as columns would need inserting, deletion and as individuals are shifted around positions, other individuals would need their information re-packed into existing columns.

Figure A: Example Roster PSID-Long Format


VIEWTABLE: Mm.Cyinds

	BATCH	SID	AQSN	CYPSN	CYAQRTH	CYRTH	CYNAMF	CYNAMM	CYNAML	CYAGE	CYMO	CYD	CYYRBRN	CYSEX	CYMP	CYMIO	CYMMIO
84	1	1050-010	01	01	101	10				048	12		1970	1	1	1	00
85	1	1050-010	02	02	202	20				048	05		1970	2	1	1	00
86	1	1050-010	03	03	301	30				014	05		2004	1	0	1	00
87	1	1050-010	04	51	301	30				020	09		1998	1	0	0	00

In order to easily manipulate individuals, their jobs, marriages, children and other domains, our editing system consists of a series of related tables with a set of primary keys. Our editors are able to easily insert, delete or reconfigure keys for tables organized in content areas.

5. The Filed Property Values (FPS) Files

The PSID Mixed Mode Pilot (MMP) data collection ended by the end of 2019, and we are using the data files from this survey to compare FPS files with survey data values.

The FPS file is cell level indicator whether a survey item (field question) is presented (on-route) to the respondents for the entire sample and questions in the survey. The file also includes remarks (F2 notes)

data, therefore, it's important to determine whether the file is complete regardless of the method of data extraction. To our surprise, the FPS file from "wide" extraction has 55,638 rows and the "relational" extraction has 50,993 rows. There were 179 "remarks" from "wide" and 174 from "relational" data extraction methods. Further checking of remarks data showed that one sampleid which was in "wide" file, it was missing from the "relational" file – something lost in transition. Since FPS file from "wide" data extraction has relatively more information on visited fields, therefore, the "wide" file was used to identify: a) any loss of data from "relational" data extraction, and b) to add or update data values of relational tables.

6. The Mapping of Survey Data and FPS Data Files

In order to compare FPS file with survey data, the first step is transform both datasets in way that they are linkable to each other and then compare "IsVisited" indictor with survey responses -- at cell level.

a) The Survey Data: From MMP survey, there are 139 tables and 6,279 variables from relational data extraction. Not all tables have data and keys variables are excluded from cell level transformation. So, there are 106 tables and 5,176 variables are used to create a "long" file. The cell level "long" data file has 1,187,455 rows where each data value is uniquely identified by set of four variables from the entire survey data collection. Below is the structure of "long" data file:

#	Variable	Type	Length	
1	PrimaryKey	Char	**7**	--+
2	Tablename	Char	**32**	| Key
3	TableInstance	Char	**75**	| Variables
4	Variablename	Char	**32**	--+
5	DataValue	Char	**40**	

During long transformation, all data values have been converted to character data type. Open text values were truncated to 40 characters. The value is used as flag to measure survey response for an item presented to the respondent.

b) The FPS Data: As explained before, we are going to use FPS from "wide" extraction in this linking. Below is the layout of raw FPS file:

#	Variable	Type	Length
1	PrimaryKey	Char	**7**
2	Path	Char	**100**
3	Property	Char	**10**
4	Value	Char	**200**

First, the data needs a lot of programming to create variable and instance level information from "path" column. The data in this field is delimited by a dot (".") and the last value is variable name. The FPS file is at .bmix level, therefore, multi mention (set of) variables need to be converted to match with survey data variables.

Second, create table instance (proxy for variable instance) from path column after removing the variable name from the text value. This is done by look at the "table instance" values of long file from Survey Data file created in step a) above.

Third, adjust array variables for an array range suffix and "set of" variables to match with variable name in the "long" data file. Then merge table and variable id from metadata created from .blax files and make

other adjustments manually as needed. The variable instance found in FPS file only may have missing values of key variables and missing values must be assigned valid values before merging FPS and survey data files. In the final merge, keep all rows from FPS file only, i.e., a left join on FPS file.

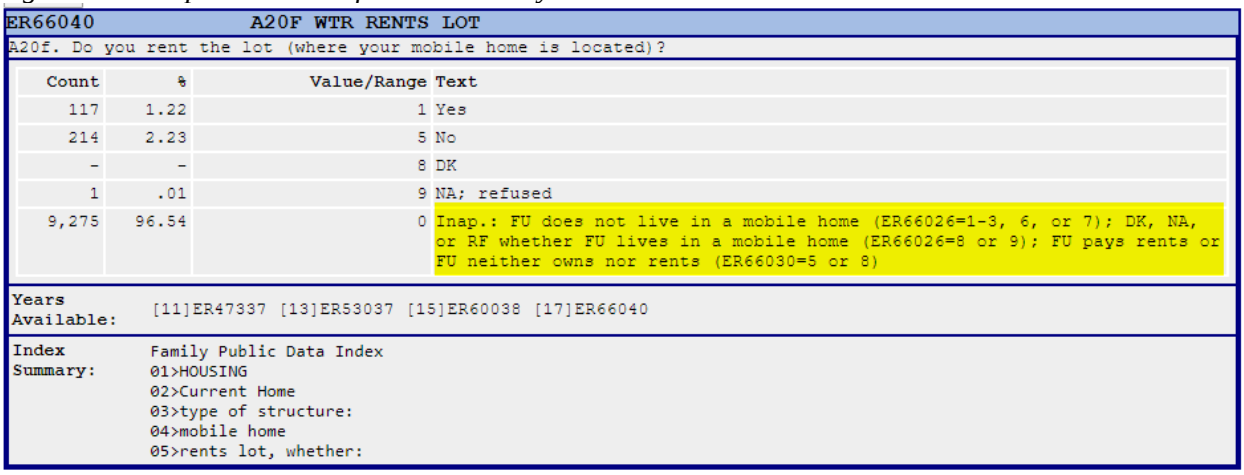
c) The Evidence: A total of 74,536 data out fields are flagged as visited by the FPS file, which is about 6.28% of all data cells in the survey data. Out of which: i) 55,458 have values in the data files, ii) 18,887 cells have indication of on-route from FPS file but no data in the data files, and iii) 191 new cells be added to the survey data files.

d) Alternative evidence: The FPS file is based on .bmix where "set of" variables have one row per sampleid-variable instance. A better approach is to rollup the "long" data file to match with variables from FPS file such that "set of" variables are counted once. The count of on-route/missing values is 4,481 compared to 18,887 reported in c-ii) above.

7. Use of FPS in missing data values: 'on-route' vs 'off-route'

As with all survey data, the ultimate goal of the study is clean, coherent data with fully documented variables. The PSID study has a long standing tradition of providing complete documentation and a description of the valid, missing and INAP (Inappropriate code) values. INAP values are assigned to items which were skipped or 'off route' for this particular person or family. The text description of the INAP values is the inverse of the universe *(Figure B).*

Figure B: Example Codebook for PSID Family Level Variable in CATI



ER66040 A20F WTR RENTS LOT

A20f. Do you rent the lot (where your mobile home is located)?

Count	%	Value/Range	Text
117	1.22	1	Yes
214	2.23	5	No
-	-	8	DK
1	.01	9	NA; refused
9,275	96.54	0	Inap.: FU does not live in a mobile home (ER66026=1-3, 6, or 7); DK, NA, or RF whether FU lives in a mobile home (ER66026=8 or 9); FU pays rents or FU neither owns nor rents (ER66030=5 or 8)

Years Available: [11]ER47337 [13]ER53037 [15]ER60038 [17]ER66040

Index Summary:
Family Public Data Index
01>HOUSING
02>Current Home
03>type of structure:
04>mobile home
05>rents lot, whether:

In CATI we capture the difference between missing (Don't Know or NA; Refused special answers) and INAP (system missing) because the variable is 'off-route'. Since all variables in the interviewer administered mode require a response, it is always clear which variables are on or off route. For self-administration mode where empty is allowed, this becomes increasingly difficult to determine the system missing values that are on-route or off route. In order to determine this critical difference, we turn to the fps file to help us make those assignments.

A three step approach is used to keep distinction between the data values which are on-route and missing vs. standard missing data.

1. In step one, data from Blaise are extracted with missing data codes (.D=Don't Know, .R=Refused, .A=Special Answer (997, 9997,…), .B=Special Answer (996, 9996, …), … more special answer codes. This step is labeled as "Blaise Data As Is".

2. In step two, another missing data code .V=Visited is used to indicate on-route and missing. If FPS to Survey Data mapping shows that a filed is visited but the data cell is empty then the cell will be updated to .V as special missing data code. If an entire is missing then a new row will be created with .V values appropriately. In this step all missing data values are represented by .D, R, .V, and all Special Answers (.A, .B, .C, .F, .G, .H, etc.).

3. In third and final step, the Don't Know and Refused missing data values are flipped to ISR standard DK/RF numeric values. The on-route missing (.V) are also converted to RF equivalent numeric values. The Special Answers missing data values are converted to their respective numeric equivalent. These files are saved in "Data Out" folder as final set of data files for further use of data with other data processing systems. The data frequencies are also calculated and uploaded to Oracle database for the use of in house applications.

With a three step approach, the process is able to keep backward linkage with Blaise data extracted in the first place (aka Blaise as is) and can used to write INAP data value explanation for the public release data file.

8. Conclusion and Summary

With Blaise 5 and mixed mode survey instrument, an identification on-route/missing data values is important for data processing and release of the PSID data. Regardless of the two approaches to identify on-route/missing data values in section 5 c) and d) above, the FPS file from "wide" and survey data "relational" extraction should be used to account for data leakage.

As described in section 5, FPS file from "relational" extraction has less data than the FPS file from "wide" extraction. The issue should be further investigated in collaboration with the PSID staff and the Blaise development team at the Statistics Netherlands.

Using Respondent Centred Design to Transform Social Surveys at the ONS

Alex Buckley, Office for National Statistics

1. Abstract

The UK government has a Digital by Default strategy which means that by 2020 digital self-service is the default option for people who can use it, not the only option. Coupled with public expectations to be able to do surveys online and increasing use of smartphones to perform online tasks means the Office for National Statistics (ONS) has invested in a transformation programme that focuses on research to deliver and integrate respondent centered online data collection. We have taken a 'blank page' approach to the redesign of our respondent journey which has been met with success.

This talk will share our approach to achieving this and the principles that the social survey research and design team use to develop online-first mixed-mode surveys. ONS practices respondent centrism in its approach to design; we place the respondent at the forefront of our design process by aligning the questions and flows with their mental models. We create a questionnaire that collects accurate data which meets the requirements whilst also delivering a positive respondent experience which is vital for voluntary longitudinal surveys.

This talk will provide tangible examples of changes to questions, including tips for other researchers to take away and apply. It will also discuss novel techniques such as combined cognitive and usability testing and why it is essential for successful delivery and good data quality in self-complete modes.

We optimise for the mode and design 'smartphone first'. This talk will demonstrate through evidence how and why this approach is critical for success. By optimizing for smartphones, we create cleaner designs, ensure higher quality data and reduce the break-off rate for respondents who are unlikely to return on another device or in another mode.

We will share our practical examples and recommendations on techniques, questions and approaches.

2. Social Survey Transformation

Under the umbrella of the Census and Data Collection Transformation Programme at the Office for National Statistics, we are working to evolve our Social Surveys so that they are relevant, efficient and suitable for the digital age. This transformative programme has come about as a result of the UK government's Digital by Default strategy. Digital by Default was introduced in 2012 and means making online services so easy to use that they become the default mode to access them in, which is reflective of an increasingly digital society (but more about that later). For us in the Research & Design team, this means putting users first – not data users, but the hundreds of thousands of members of the public who voluntarily complete our surveys every year.

3. User Centred Design

Being respondent centric involves putting respondents at the top of your list of considerations when designing – all the way from the wording you use for questions and responses, to the order that the questions are asked in. This approach to survey design and data collection is derived from the philosophy

of user centred design, in which the needs of the service user are given the utmost attention at each stage of the design process.

User-centredness in government service design is a radical approach, first adopted and promoted by the Government Digital Service (GDS), which was set up to aid the Digital by Default strategy. Alongside pioneering national digital infrastructure (GOV.UK)[10], providing government departments with technology that reduces barriers and promoting better data in government[11], the GDS has developed a set of design principles which inform our approach to survey redesign. They are:

- Start with user needs
- Do less
- Design with data
- Do the hard work to make it simple
- Iterate, then iterate again
- This is for everyone
- Understand context
- Build digital services, not websites
- Be consistent, not uniform
- Make things open: it makes it better

The threads of the GDS principles run through every aspect of our work. They influence our approach to survey design by demanding that we talk to and interact with our survey users before making decisions about something that will impact their experience. This is the only way to make sure the work we do is respondent centric. In the past, this approach has been met with resistance, but we persevere and promote this way of working because we know it produces better results.

4. Discovery and alpha

So what does this innovative approach look like on the ground, in practice? We adopt the agile service delivery approach set out by the GOV.UK Service Manual[12] which involves designing and testing services in an iterative manner. The first two of these phases are known as discovery and alpha.

Discovery does what it says on the tin. Before we can begin to create a design solution for any of the surveys in our scope, we must first understand our survey users and their contexts. First and foremost, we GOOB (get out of the building). We use a variety of qualitative research methods to gather data on our survey users. Discovery often includes focus groups and pop-up testing, methods which we consider suitable for collecting nuanced and insightful data. Focus groups, conducted with members of the public that share a characteristic in common with a set of survey questions that we are transforming, allow us to uncover people's understandings of topics and their mental models surrounding them. For example, last year I worked on the transformation of a set of questions which gathered information from self-employed individuals, including their gross and net incomes. We held a focus group with a range of self-employed people to find out how they thought about their income outside of the traditional monthly payslip that we as employees receive. We do this to make sure we understand context. After analysis, we combine our findings with our data user requirements and mock up some high level prototypes of survey questions.

[10] https://www.gov.uk/
[11] https://data.blog.gov.uk/2015/09/24/work-of-prog/
[12] https://www.gov.uk/service-manual/agile-delivery

We put these through pop-up testing, either with the public or sometimes in the coffee-shop at work, luring colleagues in with free chocolate. This is a 'quick and dirty' way of gathering thoughts and opinions on early stage prototypes, allowing us to find out early on if something we've put together is likely to fall down when released to the public. Approaching design like this in the early stages means we fail fast and fail forward.

The next stage is to take our first set of redesigned questions to test with our users. This is where we find out what's working and what isn't and is known as the alpha phase. We use a unique method called 'cogability' to determine whether or not our redesigns are useable and collecting the information that we intended them to. Cogability is a combination of cognitive testing and usability testing, and involves observing the participant going through a simulation of the survey. We then retrospectively probe the participant about their experience, to dig deeper into what they found easy, what they struggled with, what frustrated them, what made them answer in the way they did, and, most importantly, why.
We analyse the data from these sessions and develop findings - what works about the suite of questions and can stay the same, and what fell down and needs to change. Then we make those changes. And then we go out and test. And then we analyse. And then we change. And then, if needs be, we do it all again – in GDS terms, we iterate. Then iterate again. We do this because we want to design with data and evidence, rather than our own presuppositions and presumptions. Ultimately, we do it to make sure the surveys that we design are for everyone. That means they need to be accessible, inclusive and usable.

5. Inclusive design, usability and accessibility

Usability, accessibility and inclusion are key elements of making sure a design or service works for everyone. They are three separate aspects but there is a great deal of crossover between them, and often one is mistaken for another. It's important to consider them all individually, as each one will have a different impact when it comes to questionnaire design. The accessibility of your survey or service means it can be used by as many people as possible, including those with impaired vision, motor difficulties, cognitive impairments and deafness or impaired hearing. At least 1 in 5 people in the UK have a long term illness or disability, and many more a temporary impairment, so failing to ensure that our designs can be used either as they are or with adaptive software, would mean excluding a large proportion of the population from engaging with our surveys. We work closely with the Digital Accessibility Centre (DAC), a specialised team who ensure everything we output at the ONS meets Web Content Accessibility Guidelines, alongside in-house measures

'Accessible design' is often used interchangeably with 'usability design', and it's easy to see why - there are overlaps, but also some key differences. Usability is about making products and services effective, satisfying and as the name might suggest, usable. It includes dimensions such as a familiar user interface on digital services and the extent to which a user is actually able to complete the tasks they set out to in the first place. Usability is crucial in design but does not consider aspects of the experience which disproportionally impacts people with disabilities and impairments, and this is where the difference lies.

Finally, we have inclusion. This is about design for all, and encompasses a range of different aspects such as accessibility, language, computer literacy and skills, geographic location, education and personal context. We take measures to ensure that our survey designs are inclusive by conducting readability tests, which aim to make sure the text involved in our surveys is around a reading age of 9, which is the average UK reading age. We also make sure our surveys are available in other modes for those who have limited digital abilities and on the flip side, make sure our designs work on mobile, because we know that people use their mobiles to do things like filling in surveys.

Making services accessible, usable and inclusive means designing for all. The user benefits because whoever they are, they can do what they set out to do. Service owners benefit because the users have a better experience and are more likely to come back to – or keep using- that service, whatever it may be. And the latter is definitely the case when it comes to surveys!

6. Optimode design

One of the key ways that we make sure we are being respondent centric is by taking an 'optimode' approach. Optimising for the mode means designing surveys which are tailor-made to the mode that they are going to be deployed on, rather than 'lifting and shifting' content which was previously designed for CATI to CAWI, which was the approach taken traditionally in similar projects and contexts. This generally means completely scrapping a suite of questions and starting from scratch, which may seem extreme, but actually makes a lot of sense when you look at it from a respondent-centric position. As discussed in previous sections, doing the hard work at this stage means our surveys function better in the long run. The better the surveys are designed, the less burdensome they are for the respondent, therefore the data they produce is better because respondents are less likely to drop out halfway or give us incorrect information because they haven't understood what we're asking them.

In the Research & Design team, we take an online – first approach. Designing for the web in the first instance forces us to think hard about our patterns and questions so that they are as efficient as possible. Doing the hard work at this stage to reduce the complexity of the questions so that they can be answered by the respondent on their mobiles, without guidance, also means that in most cases, the designs work really well in other modes, with little need for moderation. In principle, optimode means designing separate versions of our variables for CAWI, CATI and CAPI, but we find that in reality, this isn't always necessary. It turns out that if you optimise the design for the most challenging mode (which is CAWI because there is no expert interviewer on hand to explain to the respondent what information a question is *actually* looking to collect), often the design is easily translated to other modes with little need for tweaking. Of course, this doesn't mean that this is always the case. Whilst we are moving to an online-first, mixed mode approach, we would also always offer our surveys in various modes for people who are not able to or do not wish to complete them online. Therefore we treat our telephone and F2F mode designs the same way as we treat online designs when it comes to testing. The designs are tested scrupulously in the same manner, with the general public but also with telephone and field interviewers, and changed until they work as efficiently as possible either over the phone or out in the field with our interviewers.

7. Adaptive design and mobile first

Further to designing for online in the first instance, we also take a mobile-first design approach. This is a contentious way of working, because there are lots of people of the opinion that designing in this way is harder or simply not necessary. Let's look at this as two separate points. First: 'designing for mobiles is not necessary'. In 2011, according to data from Ofcom, 2% of adults in the UK only used their smartphones to go online. In 2018, this was 11%. To break this down further, of 16 – 24 year olds, 12% only used a smartphone to get online. Of 25 – 34 year olds, the figure was 22%. By ignoring mobiles as a device that people are likely to use to complete surveys, not only are we potentially excluding a staggering amount of already hard to reach respondents, we are also excluding those from certain socioeconomic groups. Of those in socioeconomic group DE (defined by NRS as semi and unskilled manual workers, state pensioners, casual and lowest grade workers and unemployed with state benefits), 17% are only going to be accessing the internet on their mobiles. We can see from these figures that designing for mobile is crucial to make sure surveys are reaching younger respondents and people from

all socioeconomic backgrounds, meaning that crucial groups of people are not underrepresented and can have their say. Actual mobile use when it comes to survey participation is reflected in our recent Labour Market Survey Statistical Test. Of those who completed online, nearly 18% did so on mobile. If that section of the sample had not been given the option to complete on mobile, due to the availability of devices or busy lifestyles, it's possible that they would not have responded at all. This demonstrates the importance of providing an optimised mobile version of our surveys.

Second: 'designing for mobiles is harder and takes longer'. Designing for mobile first doesn't need to be harder. Whilst approaching survey design in this way might force us to think about reducing the complexity of the front-end, it makes life easier in the long-run because these types of design are easier to test and crucially, are generally easier and more effective patterns to use on other devices. It's much harder to design for desktop first and then try to squeeze that design on to a tablet or mobile than it is to design using restricted space, creating a pattern which looks great on mobile but also on a tablet or a desktop. Designing mobile first makes us really think about what we're including on a page, so there is no unnecessary fluff in the question, response or guidance (if guidance is really needed, which in most cases it isn't if the question has been phrased effectively).

Finally, designing for mobile first makes the patterns we produce more effective for the adaptive design approach that our in-house survey data collection team work with. They work in this way to ensure that the respondent's experience is optimised, whichever device they choose to use to fill in our surveys. Ultimately, we are all working towards the same thing: the best experience for the survey user, which means better data for the data user.

The Michigan Questionnaire Document System (MQDS) For Blaise 5

Gina-Qian Cheung, Cheng Zouh, Kelly Chatain, and Sarah E Broumand
University of Michigan Survey Research Center, United States

1. Abstract

The Michigan Questionnaire Documentation System (MQDS) is an application that extracts metadata and data from Blaise instruments/databases to generate various types of output using the Blaise API. With the advent of Blaise 5, upgrades to MQDS were required to integrate with the new structure and API. MQDS outputs include a data dictionary, questionnaire documentation, and codebooks. MQDS for Blaise 5 has been simplified from previous versions, specifically no longer using the Data Documentation Initiative (DDI) standard as the core output from which other transformations are generated. The Blaise-to-SAS process has also been discontinued as Blaise 5 now includes this functionality. The data dictionary documents the information of all defined fields, like field types (DataField vs. AuxField), route status (on-route vs. off-route), data structure types (integer, string, enumeration, set, etc.), question and description texts, enumerations and special answers. The results are saved to either SQL server tables or csv files. The questionnaire documentation function includes all possible on-route fields by mode and/or language and can be saved in HTML or RTF format. The codebook function generates summary statistics and frequencies of survey data for all fields. Additional functionality for the questionnaire documentation and codebook output, such as routing logic and universe statements, is in development.

2. Background

Survey Research Operations (SRO), a unit within the University of Michigan's Institute for Social Research's Survey Research Center, developed MQDS in 2003 to allow users to export Blaise metadata and data for documentation and dissemination of Blaise questionnaires to their users (Sparks and Liu 2004, Guyer and Cheung 2007). The first version was referred to as "BlaiseDoc" (Sparks and Liu, 2004). Further development produced MQDS versions 2 and 2.5, which involved rewriting the program in .NET and including additional utilities (Guyer and Cheung, 2007). The creation of version 3 began in 2009 and was outlined in Dinkelmann et al (2009). Version 3 was database driven and written to use the DDI 3.0 standard. In 2011, MQDS 4 was enhanced to more efficiently process large data models and use the DDI 3.1 standard. All versions of MQDS 4 and earlier used either Blaise version 4.6, 4.7, or 4.8.

3. MQDS for Blaise 5

The primary goal of MQDS for Blaise 5 is to extract Blaise information for testing instruments, reviewing questionnaires, preparing documentation, and comparing questionnaires across data models or across studies.

Blaise 5 brings forth many major changes in what survey developers are able to accomplish. It also has a completely new implementation of the Application Programming Interface (API). This is important, as all tools that currently exist around Blaise 4.8 will need to be updated to take into account the new structure of the API. Therefore, in order to access the metadata within Blaise 5, MQDS updates were required.

MQDS for Blaise 5 has been simplified from previous versions. It was developed to support the following tasks:

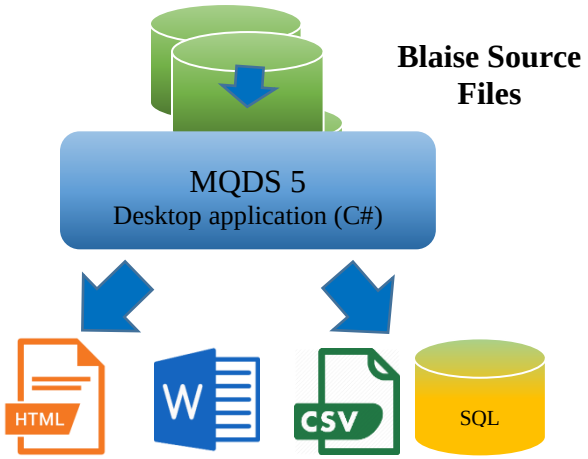
- ➢ Analyses of the data model and its associated files via the import of the Blaise metadata and data to a SQL database, SQLLite or a CSV file
- ➢ Outputs the questionnaire in multiple file formats
- ➢ Provides summary statistics and universe logic for each variable

MQDS for Blaise 5 no longer supports:

- ➢ Generating DDI metadata
- ➢ Blaise-to-SAS functionality (now provided by Blaise 5)

Previous versions of MQDS were tightly coupled to the DDI standard, requiring a DDI instance to be created from which all other output was generated. As new versions of the standard were released, migration from the previous versions and significant redevelopment was required. This led to limitations in the database design used in MQDS 3 and 4. In creating MQDS 5, it was decided to no longer use DDI as the core, but instead may provide DDI instances as an additional output.

HIGH LEVEL SYSTEM ARCHITECTURE



HIGH-LEVEL CAPABILITIES

- Database output
- All output will be exportable
- Help documentation for every function

ACCESSIBILITY

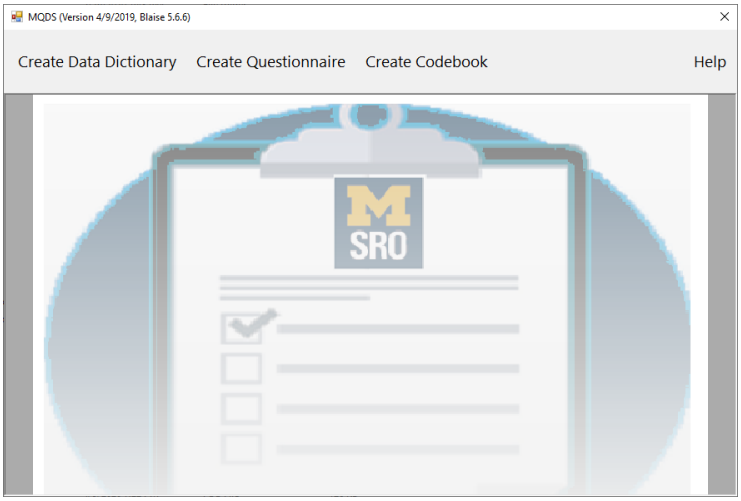
- Desktop application
- No login required
- Minimal configuration required to install

4. Functional Capabilities

MQDS for Blaise 5 has three main functions:

1. Create a data dictionary
2. Create a questionnaire
3. Create a Codebook

Figure 1: Main Menu Display



4.1 Create Data Dictionary

The data dictionary documents information for all defined fields, like field type (DataField vs. AuxField), route status (on-route vs. off-route), data structure types (integer, string, enumeration, set, etc.), question and description texts, enumerations and special answers. The results are saved to either SQL server tables or csv files. Such a data dictionary can be used as a lookup table for survey data delivery or quality check of instrument design (e.g., avoiding accidentally defining important on-route fields as AuxField).

The new interface prompts the user to follow a sequence of steps:

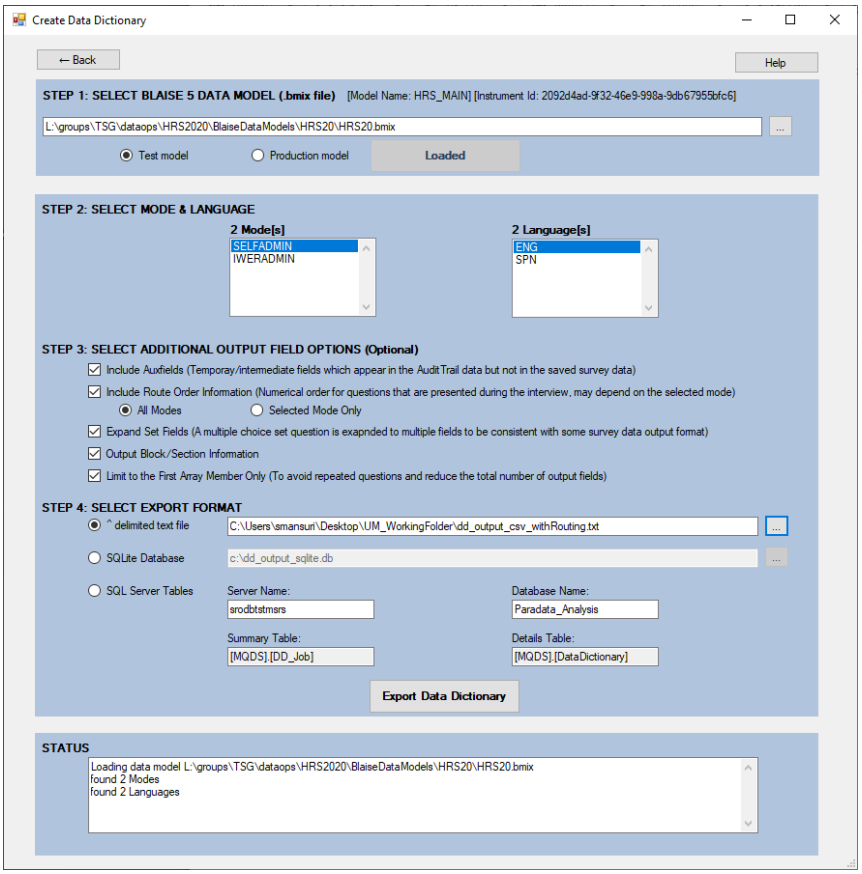
- **Step 1**: Select Blaise 5 Data Model (.bmix file)
- **Step 2**: Select Mode & Language
    - These fields get automatically populated based on the content on the .bmix file. The user then has a choice to select a mode or language preference to export
- **Step 3**: Select Additional Output Field Options (Optional)
    - These are all optional items to select from and include the following:
        - Include Aux Fields: Aux fields are variables computed or programmed in Blaise and used in checkpoints, routing instructions, and question text, but are not part of the merged data set. For example, "age" can be an aux field that is computed from the Date of Birth.

- Include Route Order Information: Route order is a numeric value associated with each variable that indicates the order in which each question is asked/presented in Blaise. It does not indicate routing logic or whether or not the question was asked.
- Expand Set Fields: A Blaise question set might include response options for a 'select all that apply' question about the race of the respondent. If there are seven options to choose from, the expanded set will include variables "race[1]" through "race[7]" and indicate if the variable is part of a set.
- Output Block/Section Information: Will provide the block name or section name for each variable.
- Limit to the first array member only: A looped question (array) will produce virtually identical variables up to the number of loops defined in the instrument.

- **Step 4**: Select Export Format
- **Step 5**: Export Data Dictionary

See Figure 2 below for an example of the user interface.

Figure 2: Sample Data Dictionary Form



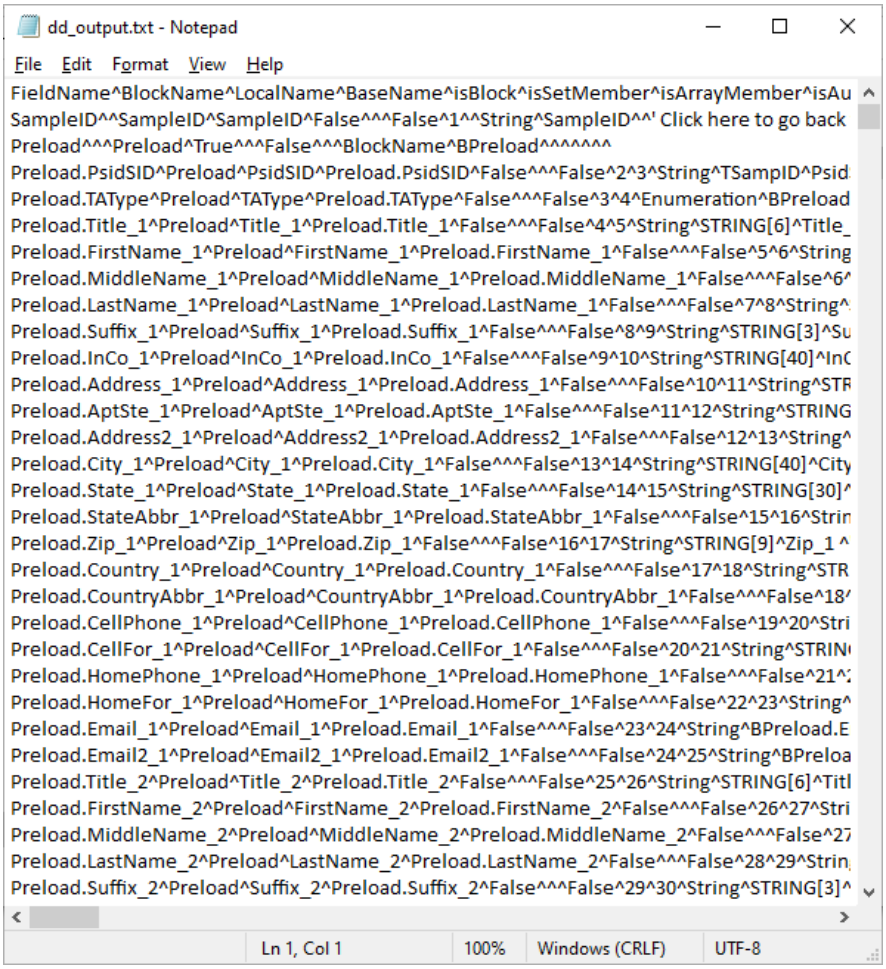
Data Export Options

There are 3 options for exporting data:

1. (^) Delimited text file
2. SQL Lite Database
3. SQL Server Tables

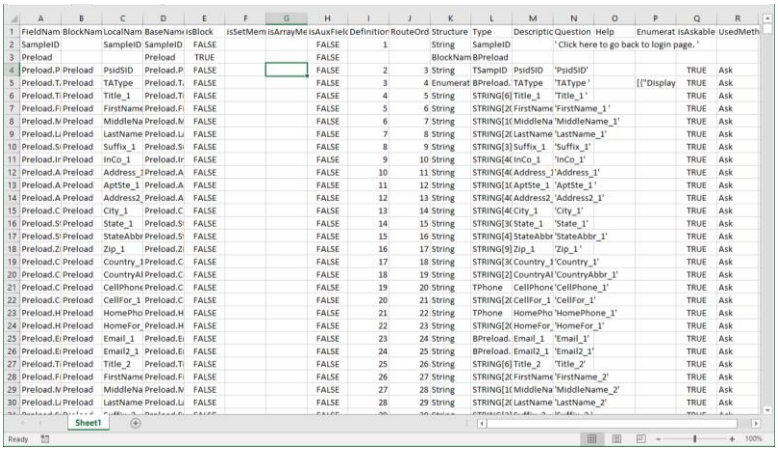
The user can select the option to export to a (^) demiled text file. When the "Export Data dictionary" button is clicked, the file will be saved in the same directory as the MQDS.exe file.

Figure 3: Sample Data Dictionary TXT Output



Once the file is downloaded, the user can import the delimited (^) file into excel and view all the data in a table format.

Figure 4: Sample Data Dictionary imported into Excel or CSV



Below is a summary of all of the available output fields using MQDS for Blaise 5.

Table 1: Sample Data Dictionary Output

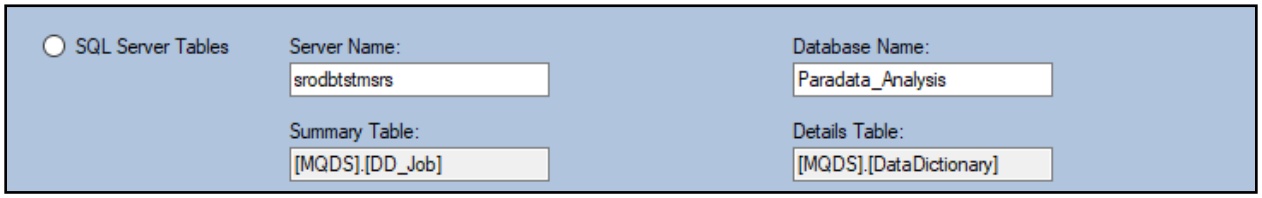
Column Name	Description
FieldName	The Blaise field name with full path,e.g. SectionA.Family.Housband.Name
BlockName	Block name is full path name without the last local name.E.g. SectionA.Family.Housband from the above case
LocalName	The last part of the full path field name, e.g. Name from the above example
BaseName	For set question without index
isBlock	1: it is a question name; 0: it is block/section name
isSetMember	1: expanded set member;0: base name of a set question;
isArrayMember	1: array member; 0/empty: not a array member
isAuxField	1: AuxField; 0/empty: Data field
DefinitionOrder	Numeric order of each field defined in the data model
Column Name	**Description**
RouteOrder	Numeric order of each field presented during a interview
Structure	The structure of a field, e.g. Date,Enumeration,Integer,Real,Set,String,Time

Type	Blaise defined data type
Description	Description of a question
Question	Question text of a question
Help	Help text of a question
Enumeration	Detailed enumerations/choices for enumeration & set questions
isAskable	1: an route order field is askable; 0: not askable
UsedMethods	How is a route order filed used, e.g. Ask, Ask&Show, Show

Exporting to SQL Lite is straightforward (assuming SQL Lite has been downloaded to the users machine) because the database instance and tables are generated automatically.

MQDS for Blaise 5 can also export data to a SQL server. The installation of the application will include default settings for the server and database. See Figure 5 for default configuration parameters.

Figure 5: SQL Server Parameters



Users are able to change these configuration settings by editing the MQDS.exe.config file and specifying the new server and database name.

Figure 6: Example MQDS.exe.config File

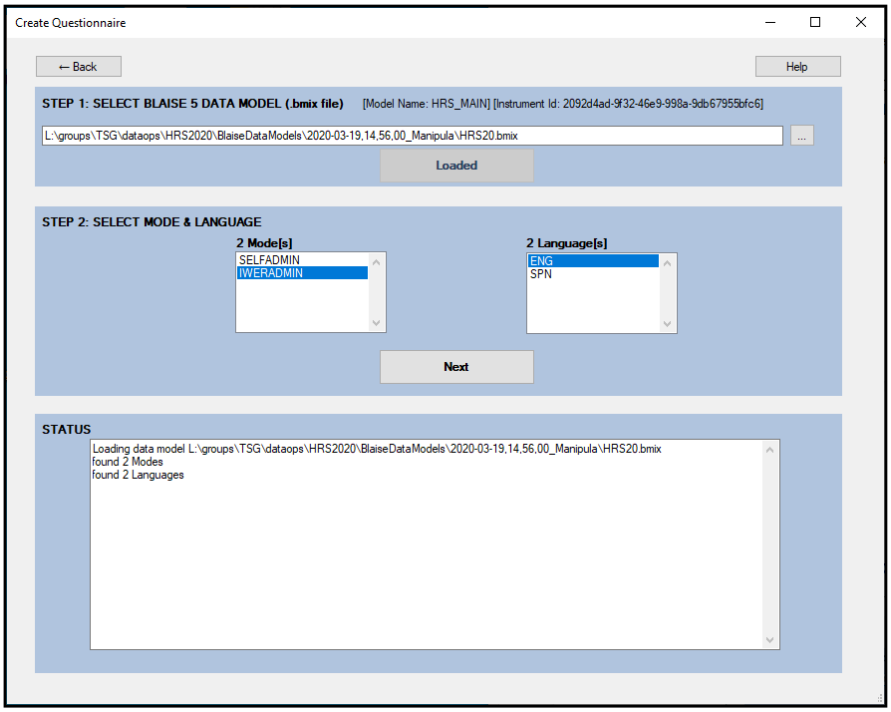
```
MQDS.exe.config
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
    Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
    <sectionGroup name="userSettings" type="System.Configuration.UserSettingsGroup, System, Version=4.0.0.0, Culture=neutral,
    PublicKeyToken=b77a5c561934e089" >
      <section name="MQDS2019.Properties.Settings" type="System.Configuration.ClientSettingsSection, System, Version=4.0.0.0,
      Culture=neutral, PublicKeyToken=b77a5c561934e089" allowExeDefinition="MachineToLocalUser" requirePermission="false" />
    </sectionGroup>
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <connectionStrings>
    <add name="Paradata_AnalysisEntities" connectionString="metadata=
    res://*/Model1.csdl|res://*/Model1.ssdl|res://*/Model1.msl;provider=System.Data.SqlClient;provider connection string=&quot;data
    source=srodbtstmsrs;initial catalog=Paradata_Analysis;integrated
    security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />
  </connectionStrings>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
      <parameters>
        <parameter value="mssqllocaldb" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices,
      EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
  <userSettings>
    <MQDS2019.Properties.Settings>
      <setting name="checkBox_outputAuxfield" serializeAs="String">
        <value>0</value>
      </setting>
      <setting name="checkBox_AddRouteOrderInfo" serializeAs="String">
        <value>0</value>
      </setting>
      <setting name="checkBox_expandSet" serializeAs="String">
```

4.2 Create Questionnaire

MQDS for Blaise 5 creates documentation of questionnaires that include all possible on-route fields by mode and/or language. Field names, labels, question text, response options, as well as interviewer instructions are currently available for output. Output options allow the user to save the questionnaire in either HTML or RTF format. Additional functionality is in development.
The new interface prompts the user to follow a sequence of steps.

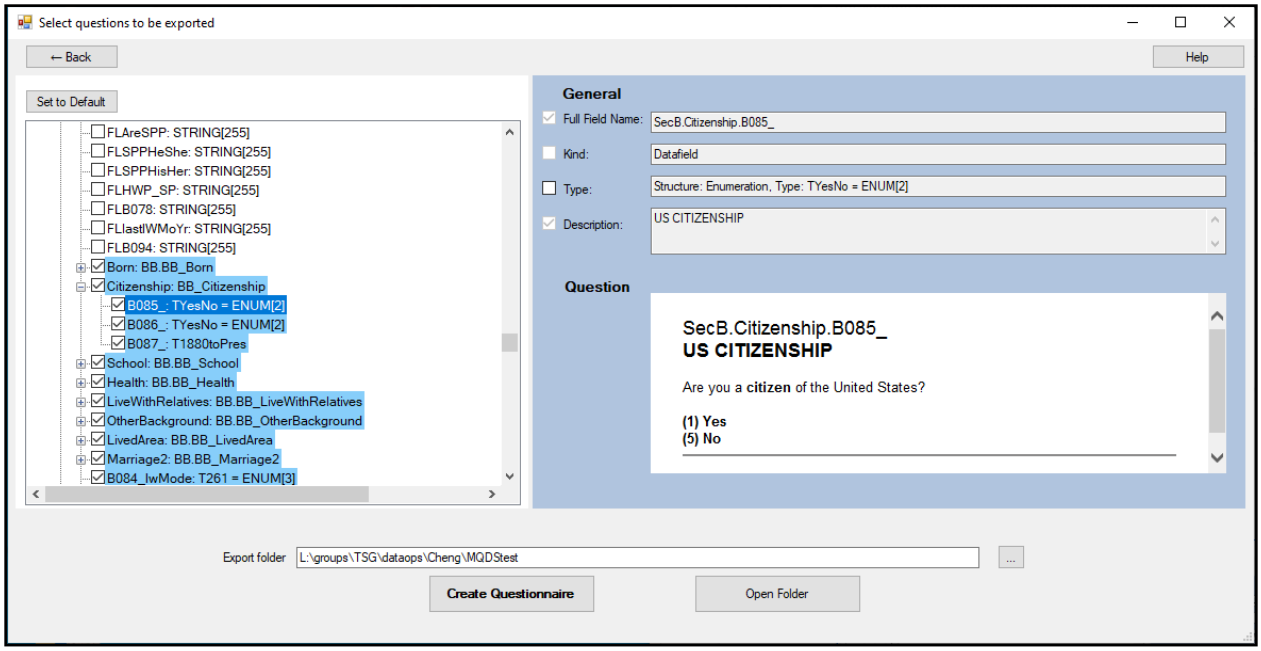
- **Step 1**: Select Blaise 5 Data Model (.bmix file)
- **Step 2**: Select Mode & Language
    - These fields get automatically populated based on the content on the .bmix file. The user then has a choice to select a mode or language preference to export

Figure 7: Create Questionnaire Input Form



Once the data model is uploaded, the user will be presented with the following screen:

Figure 8: Sample Select Questions to be exported

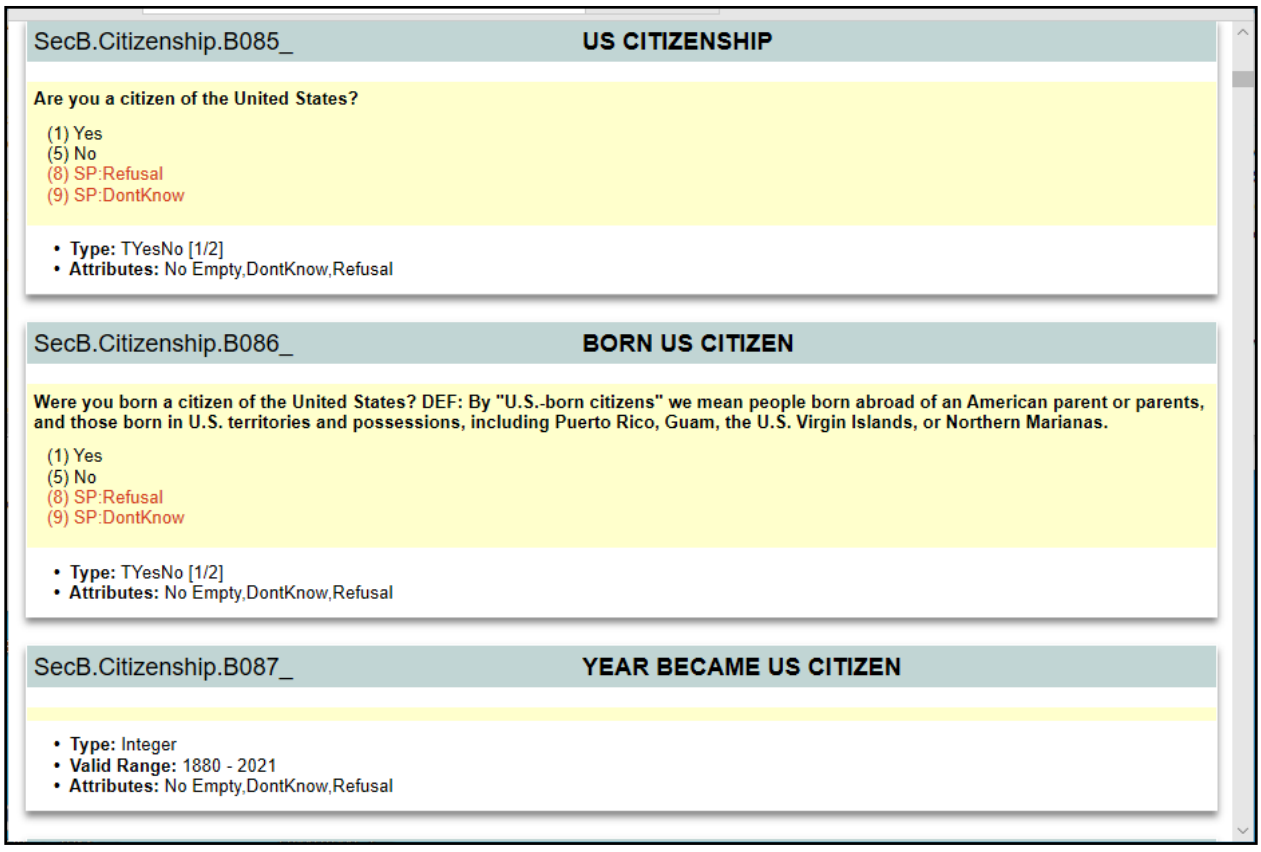


The Select Questions Form is divided into 2 panes, the left pane lists all the questions in the survey. The questions highlighted in light blue are the “on route” questions. These are questions that are displayed to the interviewer. The right hand side displays information about the question selected on the left pane. This includes:

- Full Field Name: The Blaise variable/field name
- Kind: The type of field (Datafield, Auxfield, Block, etc)
- Type: field data type
- Description
- Question

The user is able to select one or more questions from the left pane, select the Export Folder and then Create the questionnaire by clicking on the **Create Questionnaire** button.
Below is an example of a current generated Questionnaire.

Figure 9: Sample Questionnaire Output



To reduce the questionnaire size, the user is able to choose to limit array questions to the first member-only and avoid outputting repetitive question text.

Future functionality:

- Include routing logic and universe statements for each variable.
- Additional output formats

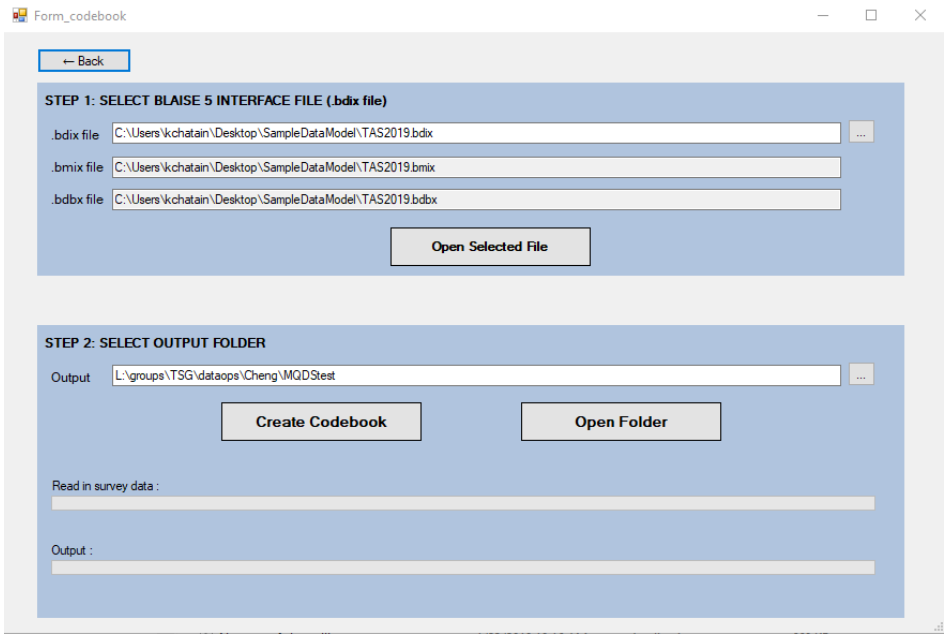
4.3 Create Codebook

MQDS is also used to create the summary statistics and frequencies of a survey instrument, often referred to as the codebook. Frequencies for all variables are broken down by empty answers, non-empty/non-special answers, and special answers. Enumerated type variables will include summary statistics for each response option. Integer type variables include mean, min, max, standard deviation, and the 25th and 75th percentile. Additional functionality is in development.

The new interface prompts the user to follow a sequence of steps:

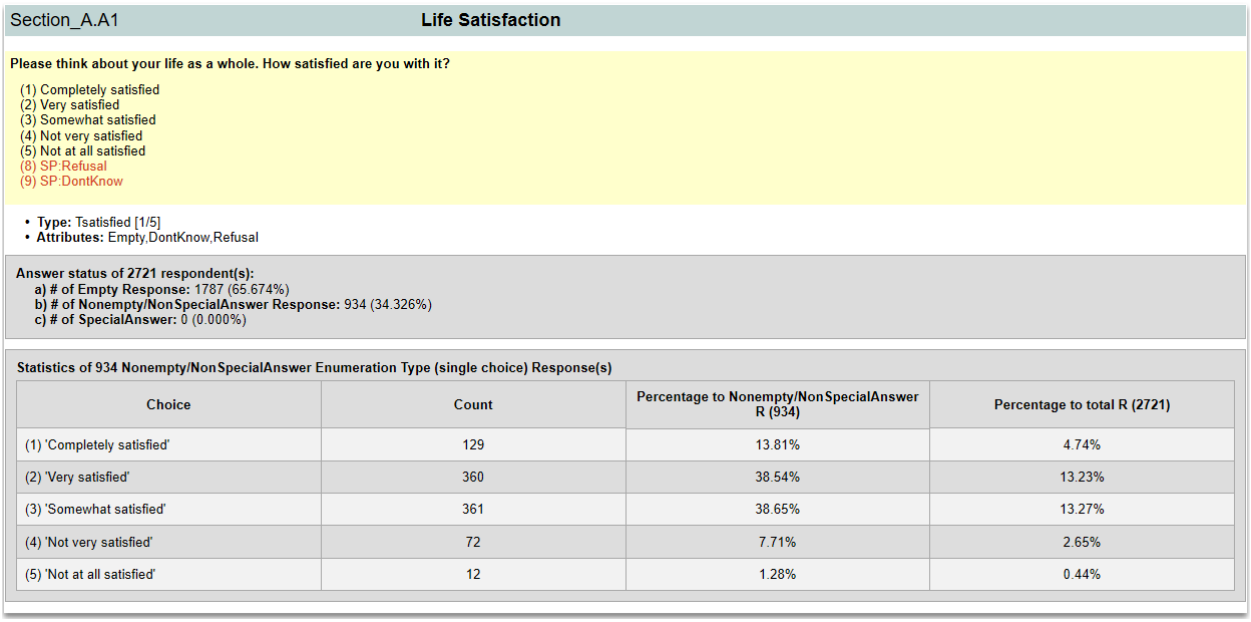
- **Step 1**: Select Blaise 5 Interface File (.bdix file) and Open Selected File
- **Step 2**: Select Mode
    - This field is automatically populated based on the content on the .bdix file. If there is more than one mode, the user will select from available options.
- **Step 3**: Choose Output location
- **Step 4:** Create Codebook

Figure 10: Create Codebook Selection Form



Below are examples of enumerated and integer variable output.

Figure 11: Sample Codebook Enumerated Output



Section_A.A1 **Life Satisfaction**

Please think about your life as a whole. How satisfied are you with it?

(1) Completely satisfied
(2) Very satisfied
(3) Somewhat satisfied
(4) Not very satisfied
(5) Not at all satisfied
(8) SP:Refusal
(9) SP:DontKnow

- **Type:** Tsatisfied [1/5]
- **Attributes:** Empty,DontKnow,Refusal

Answer status of 2721 respondent(s):
a) # of Empty Response: 1787 (65.674%)
b) # of Nonempty/NonSpecialAnswer Response: 934 (34.326%)
c) # of SpecialAnswer: 0 (0.000%)

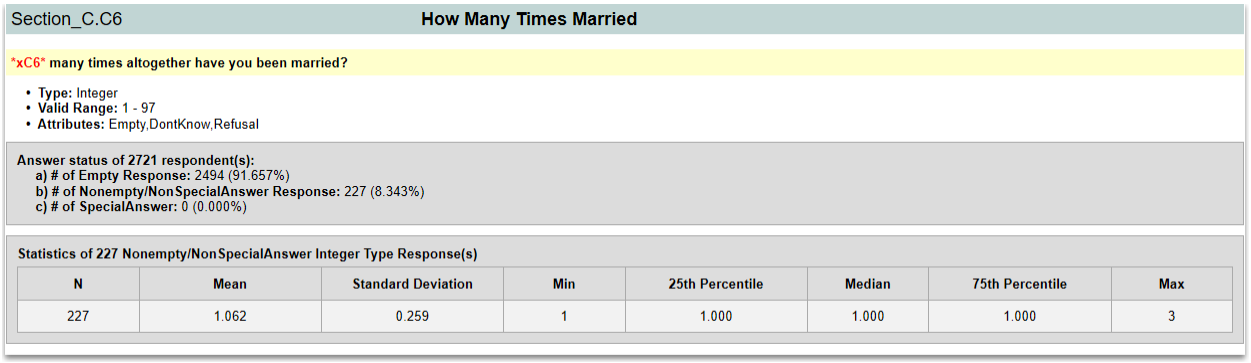
Statistics of 934 Nonempty/NonSpecialAnswer Enumeration Type (single choice) Response(s)

Choice	Count	Percentage to Nonempty/NonSpecialAnswer R (934)	Percentage to total R (2721)
(1) 'Completely satisfied'	129	13.81%	4.74%
(2) 'Very satisfied'	360	38.54%	13.23%
(3) 'Somewhat satisfied'	361	38.65%	13.27%
(4) 'Not very satisfied'	72	7.71%	2.65%
(5) 'Not at all satisfied'	12	1.28%	0.44%

Enumerated variables will summarize each response by the following:

- Counts: Number of responses
- Percentage to non empty and non special Answers
- Percentage of total responses

Figure 12: Sample Codebook Integer Output



Section_C.C6 **How Many Times Married**

xC6 **many times altogether have you been married?**

- **Type:** Integer
- **Valid Range:** 1 - 97
- **Attributes:** Empty,DontKnow,Refusal

Answer status of 2721 respondent(s):
a) # of Empty Response: 2494 (91.657%)
b) # of Nonempty/NonSpecialAnswer Response: 227 (8.343%)
c) # of SpecialAnswer: 0 (0.000%)

Statistics of 227 Nonempty/NonSpecialAnswer Integer Type Response(s)

N	Mean	Standard Deviation	Min	25th Percentile	Median	75th Percentile	Max
227	1.062	0.259	1	1.000	1.000	1.000	3

Integer variables will be summarized in the following metrics:

- (N) Number of responses
- Mean
- Standard Deviation
- Minimum

- 25th Percentile
- Median
- 75th Percentile
- Maximum

Future functionality:

- Ability to select a subset of variables for output.
- Include routing logic and universe statements for each variable.
- Additional output formats.

5. References

Sparks, P. & Liu, Y. (2004). Blaise Documentation System. The proceedings of the 9th International Blaise Users Conference (IBUC). Gatineau, Québec.

Guyer, H. & Cheung, G. (2007). Michigan Questionnaire Documentation System (MQDS): A User's Perspective. The proceedings of the 11th International Blaise Users Conference (IBUC). Annapolis, Maryland.

Dinkelmann, K., Kirgis, N., and Cheung, G. (2009). Michigan Questionnaire Documentation System, Version 3 (MQDS-V3). The proceedings of the 12th International Blaise Users Conference (IBUC). Riga, Latvia.

Stats NZ's Blaise-Azure environment = Blaise 5 + Azure (B2C + Offshore Data Cloud Storage)

Lynley Speers and Graeme Simpson, Stats NZ

Statistics New Zealand could not present due to the time difference but provided a paper of their presentation.

1. Abstract

Stats NZ is in the process of transitioning Blaise 4.8 to Blaise 5. Using Blaise 5 means that the current back-end functionalities we have used to deploy Blaise 4.8 survey questionnaires will completely change.

Household surveys are currently completed through face-to-face with our field interviewers. Subsequent interviewers for the Household Labour Force Survey (HLFS), quarters 2 to 8 are usually done by phone by our Contact Centre. Surveys are completed off-line using a laptop or a desktop. Blaise 4.8 is used to create customised off-line questionnaires for Stats NZ's household surveys. The offline Blaise 4.8 questionnaire is uploaded to laptops of field or desktops of Contact Centre staff. These are delivered via Lotus Notes. Replication is used to upload the completed interviews in the IDE Load Area.

Stats NZ is testing Blaise 5 such that respondents will be able to complete their surveys online in addition to the current CAPI and CATI. To do so, Blaise 5 must integrate with our current IT systems (e.g. the use of Microsoft Azure Cloud, Salesforce customer relationship, Trak case management, EPIC platform) and should align with the DevOps model of our Digital Business Services. Stats NZ has endeavoured to build the automation of environments using Infrastructure as Code (IaC) to rapidly deploy solutions that are able to communicate to cloud solutions and the onshore enterprise solutions. This includes deploying Extract Transform Load (ETL) tools into the cloud in order to make the overall solution flexible and scalable to other surveys in the future.

Our presentation also includes the challenges to secure the privacy and confidentiality of survey respondents' data in near-shore Australia cloud, maintenance of our social license as a trustworthy organisation in protecting respondents' data, value-for-money of onshore vs off-shore cloud storage and integrating Blaise 5 with our current IT systems.

2. Introduction

At Stats NZ, we are faced with limited financial resources compounded with demands from our data suppliers to make it easy for them to comply with their mandatory obligation in providing data to us. Therefore, we need to find ways to balance our expenses and make-it-easy for data suppliers. One such way is to digitally source collection of survey data, which is presumed to reduce operating expenses and improve survey completion.

In 2018, we embarked on two projects to address digital survey collection: Modernise Collections (overarching plan) and Digital Sourcing (implementation). During the discovery phase of Modernise Collect, we evaluated existing questionnaire tools - Blaise and another questionnaire tool we use for business surveys and the 2018 Census - to establish which one is the best fit to address our requirements. There are other tools in the market, but we limited our evaluation to existing tools because we only need an extension of contract, not the lengthy procurement process for a new application.

Based on our evaluation, Blaise 5 is the recommended and most suitable questionnaire tool for statistical business and social (household and individual) surveys. The recommendation was approved by our Executive Leadership Team. We then embarked on a comprehensive Modernise Statistical Operations (MSO) project and subsequently sought funding for an end-to-end proof-of-concept using Blaise 5 for one household and one business survey.

The succeeding sections provide information on our journey from starting MSO, to testing a prototype survey built in Blaise 5, to integrating Blaise 5 with our current IT systems including Microsoft Azure, to mitigating risks in storing data in offshore cloud.

2.1 Background

2.1.1 Problems encountered

At Stats NZ, we have increasing challenges such as: increasing organisational pressures, increasing demands from customers, increasing demands from data suppliers to make-it-easy for them to comply with their legal obligation in completing a Stats NZ survey and/or supplying administrative data, and a siloed and disconnected statistical operating model.

2.1.2 Reasons to modernise

To address and lessen the impacts of these challenges, we first evaluated our current collection operations (surveys and provision of administrative data), decided to modernise data sourcing, and commenced to modernise statistical survey collection. In 2018 we started the 'Modernise Collections' project which was later renamed to 'Modernise Statistical Operations' (MSO) in May 2019.

In the Modernise Collections period, we explored and evaluated current operations, and established whether these are still relevant to address our current and future challenges. In the exploratory and discovery phases of Modernise Collect we delivered the target operating model, high-level design, and existing tool testing and evaluation. These three outputs served as blueprints as to what we want and where we want to be in the medium to long term.

Our reasons for modernisation are to:

- work collaboratively and improve the culture at Stats NZ
- reduce the impact of legacy systems
- improve how the New Zealand public experience interacting with Stats NZ
- provide more options for collection channels
- digitise, where possible

2.1.3 Modernise Statistical Operations (MSO)13 project

MSO is a short-term initiative to address our challenges. It is focused on implementing sustainable and efficient processes within the statistical value chain.

[13] Sources: Krause J. High-level design, 2018, and Watt J, Target Operating Model, January 2019, Modernise Collect Discovery Phase, Stats NZ

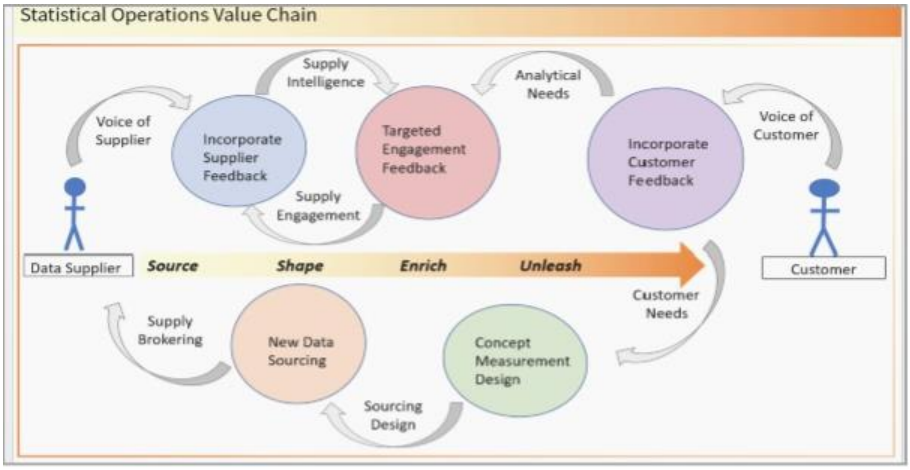
Collaboration is the key to MSO's success. This means that we work together to leverage our specific expertise on an ongoing basis. This involves reviewing business processes with a view to further optimise and automate, including responding to new customer and data supplier needs.

Characteristics of the "Modernise Statistical Operations" Operating Model

Our desired target state is for Stats NZ to have a sustainable and modernised statistical operation. The modernised statistical operations must exhibit the following characteristics:

- Highly transparent and collaborative
- Cross functional
- Connected business processes
- Business enabled tools
- Automated and algorithmic
- Adaptive and sustainable
- Responsive to customer needs
- Adaptive to data supply opportunities
- Best use of our human and technology capital

The MSO Model[14] largely focuses on the sourcing, shaping and enriching of data. However, these cannot be viewed in isolation. Instead this needs to be considered and treated as integral within the end-to-end value chain as illustrated below.



Target state attributes

To achieve the target state of a modernised statistical operations, we want:

- an end-to-end design of business processes that focuses on coordination and integration across the statistical value-chain.
- more focus on customers
- more focus on suppliers

[14] Source: Krause J, High-level design, September 2018, Modernise Collect Discovery Phase, Stats NZ, Wellington New Zealand

- more data discovery and use of data across the data ecosystem
- evidence-based decision making that eliminates 'unconscious-bias' in delivering business transformation
- increased automation
- drive to lower-cost digital sourcing options (business and social surveys and administrative data)
- more business enablement and better utilisation of business enabled tools
- increased standardisation of business processes
- more focus on data management
- collaboration and cross functional teams with team members contributing their specialist expertise
- sustainable, fast-paced, expert-led, self-sufficient and empowered change.

Working parameters
To achieve the MSO outcome, we will consider a digital solution or an operational approach if it delivers the following:

- Integrated end-to-end design
- Single supplier / customer view
- Easy to respond
- Adaptive and responsive
- Digital first
- Business-enabled enterprise tools

MSO will minimise impact on Stats NZ operations by creating a parallel digital environment to trial and test new solutions before implementing them in BAU.

There are several stages to get us to where we want to be.

- In 2018, we completed the exploration15 and discovery16 phases of Modernise Collect.
- In 2019, we set-up the MSO incubator.
- In September 2019, we commenced the prototype testing using Blaise 5 for one business and one social survey. This work will be completed in June 2020.

2.2 Stats NZ survey data collection

2.2.1 Current survey data collection

- Business surveys are completed by filling in paper, editable PDF, Excel, or online versions of the questionnaire.

- Household surveys are completed through face-to-face interview with our field interviewers. Subsequent interviews for the Household Labour Force Survey (HLFS), quarters 2 to 8 are usually done by phone. Surveys are completed off-line by field staff using a laptop. Blaise 4.8 is the questionnaire development tool used to create customised off-line questionnaires for Stats NZ's household surveys. The offline questionnaire is uploaded to laptops of field interviewers

[15] Exploration Phase delivered environmental scans and discussions which became basis of MSO's conceptual framework.
[16] Discovery Phase delivered the following outputs: Target Operating Model, High-level Design, and Existing Tool Testing and Evaluation.

and are delivered via Lotus Notes. Replication is used to upload the completed questionnaires in the IDE Load Area. Stats NZ Contact Centre also uses Blaise 4.8 to conduct computer-assisted telephone interviews (CATI) for the HLFS.

- To deliver the online option for some BAU business and household surveys and the 2018 Census, we outsourced the collection tool development.

- In November 2018, we completed a tool evaluation17 which recommended Blaise 5 as the survey questionnaire tool to be used to progress our online mode for survey completion. However, we cannot progress using Blaise 5 since we only have one dedicated Blaise developer whose skill is in Blaise 4.8. To address this gap, in June 2019 we invited to Stats NZ Wellington main office two Blaise staff to train our 15 staff in Blaise 5. In July 2019, we changed to the Blaise 5 subscription which includes free Blaise 4.8 subscription. This allows us to develop, test and integrate Blaise 5 and concurrently maintain BAU social surveys in Blaise 4.8.

2.2.2 Target state of survey data collection and its benefits

Target state

- Tested multi-modal surveys and associated data flows, in the first instance, and then provision of an online option for social surveys.
- All business surveys are digitally collected
- Intra-organisational collaboration on continual survey improvement
- Connected and automated business processes across the statistical value chain

To demonstrate value, survey data collection benefits should:

- Improve response rates to online surveys
- Save money through; increased use of digital channels by data suppliers, and fewer paper questionnaires
- Survey questionnaires are easier for respondents to complete
- More responsive to customer needs
- Responsive design – use of test results and feedback to improve surveys
- Scalability – can scale time and size of collection
- Improve data quality (e.g. capture harder-to-reach segments of the population by providing easier to use channels).

2.3 Current IT systems

When we outsourced online collection tools for some of our business and social surveys, and the 2018 Census we were able to integrate online questionnaires with Salesforce, EPIC and onshore data storage. We are currently developing and testing Blaise 5 for an online business survey and experimenting on household respondent's preferences on social surveys. Our target is for Blaise 5 to be integrated with our current IT systems and data flows by June 2020.

[17] Source: Tañedo C, Existing Tool Testing and Evaluation, November 2018, Modernise Collect Discovery Phase, Stats NZ, Wellington, New Zealand

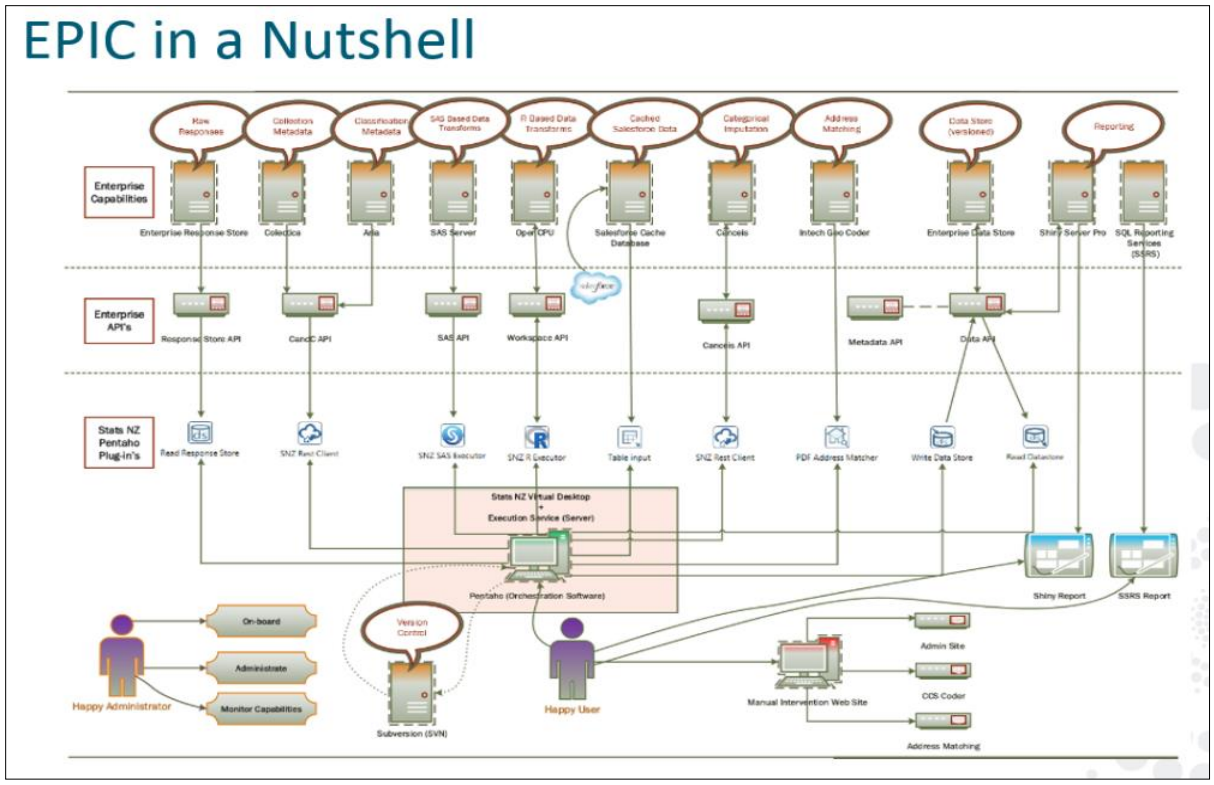
2.3.1 Salesforce

Salesforce is a respondent management tool we have used since 2016 when we started with an online option for Agriculture Production Survey. It was used in the 2018 Census and succeeding business and social surveys. Salesforce is used for managing survey respondents, recording of micro-data applications in Integrated Data Infrastructure (IDI), and engagement with external stakeholders (e.g. external agencies and Maori iwi relationships).

We are not planning to change Salesforce in the medium term therefore it is important that we can integrate Blaise 5 with Salesforce. Integrating Blaise 5 and Salesforce is feasible (refer to section 4).

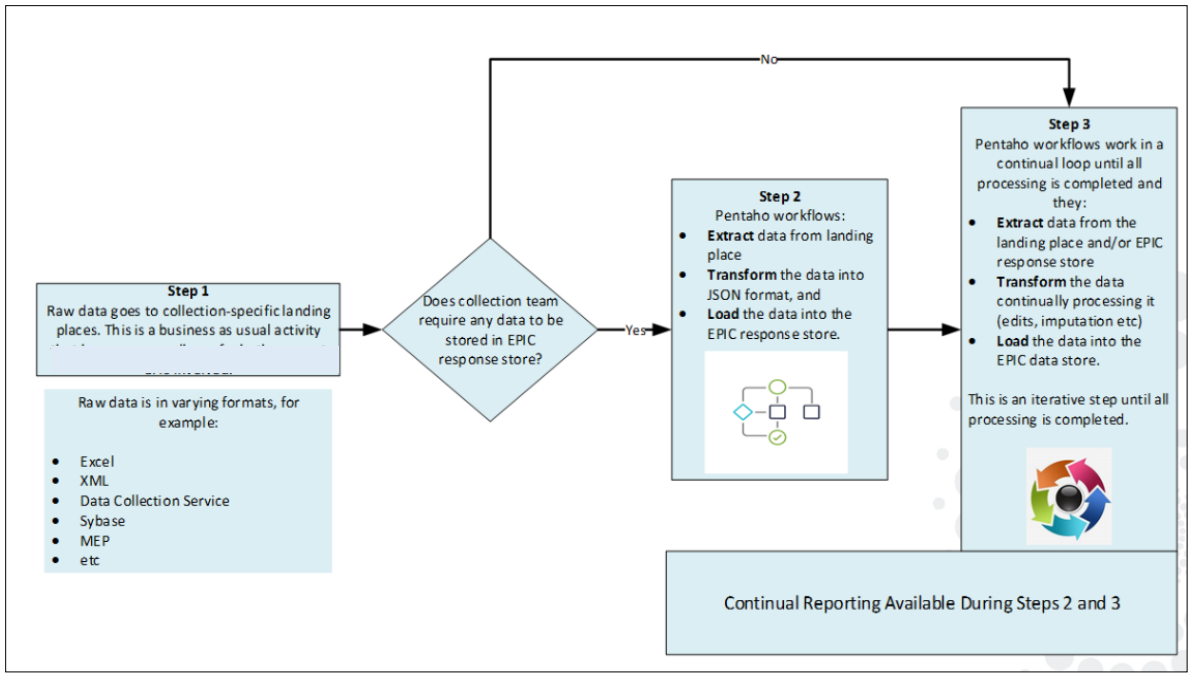
2.3.2 Enterprise Processing Integrated Capabilities (EPIC)18

EPIC is the collective name for a group of generic, integrated tools and services used to process raw data that has been collected from surveys and administrative data.



EPIC has a response store (a database used to store raw data prior to processing) wherein our Statistical Infrastructure works with our Collection Operations to ensure that their data flows from their collection-specific systems into the response store are ready for processing. (Refer to diagram below.)

[18] Source: Statistical Infrastructure, What is EPIC, January 2020, Stats NZ, Wellington, New Zealand



Like Salesforce, we are not planning to discontinue EPIC, thus it is important that Blaise 5 integrates with EPIC. Integrating Blaise 5 and EPIC is feasible (refer to section 4).

2.3.3 Storage of respondent data

The transition from Blaise 4.8 to Blaise 5 will also involve the storage of data, both in transit and at rest on SQL databases with Microsoft Azure. Survey respondents' data will be collected from Computer-Assisted Personal Interviewing (CAPI), Computer-Assisted Telephone Interviewing (CATI), and self-completion Computer-Assisted Web Interviewing (CAWI).

- Onshore storage in New Zealand

  Respondent data is collected, primarily via interviewer-led collection or online and is processed by our systems to identify the data and correct data issues. It is then stored in onshore (New Zealand) data cloud storage using All of Government Infrastructure as a Service (IaaS) servers. The data is accessed from this onshore data storage location by Stats NZ either for the purposes of publishing data, such as CPI measures, or for analytical purposes, such as the Integrated Data Infrastructure data lab.

  Onshore cloud storage is not financially sustainable to Stats NZ. In the medium to long term, we are considering offshore data cloud storage.

- Offshore (Australia) Microsoft Azure data cloud storage

  Data will be collected in the same manner as today following the same processes and governance mechanisms. It will then be stored in a hybrid environment that will be a combination of the current infrastructure services as well as cloud storage in Microsoft Azure servers located in Melbourne and Sydney, Australia. The data will be accessed from this location by Stats NZ either for the purposes of publishing data or for analytical purposes.

Offshore Microsoft Azure data cloud storage has been in Stats NZ's plans since 2016. It was delayed while we worked through our technical, security and legislative requirements and ensured we adhered to all of government requirements. We now have the necessary controls; thus, it is important that Blaise 5 integrates with Microsoft Azure.

3. Blaise 5 self-response session management[19]

3.1 Questionnaire completion in multiple sessions

Feedback from businesses has indicated that being able to complete the survey in multiple "sessions" will improve their user experience. Currently, on receipt of the pre-notification letter (PNL), business respondents are required to gather all information before completing the survey questionnaire. However, this is not always possible because specific questions are only known as they go through the survey. Having to provide responses to specific questions, when these are not readily available, increases respondent burden. Businesses gather data from their colleagues to complete the survey. Thus, it is important that they can logout and login multiple times to fill-in the gaps whilst completing a survey.

Completion of a survey over multiple sessions is a core part of Blaise 5 offering. Blaise 5 can integrate authenticated access to the questionnaire and thus ensure that completion of the questionnaire is carried out by an authorised individual.
After feedback from business survey respondents, we reviewed several options to improve user (survey respondents) experience and at the same time secure their data. Four options were considered:

- Option 1: A cloud based external authentication wherein the Azure B2C manages user authentication
- Option 2: Stats NZ manages basic user authentication within the questionnaire logic
- Option 3: No user authentication, the respondent will complete the questionnaire in one sitting
- Option 4: Use of 'Real Me[20]' (basic or verified) as the Customer Identity and Access Management (CIAM) solution to authenticate respondent.

The most viable option is option 1: Integrated authentication of survey respondents using pre-setup credentials where the Azure B2C manages user authentication.

3.2 Azure B2C to authenticate survey respondents using pre-setup credentials

The preferred option is to implement a customer identity (authentication) solution using **Azure B2C**, a cloud-based external authentication. This solution will:

- enable management of respondent identities to be secure
- allow respondents to come and go using their identity to log in and out
- allow implementation without some restrictions, if trying to implement purely in Blaise questionnaire logic.

[19] Sources: Jog S and Krause J, MSO Blaise 5 Self Response Session Management Options Paper, October 2019, and Options Paper Addendum, November 2019, Stats NZ, Wellington, New Zealand
[20] Department of Internal Affairs' authentication/verification system for any online transactions with New Zealand government agencies.

3.2.1 How Blaise 5 and Azure B2C works

Azure can manage customer/supplier identifiers in a manner that can be seamlessly integrated into our online questionnaire platform (Blaise5).

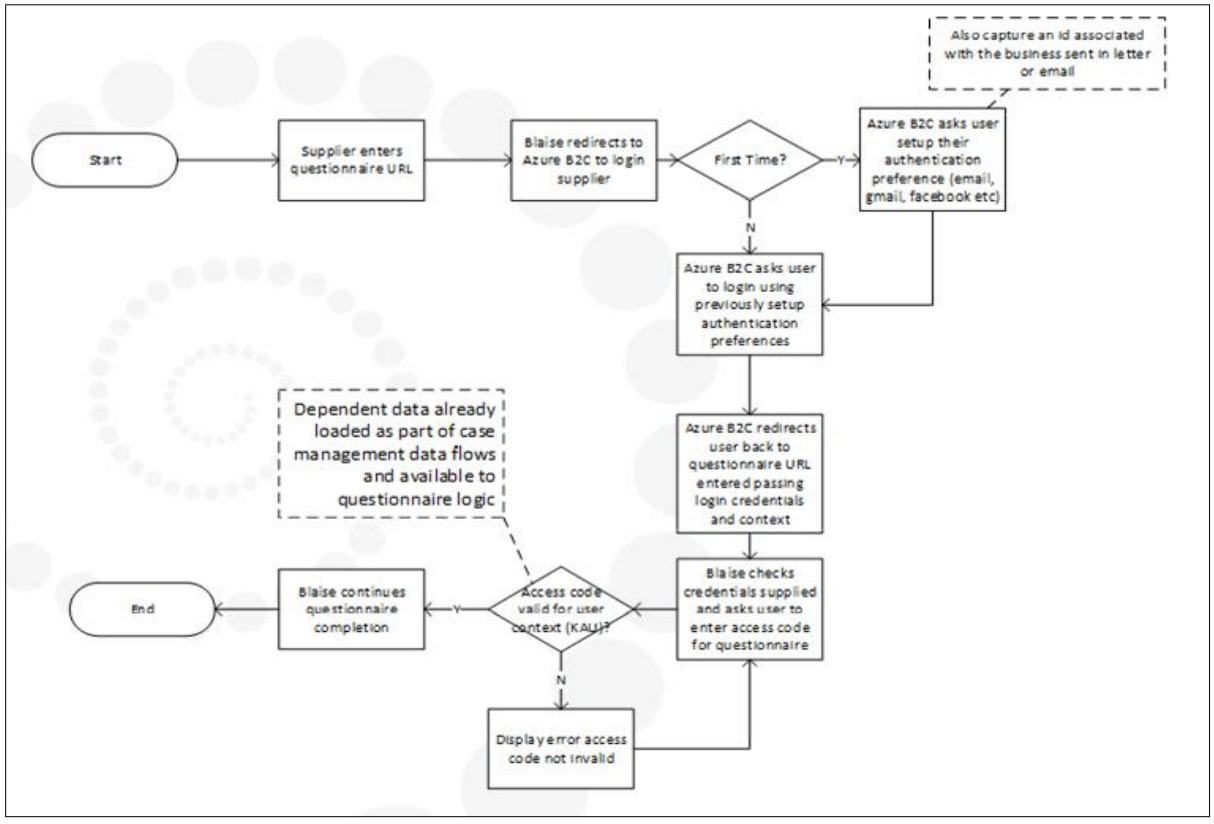
- **First**: A unique access code entered at the start of the questionnaire associates the response with a case that has been pre-filled with data required by questionnaire logic. Prefill of case data is handled by separate data management workflows outside of Blaise. Enforced unique response for access code managed by default in the Blaise platform.

- **Second**: Before completing an online questionnaire, a supplier will be asked to login with a username and password which are managed by Azure B2C or use one of the externally supported authentication mechanisms such as Gmail or Facebook. For first time respondents, they are required to register on Azure B2C.

- **Third**: Once authenticated, Blaise will check that there is an authenticated user completing the questionnaire. If not, then the user is directed back to an Azure B2C login before continuing to complete the questionnaire. User credentials are available to Blaise to stamp onto the questionnaire response data as they complete the online questionnaire.

- **Fourth**: If the respondent is returning to a partially completed questionnaire, Blaise will check that the authenticated user is authorised to complete the questionnaire (e.g. if they are the same user or someone else in the same organisation).

Below is the diagram on how Blaise 5 and Azure B2C works:



3.2.2 Advantages in using Azure B2C

- Azure B2C manages user identity in a self-service manner (e.g. with minimal/no ongoing Stats NZ environment)
- Users get to choose how they authenticate to Stats NZ. They can use a mechanism they already use avoiding the need to remember another password for yet another system.
- Enables flexibility and thus allows users to complete the questionnaire in multiple sessions.
- Access to any sensitive information displayed or used behind the scenes is controlled.
- Sensitive information can be used directly or indirectly to improve user experience (e.g. in making-it-easy for respondents to complete the questionnaire).
- It can be extended to more stringent security checks if required with no development effort (e.g. Multi-Factor Authentication – MFA).
- Adopting Azure B2C now is a step towards a data supplier portal should we decide to implement one in the future. In this case, user authentication is decoupled from questionnaire logic.
- We can setup Blaise cases ahead of time, pre-filled with dependent data and associated with the primary key (e.g. Access code).
- Makes multi-mode implementation more straight-forward as authentication is handled outside of Blaise questionnaire logic.

3.2.3 Disadvantages in using Azure B2C

- Increased integration development time (approximately 8 weeks) and costs (estimated at NZ$120k) to integrate into the Blaise 5 platform. There is a possibility that the 8 weeks can be shortened if we use CBS Blaise development resource in the initial development.
- Ongoing costs increase if we exceed the free volume of 50,000 authentications per month. (As at 23 Oct 2019, the next 950,000 authentications are charged at US$ 0.0028 per authentication per month or US$2,660 or NZ$ 4,164.04).

4. Microsoft Azure offshore data cloud storage21

As previously mentioned, the transition from Blaise 4.8 to Blaise 5 will involve the authentication of respondents and storage of data both in transit and at rest on SQL databases with Microsoft Azure offshore cloud.

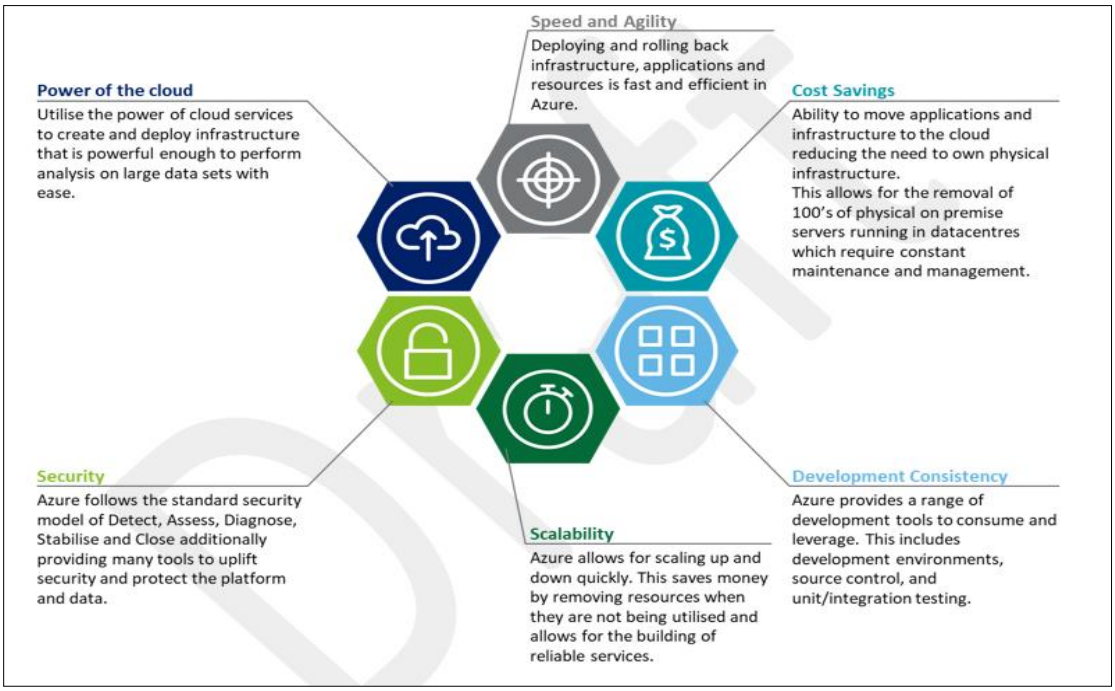
We have evaluated the potential of offshore cloud storage since 2016 (since Salesforce was adopted as our enterprise relationship management system). Offshore cloud storage is financially sustainable, but there are IT security risks which must be assessed and certified if offshore cloud storage is securely viable or not.

4.1 Benefits in offshoring cloud data storage

Cloud storage is not new, it is widely used by enterprises and individuals. There are benefits to cloud storage. Storing in the cloud is good value-for-money, given we have robust policy controls on privacy, security and confidentiality of information supplied to us.

[21] Source: Digital Business Services, Privacy Impact Assessment of offshoring Stats NZ cloud data to Microsoft Azure 365 hosted in Australia, November 2019, Stats NZ, Wellington, New Zealand

It is an important step forward for Stats NZ to migrate the data and processing power to the cloud. It brings several benefits to the business as shown in the figure below, including redundancy, scalability, cost, and security.



4.2 Major risks and mitigation in offshore cloud storage

Offshore cloud storage is perceived by the public to be too risky. That is, data from respondents may prejudice business and individual information and in worst case scenario may damage the economy of New Zealand, impede trade negotiations with other countries, and risk the reputation of an individual.

If data in the cloud gets hacked and jeopardises the details of our respondents, consequently, the public and government's confidence on us as a trusted agency will be reduced and negatively impact on our social license to protect the data supplied to us.

To mitigate the risks of offshore cloud storage, we have both external and internal controls and policies and guidance in place. Externally, we have clear guidance from the government on offshore cloud storage including its use. Internally, we have a robust policy reflecting the government's guidance. Having these enabled us to balance technological changes, our social license, and the desired outcome for New Zealand.

4.2.1 Offshore (Sydney and Melbourne22) cloud storage controls

- Stats NZ's survey respondents will be advised, at the first instance, where their response data is stored. They must be assured that their responses are safe, and their privacy is maintained regardless of storage location.

[22] To verify this perception, in 2018 Stats NZ contracted AC Nielsen to conduct a social license research. Results show that only one percent of respondents were concerned with the location of stored data. This level of concern is consistent across those who trust and those who do not trust Stats NZ as an organisation. (Source: Digital Business

- The New Zealand public is aware that we follow stringent processes to best protect the privacy of the New Zealand public to achieve our legislated privacy obligations in New Zealand (e.g. Statistics Act 1975 and Privacy Act 1993) and the controls in Australia (e.g. Privacy Act 1988, Telecommunications and other legislation amendment – assistance and access - Act 2018) where it will be stored. In addition, we have also considered lawful requests for access to our data and Microsoft's response process.

4.2.2 Data security controls

Security requirements and drivers in the cloud differ significantly from traditional data centre environments requiring new security models and architectures. Characteristics of cloud data storage:

- dynamic nature of the cloud and its related infrastructure
- no customer ownership or control of infrastructure
- limited visibility of architectures and transparency of operations
- shared (multi-tenanted) physical and virtual environments
- may require changes to the architecture of agency system to optimise use of cloud services.

To address these issues in data security, we robustly investigated the:

- Use of encryption and control of cryptographic keys

  As defined in the New Zealand Information Security Manual (NZISM) released by Government Communications Security Bureau (GCSB), prior to adoption of cloud-based services we should assess the need for encryption for data in transit and at rest. We have assessed that all Sensitive or Personally Identifiable Information, or information that forms part of other taonga of New Zealand, should be appropriately encrypted while in transit and at rest.

  To maintain sovereign control of our data, our current Information Privacy, Security and Confidentiality (IPSaC) controls state that we need to retain control of cryptographic keys. Retention of cryptographic keys means that we control access to all data, with no-one able to gain access to the data without our permission. Retention of cryptographic keys is our mitigation action for most of the privacy risks.

  Cryptographic keys work by acting as a codebreaker, where data is encoded in a way that cannot be understood without access to the key. This key cannot be subverted by modern computers, which would take centuries to break the code. This means that to access the data, one must possess both the data and the cryptographic keys. We can, therefore, prevent unauthorised data access even when data is not physically located in New Zealand.

  Note: Where data is to be processed in the public cloud by an application, the data needs to be decrypted before processing. As a result, an unencrypted snapshot of the data is stored **temporarily (milliseconds)** in the cloud before being overwritten.

Services, Privacy Impact Assessment of offshoring Stats NZ cloud data to Microsoft Azure 365 hosted in Australia, November 2019, Stats NZ, Wellington, New Zealand)

- Retrenchment of offshored data
  If the need arises, such as if the jurisdictional risk profile changed, Stats NZ would be able to retrench data to New Zealand, in its entirety. This process would be documented to ensure that this could be completed in a timely manner and is a key mitigation in ensuring that our data holdings remain secure.

  To provide further protection, a copy of all Sensitive or Personally Identifiable Information, or information that forms part of other taonga of New Zealand, will be held on shore.

- Microsoft's processes

  Microsoft's Online Service Terms (OST) has been used as a guide in assessing what contractual commitments can be made.

  Further focussing our assessment on Microsoft Office 365 and Azure, enables us to leverage the additional All of Government contractual arrangement put in place by New Zealand's Department of Internal affairs.

4.3 Ongoing mitigation actions to minimise risks

Our ongoing mitigation actions to minimise risks which may arise anytime are to:

- Inform survey respondents (regardless of who they are) at the first instance of contact (e.g. pre-notification letter/email/phone that they are selected to participate in Stats NZ's statistical survey, and our website) where their data is stored and to assure them that it is safe.

- Engage with Māori at the first instance where data is stored and why. Although there is some work being done with Māori on data, there will need to be significant additional work relating to cloud. We will need to engage with Māori on:
  - Data handling in the cloud, this includes creating awareness that it will be collected, processed and potentially stored within O365/Azure.
  - Azure security measures that will be utilised to protect data and explaining that some of these may not be available through current on-premise solutions.
  - The benefits of using Azure. This will help assure Māori that providing data to Stats NZ (or alternative agencies) will not only help them and the whole of New Zealand.
  - How Māori data sovereignty expectations can be honoured in a cloud environment.

- Update privacy policy to clearly outline to suppliers that their data will be processed and stored in Australia. Assure them that this aligns with the NZ privacy principles and other relevant legislative requirements.

- Continue to update the Security Certification and Accreditation (C & A) of hosting environments and systems. Make sure that C&A is relevant.

- Inform the public through the Stats NZ website (www.stats.govt.nz). It is an accessible and transparent channel for NZ public to understand the benefits in using MS Azure in Australia. This should include information about potential use cases, security and privacy considerations.

- Configure systems to ensure encryption of data in transit and at rest with sovereign retention of cryptographic keys.

- Integrate the current Security Information and Event Management (SIEM) capabilities to monitor the Microsoft cloud environments for integrated event reporting and management.

- Adapt and implement a formal Committed Information Rate (CIR) process to assist in the event of a cyber incident related to O365/Azure.

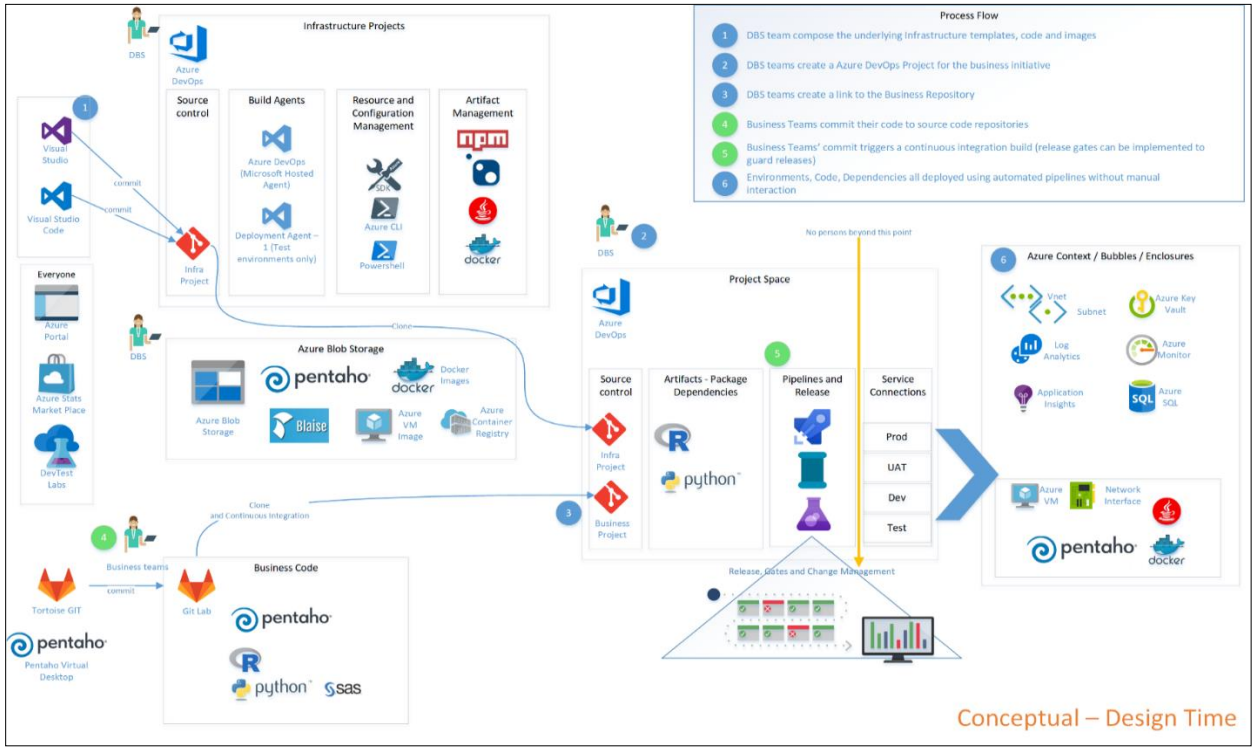
- Implement a Cloud Access Service Broker to gain better oversight of how users are interacting with cloud applications and services.

- Review and enhance existing security architectures and systems design to prudently manage the changed risk, technology and security environment in adopting cloud services.

5. Architectural design - integrating Blaise 5 and Microsoft Azure[23]

5.1 Environment work stream

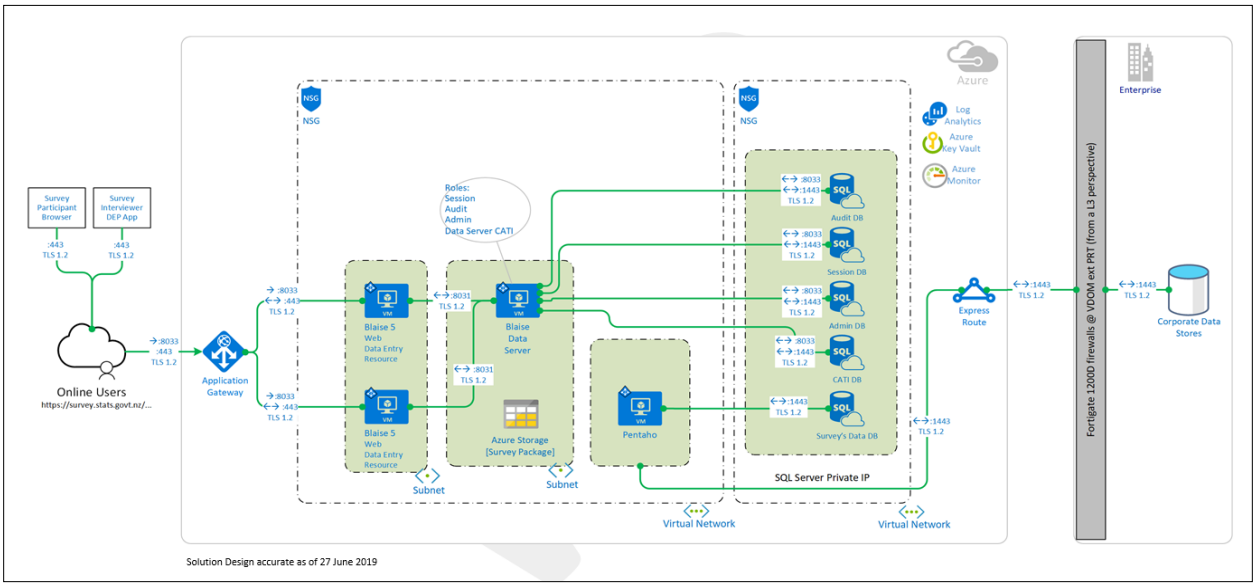
Below is the design diagram reflecting how MS Azure fits into the new architecture resulting from the use of Blaise 5.



We have built the automation of environments using Infrastructure as Code (IaC) to rapidly deploy solutions that are able to communicate to cloud solutions and the onshore enterprise solutions. This includes deploying Extract Transform Load (ETL) tools into the cloud in order to make the overall solution flexible and scalable to other surveys in the future.

[23] Suyog Jog and Clinten Gillespie, Solutions Architects, Digital Business Services, Stats NZ

5.2 Solution Architecture Diagram24



Solution flows

- A respondent will be sent a short URL code (generated via Salesforce), which will allow them to access the survey via their web browser. When they browse to the survey, they interact with the Blaise 525 web server. This server also holds the data entry resource applications. This server will sit behind the Azure Application Gateway which provides services such as a web application firewall and load balancing to balance users between multiple web servers.

- During the respondent's session, various data will be logged on to their activity, through the audit trail log and session log. This may include which page they are on, which field they are up to or record every keystroke they make on the form. The session log is used to restore a user's session if they do not complete it in one sitting and it will persist in the Azure database until their final survey answers have been written to the Azure Blaise database.

- Stats NZ staff will use the Azure Pipeline to continually deploy code to the environment. Blaise will need code from an infrastructure perspective and from a business perspective. The infrastructure code will determine the configurations of the Blaise/Azure environment, while the business code will dictate the workflows, business rules and function of the surveys. There may be multiple users writing code for the business side of Blaise. Both sets of code are checked in to source code repositories (e.g. GitLab), where it has version control and can be peer reviewed. The code is taken from the source code repository and put into a project space in Azure DevOps, where automated/manual tests can be performed before being run through the Test, Dev, UAT and finally deployed in the production environment.

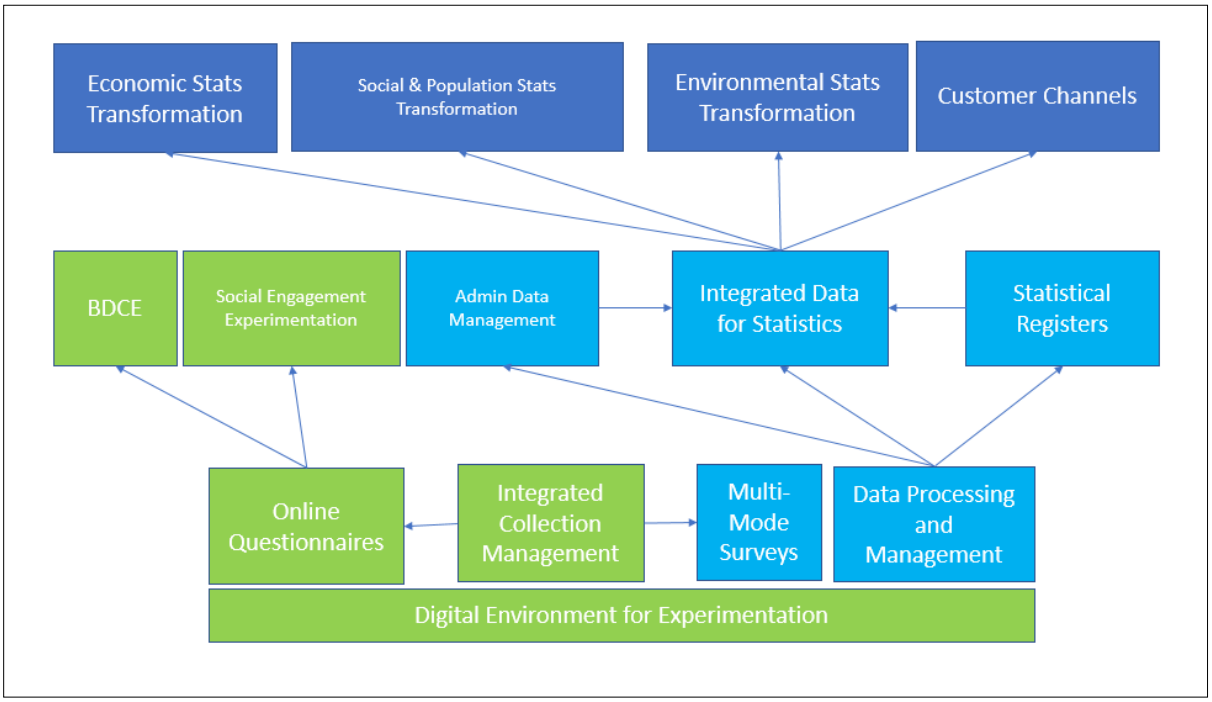
[24] Source: Modernise Statistical Operations, Blaise 5 – Information Security Certification and Accreditation Report, June 2019, Stats NZ, Wellington, New Zealand

- In most cases there will be a one-way communication flow from Azure to Stats NZ. However, some surveys may require previous respondent information to be used to populate the survey or dictate the logic flow of the survey. In this case the workflows will pull the needed information from the Stats NZ response store and pass that to Blaise.

- The Azure pipeline will use the Blaise server manager and admin server to deploy survey packages and Blaise files to the environment to allow respondents to complete their surveys.

Appendices

Appendix A: Diagram - Building business transformation on strong foundation



Appendix B: Blaise 5 – Tool evaluation results[26]

In the MSO Discovery Phase, we have delivered the following outputs: MSO target operating model, high-level design model, and evaluation of existing tools (e.g. Blaise and outsourced tools we are using for online questionnaire completion option).

In evaluating and choosing the tool to be used, we worked within the following parameters:

- Cost effectiveness of the tool. – the cost of investment and maintenance of the tool will produce cost savings and efficiency to Stats NZ in the medium to long term.
- Result in an acceptable experience for respondents (through delivery of front-end functionalities of the tool) –that we make-it-easy for respondents to complete the survey questionnaire.
- Integrate with our current IT systems (back-end functionalities of the tool). That is, when the tool is plugged-in, our Digital Business Services (DBS) team can integrate the tool with our current IT and security systems. For instance, we are using Salesforce, EPIC, and cloud storage specifically onshore cloud and proposed MS Azure offshore data cloud storage.
- Be sustainable –we can deliver our outputs, address changes in government policies and programmes, and align with the changing operating environment at optimal cost and least time.
- Align with the high-level collection design for a Modernised Stats NZ.

[26]Source: Tañedo C, Evaluation Report: Existing tool testing and evaluation, November 2018, Modernise Collect Discovery Phase, Data and Digital Services Development, Stats NZ, Wellington, New Zealand

Our existing tool evaluation recommends the upgrade of Blaise 4.8 to Blaise 5 due to the following reasons:

- Blaise 5 can deliver CAWI for both household and business surveys effectively and efficiently.
- License and maintenance costs are cheaper compared to alternative tools.
- Development is aligned with international best practice adopted by overseas statistical institutes.
- Continuous assistance and support are available.
- The provider is a national statistical agency, thus Blaise 5's availability is sustainable in the next five years and beyond. The existence of Blaise is linked to statistical surveys for official statistics.
- Stats NZ's existing household surveys can be converted from Blaise 4.8 to Blaise 5 using the provided conversion tool, thus saving development time. Business surveys would need to be developed in Blaise 5 from scratch.

Appendix C: Q & A on the use of Azure B2C to authenticate respondents[27]

Q: Do we have any control over the functionality to be able to log in via Gmail or Facebook? I think organisations wouldn't want their staff uploading data whilst logged into FB and it's probably not a good look for Stats NZ. This option might be acceptable to social survey respondents.

A: Yes, we can configure what we enable from an external authentication mechanism, for instance remove Facebook.

Q: How does option 1 work with 2 instruments? I understand that user #1 can see the data user #2 enters for the financial form. But if user #2 also completes the employment form, can user #1 view that data or can we lock it down? This will be a hot topic for some businesses who want to keep employment data confidential.

A: It is up to us how we conduct access control to the record after they have been authenticated. We could restrict access to only the user who started completing the questionnaire, or we could say any authenticated user associated with KAU can complete the response. Association with KAU via the Contact ID passed in with their Authentication (setup when they register). This feature can be altered on a questionnaire instrument basis.

Q: With option 1 what happens when something goes wrong or a user needs help? How do we support them? How about if they forget their password?

A: The 'forget password' is handled by the Azure B2C service. If they are using Gmail for example, this is then delegated to Google. There could also be scenarios where they moved organisation and forgot to update their registered contact ID (if we send them a new one). In this case we could prompt them to check the Contact Id registered in their profile which is provided in the letter.

Q: Can the Contact Centre use the Blaise tool to collect data over the phone. We are going to need to enable this otherwise it will directly impact our ability to meet response rates.

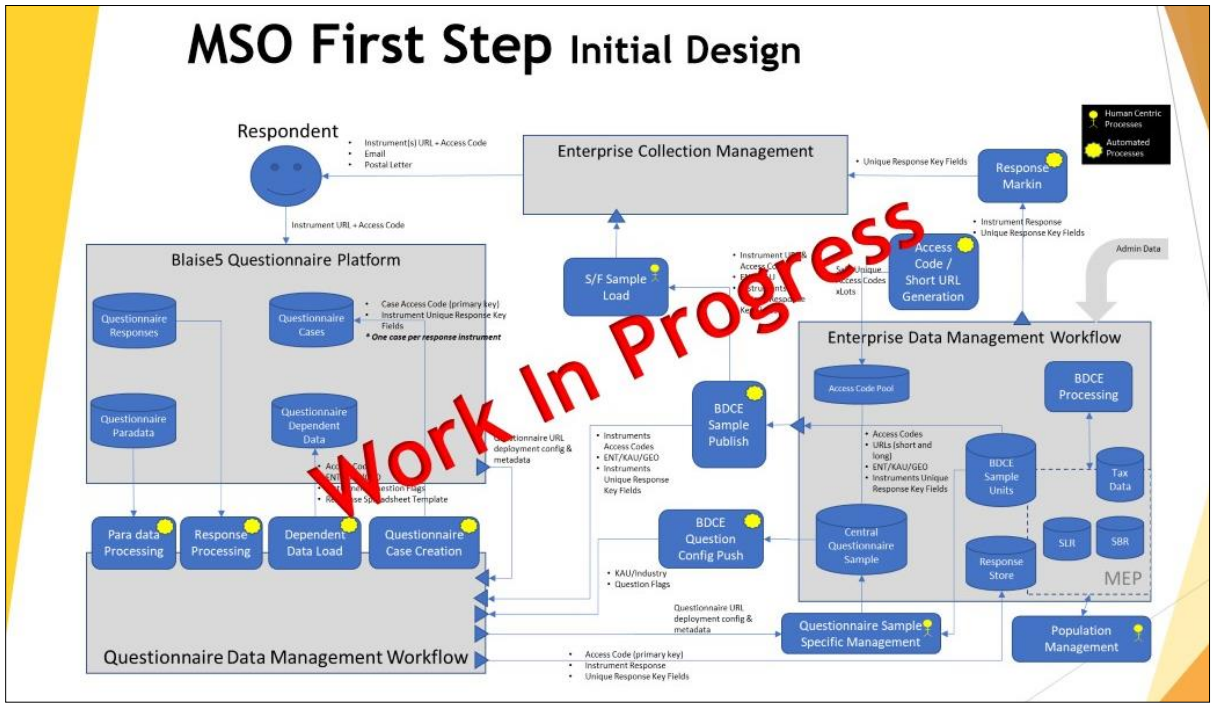
A: Yes. All you need is to be authenticated and the Stats NZ user's profile drives an override of the KAU/Access Code check. It will enable Stats NZ to complete and the already started questionnaire on behalf of the respondent.

[27] Source: Jog S and Krause J, MSO Blaise 5 Self Response Session Management Options Paper, October 2019, Stats NZ, Wellington, New Zealand

Q: How do we understand the impact on respondents?

A: We are testing this option with respondents (business initially) and get direct feedback, from them including what authentication preferences they have before we disable them (e.g. Facebook and/or Gmail).

Appendix D: MSO First Step initial design (from September 2019 to June 2020)




IBUC 2020
LIMASSOL