Back to the Future: Converting NCVS to Blaise5

Daniel Moshinsky, US Census Bureau

Purpose and scope of this paper

The primary objective of this test is to examine the functionality of the instrument under Blaise 5. Can we collect all the same data we collected in Blaise 4? What modifications, if any, would we need to make to our Blaise syntax, our instrument design, or our data collection procedures?

Questions about survey management, deployment capabilities, and data storage were outside of the scope of this test. The paper will touch on issues related to the layout and data storage, but it is not our central focus presently.

This test was intended solely to test a CAPI survey running on a Windows laptop. Thus we made no attempts to run the converted survey on other mobile devices.

It is understood that there are many features of Blaise 5 that have not been implemented at all, or that have not been implemented fully. Nevertheless, it is worthwhile to explore the capabilities of the Blaise 5 system both in order to provide developers with valuable feedback, and most importantly in order to better get to know the system.

What is NCVS and how does it work in Blaise 4

The National Crime Victimization Survey (NCVS) is a monthly household crime survey. It was first fielded as a Blaise 4.6 CAPI survey in 2006, and has since been converted to Blaise 4.8.4. Every household member over the age of 12 is asked about crimes that happened to them in the last 6 months.

The NCVS instrument initially conducts a "Screener" where the respondent is asked about types of crimes that happened (such as theft, assault, car theft, etc.), and about the frequency of each type of crime. Once the screener is complete, the "Incident Report" section is asked. Here the respondent is asked a long series of questions about details of each reported crime. For example, if R1 reports that his car was broken into 2 times, and he was assaulted 3 times, R1 will be asked to provide 5 Incident Reports.

After the Household Respondent's interview is complete, the rest of the household is interviewed in a similar fashion.

Due to the size of the instrument and the nested looping involved, it was not possible to program the instrument, in Blaise 4, in a straightforward fashion of looping through each person's incidents. We ran into page limits, block size limits, and performance lags. As a result, we implemented an "ask, store, empty, and re-ask" approach presented at the IBUC conference in Annapolis, 2007 ("*Challenges in Converting the National Crime Victimization Survey to Blaise*" Charlie Carter, Roberto Picha, Michael Mangiapane, Alexis Arrington and Daniel Moshinsky; US Census Bureau, USA).

In brief, this approach asked the screener questions in a non-arrayed block. It then created a rostered "storage" block of the same type. When the screener questions were completed, the values were copied into the "storage" block. Similarly, there was also one non-arrayed Incident Report block. After each incident report, the block was copied into the arrayed "storage" block along with linking keys. Then, the incident report block would be emptied and come back on route, and the process would repeat until there were no more incidents. When the interview with the respondent was complete, both the screener and incident report blocks would be emptied to prepare them for the next person.

This approach resulted in substantial improvement of run-time performance.

Why choose NCVS for the conversion test

The Census Bureau has been investigating Blaise 5 since its earliest test releases. Now that Blaise 5 is approaching maturity we wanted to implement a full conversion of one of our established Blaise 4 CAPI surveys into Blaise 5.

NCVS was chosen because of its complex Blaise language functionality and large size. We have previously successfully converted small surveys in our test environment, but this time we wanted to test the limits of Blaise 5 to see if we would uncover issues for Statistics Netherlands to address as it continues to develop the software.

In addition, NCVS does not heavily rely on the use of Manipula. Since Manipula is not yet available under Blaise 5, it was appropriate to pick a survey that is not reliant on it.

Conversion process

The specific steps in converting a Blaise 4 instrument are described in the Help documentation. For this paper, just the general steps are outlined, along with a few suggestions.

First, all components of the Blaise 4 instrument were converted to the latest available build of Blaise 4.8.4. In our case, it was 4.8.4. Build 1888. Next, the source code must be converted and prepared in Blaise 5 in order to generate a compiled datamodel. It was important to first make sure that the Project Manager in the Blaise Control Center included all the files associated with the project, including the modelib, and all the source search path locations were listed (without spaces) in the Blaise 4 ->Blaise 5 source converter utility.

The .bpf project file is then converted into a Blaise 5 solution (.bsol). Several issues had to be corrected in order to successfully compile the instrument under Blaise 5.

After a .bmix file is successfully generated, the test input data contained in our Blaise 4 database had to be converted to a Blaise 5 database. This Blaise 5 database will provide the input data needed for testing the NCVS instrument. Manipula was used to load test data (sample control values, address information, and existing information about the household) into the Blaise 4 case

database. We used the database conversion process to load the test data from the Blaise 4 database into our Blaise 5 database.

The converted .bdix and .bdbx files were placed in the location specified in the "Deploy Path" under File -> Options. Overall, the process of converting the source code and the main and external databases for this survey was relatively painless. Only minor modifications, that were self-explanatory, needed to be made.

Sets and Enumerated Lists

When answer categories are converted, answer category numbers do not display by default. The easiest way to display answer category numbers is to "Move Up" SetCodeText and EnumeratedCodeText in the list of Category Templates, such that SetCodeText is above Set, and EnumeratedCodeText is above Enumeration. Templates are applied in the order they appear on the list, if they meet the Applicability conditions.

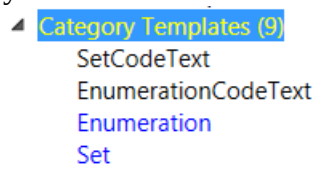

Figure 1. Re-ordered Category Templates

In addition, it may be necessary to change the default templates for answer lists. To do that, go to Input Controls > Data Value Templates > Answer List, and, in Properties, click on "DefaultTemplates" and add Category (EnumerationCodeText) and Category (SetCodeText).

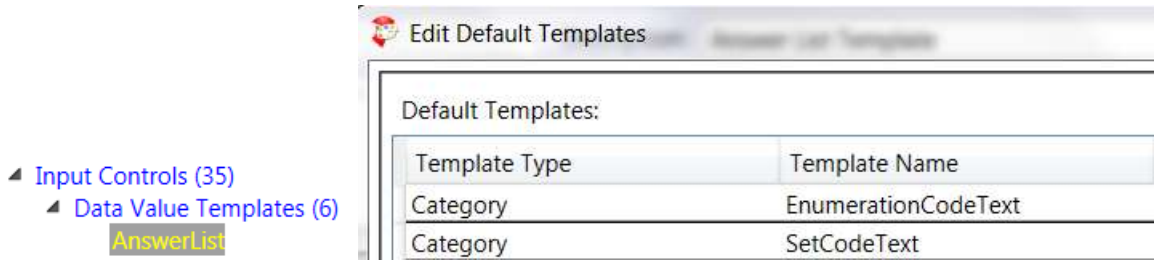

Figure 2. Setting the default template to use CodeText

After these changes, the answer category numbers will appear:

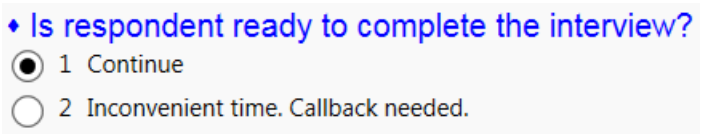

Figure 3. Numbers now appear next to answer categories.

Testing approach

Several testing scenarios were keyed with Blaise 4 and Blaise 5 instruments running side-by-side in order to test routing, layout, and performance issues. After each data entry session, the Blaise database was checked for the expected values of various significant fields. Specific issues uncovered during the testing are discussed in the sections that follow.

Each time a new release of Blaise 5 was made available, we attempted to re-test, as much as possible, the issues discussed in this paper. The latest available version at the time of this writing is 5.0.2.721.

Blaise 5 Instrument Testing Results

Attribute EMPTY, DONTKNOW not allowed for key field

Several of our external lookup files allowed for EMPTY in key fields (for example, CountryCode). Also the datamodel-wide setting was to permit DONTKNOW. This rule has been tightened in Blaise 5, and a compile-time error will result. So, if CASEID is your primary key, you will need to give it an explicit NODK attribute.

Compare functions

One of the biggest sources of generated warnings – as well as incorrect run-time behaviors have to do with a change in how compare functions, such as '=' and '<>' operate in Blaise 5. In fact, over 100 warnings were generated:

⚠	176	16	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	145	16	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	293	63	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	340	3	Layout section is ignored.
⚠	125	19	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	162	57	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	162	55	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	164	33	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	165	37	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	189	48	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	205	29	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	321	22	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	331	41	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	337	83	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	144	50	The meaning of this expression differs from Blaise 4: it is only 'false' when the left hand side has been assigned the value '0'.
⚠	250	21	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'
⚠	284	36	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	580	17	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	280	49	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	303	19	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	316	36	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	393	19	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	504	76	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	598	25	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	613	71	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	113	14	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	117	16	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	145	38	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	145	79	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.
⚠	171	37	The meaning of this expression differs from Blaise 4: it is only 'true' when the left hand side has been assigned the value '0'.

Figure 4. Expression warnings.

According to the Help documentation:

The compare functions '=' and '<>' consider values to be different from empty and DK and RF. (In Blaise 4.8 Empty, DK and RF are equal to the default value of the type of the field. Exception: an enumeration category with code 0 is not equal to empty). An empty string '' is unequal to the special value EMPTY.

When inspecting these warnings, it was difficult to tell which of these created a potential instrument issue, and which could be ignored. For example, many of the warnings were caused by the line:

```
If mode = '0' then…
```

The value of "mode" is filled from the input file. It is a field of type STRING[1], and it can be set to '0'. Thus comparing the value to '0' is perfectly appropriate.

Other instances of this warning message, however, did point to a real problem. For example:

```
if Roster[I] = '' then …
```

Warning - The result of the expression is always 'false'. Use = EMPTY instead.

In this case, we are looking for a member of the Roster string array that is empty. Previously the blank quotation mark notation worked, but in Blaise5 you must use **`Roster[i] = EMPTY`**. This type of notation (and its opposite, `Roster[i] <> ''`) was present throughout our instrument and was the cause of many instances of incorrect instrument routing in Blaise 5. Similarly,

```
…AND harmedFlag = 0 THEN…
```

relied on an implicit value of 0. In many instances this had to be modified to harmedFlag <> 1. In the following example, field **`LOSTPAY`** was coming on route when not expected:

```
IF (DAYSLOSTWORK <> 0) THEN
              LOSTPAY ...
```

`DAYSLOSTWORK` is off-route with an empty value. In Blaise 4, this resulted in the comparison **`DAYSLOSTWORK <> 0`** evaluated to "false" and the **`LOSTPAY`** field was skipped. But in Blaise 5, **`LOSTPAY`** was coming on route.

Modifying the syntax to **`IF (DAYSLOSTWORK > 0){changed to greater than}`** corrects the issue.

"Self-Reference"

Numerous warning messages were generated with the following message, referencing a large number of fields:

Self Reference: Field PoliceSupp of Block Type BCoreSection.BPoliceSupp is put on route with generated parameter PoliceSupp.PPCSScreener.SQ_WPOL that contains an instance of the same type.

A look inside the Blaise 4 data structure reveals that the compiler here is pointing to "generated parameters." It is true that good Blaise programming practice advises against generated parameters. Still the presence of so many warnings on the error screen results in excessive clutter, making it difficult to notice other types of warnings. Perhaps there should be away to ignore or filter certain types of warnings.

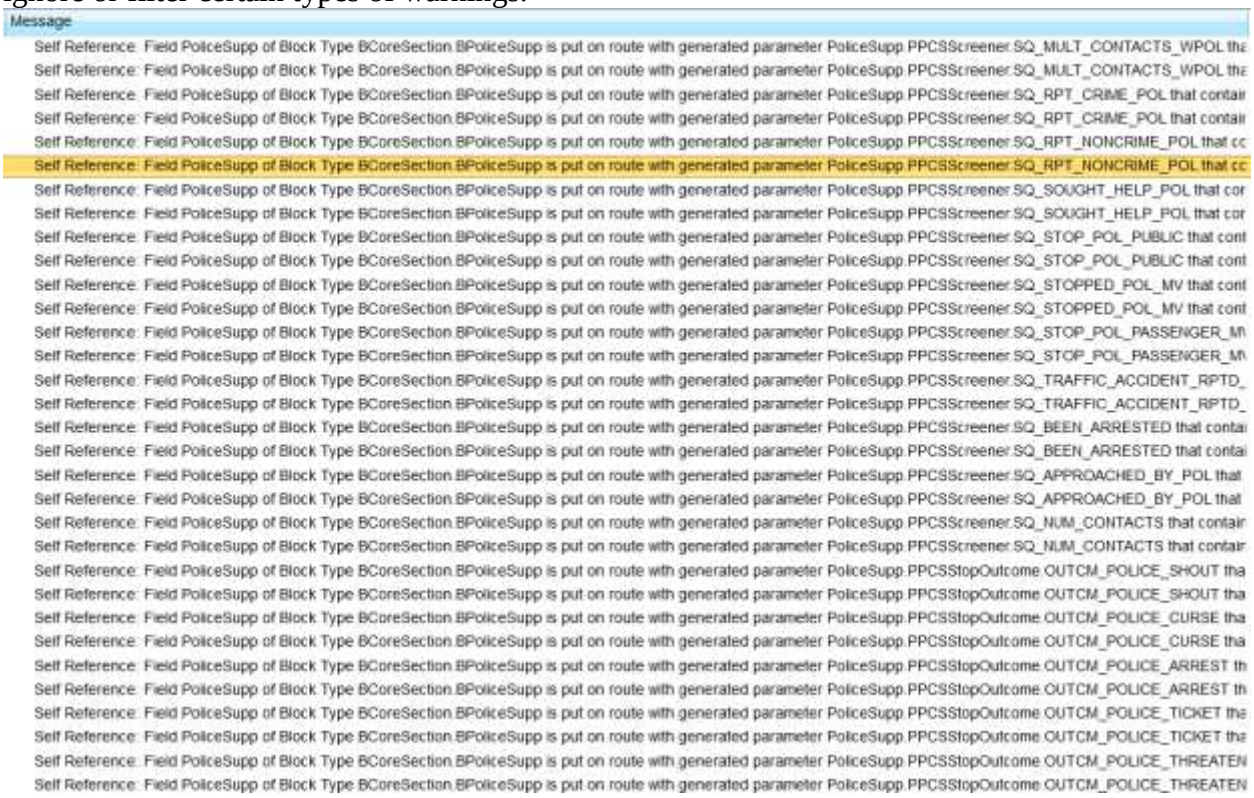

Figure 5. Self Reference warnings.

Loop indexes

Several parts of our code contained the following syntax, resulting in an error:

```
for i := 1 to 30 do
   if i = WHOTOCHANGE then
     BChange[WHOTOCHANGE](WHOTOCHANGE, changeFlag)
…..
```

ERROR: Error Line 1571 (20) - Loop variable i expected

This is another example of the Blaise 5 compilation rules being stricter. In this case, replacing the array index with "i" was a straightforward change:

```
     BChange[i](WHOTOCHANGE, changeFlag)
```

Reusing field names

In Blaise4 we had an array called "count_down[1..30]". An element of this array was passed to a sub-block. The received parameter (ungracefully) used the same name:

```
import count_down    : integer
```

This worked in Blaise 4, because the rules engine referenced the local name first, and so any reference to count_down in the sub-block was understood to be a reference to an integer, and not to the array declared at the higher level.

In Blaise 5, however, this appears to be reversed, resulting in an error message when trying to pass this integer as a parameter:

```
Sect09 ( {Import} count down)
Error - Cannot (implicitly) convert 1st parameter of type ARRAY[1..30] OF 0..30 to
type Number
```

The solution is to rename the local variable.

Emptying a block

As previously described, emptying blocks in order to bring them back on route is a critical feature of the NCVS instrument design under Blaise 4. It is the mechanism with which the instrument accomplishes multi-layered looping.

However, we discovered that assigning a block to EMPTY is not currently possible in Blaise 5. It has been promised that the functionality will be added in future releases.

This limitation also affected the ability to create a Replacement Household – to empty the household roster when all the people who lived there during the last interview wave have moved out.

To work around the lack of the block emptying functionality, we emptied just certain key fields in the beginning and end of the blocks in order to enable us to cycle through the entire household.

Layout Issues

We are still in the process of experimenting with various screen layout options under Blaise 5. We would like to continue to run our surveys in keyboard-driven environments that our interviewers are used to, while also supporting touch screen interactions.

Fonts

Blaise 4 font tags (e.g. @F, @\, @|) are automatically converted into Blaise 5 tags. For font tags, this happens if corresponding user-defined fonts are defined in the Resource Database:

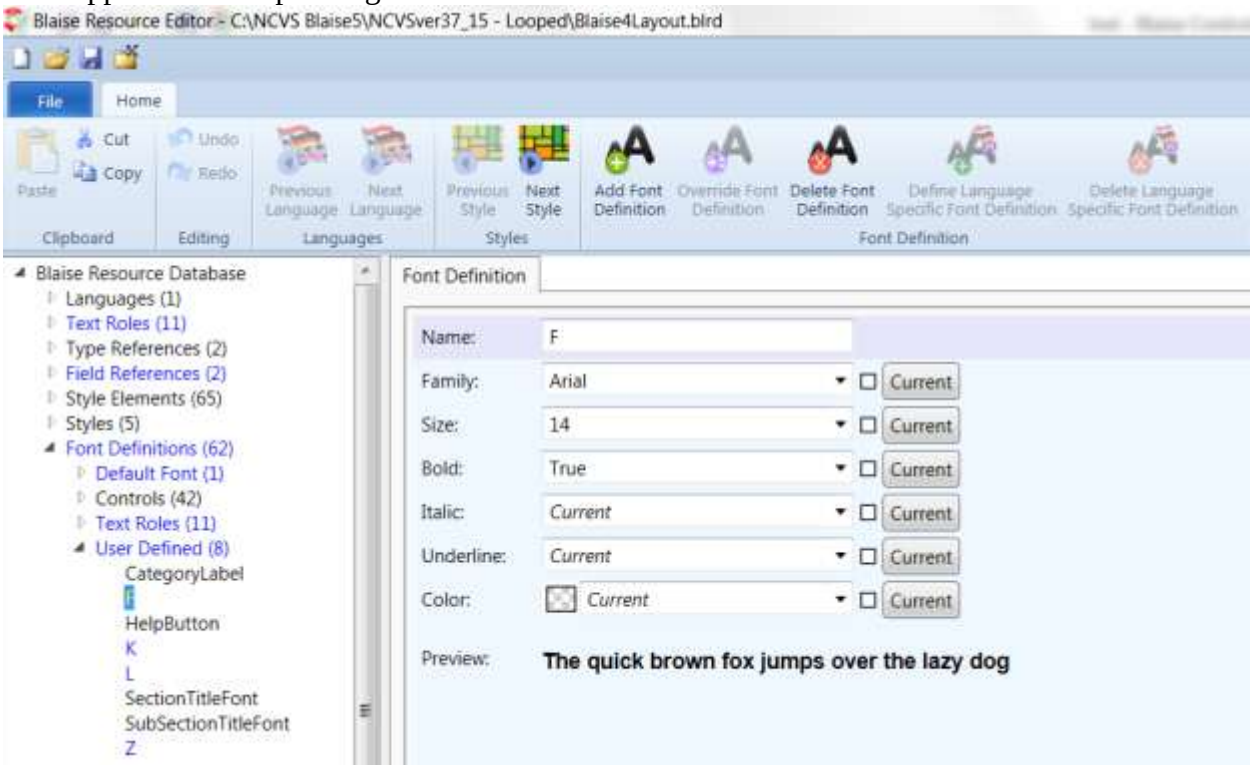

Figure 6. User Defined font definitions

In the screenshot above, you can see the font definitions that will result in tags <F>, <K>, <L>, and <Z> in our question text.
Thus, in Blaise 4, text for the field LOCATION is defined as follows:

```
LOCATION  ("LOCATION")
   "@/@|@Zs@Z@L Ask or verify:@L
@/@/@/@|@FWhere in an open area, on the street, or on public transportation
did this incident happen? ^displayRoster@F"
```

In Blaise 5, the text definition looks like this, with the @ tags converted to <> brackets:

```
LOCATION  ("LOCATION")
   "<newline><tab><newline><z>s</Z><L> Ask or verify:</L>
<newline><newline><newline><space count=4><F>Where in an open area, on the
street, or on public transportation did this incident happen?
^displayRoster</F>"
```

Fill fonts

Now, what happens if the field contains a fill and there are font tags inside of that fill? As it turns out, Blaise 5 will not interpret those font tags, and will instead display them on the screen, as seen in this example:

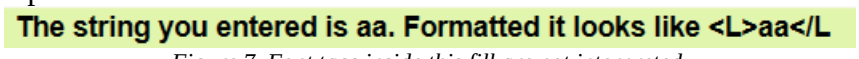

Figure 7. Font tags inside this fill are not interpreted.

To prevent text enhancement tags from appearing literally in the text, the variable text should be placed between curly brackets (e.g. ^{MyFill}). Text enhancements will then result in enhanced rich text. It is not clear what the purpose of this syntax change is because it seems to me that you almost never want to display literal font tags (and you can use escape characters should you ever need to).

Here is an example using MyFill:

```
MyField "Formatted it looks like ^{MyFill}": 0..1
```

With the inclusion of the curly brackets around the fill, the text looks like this:

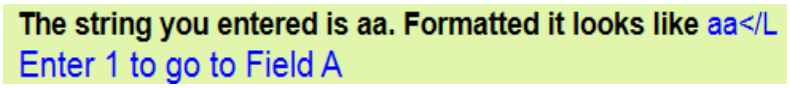

Figure 8. Fills placed inside curly brackets have their font tags interpreted.

Hopefully a future Blaise 5 release will either automatically place all fills in curly brackets, or change the default behavior of fills to interpret tags.

Of course, the fill in Figure 8 above has another issue – the closing tag is cut off and therefore not interpreted.

Fill string size

A complicating factor is that, as in Blaise 4, font tags count towards the declared string length. However, Blaise 5 tags use considerably more characters (e.g. </F> vs @F, <newspace> vs @\, etc). This renders many fill declarations insufficient, and we will need to go through all the fills and increase their string length. For example, a string **MyFill** declared as string[6] in Blaise 4 will contain 9 characters when the code is converted to Blaise 5.

```
MyFill : string[6]
..
MyFill := '<L>aa</L>'
```

This will result in the string MyFill being truncated and assigned a value of "**</L**", as seen in Figure 8. So, the length of MyFill will need to be increased in order to fit the new tag format.

In addition to the incorrect values being assigned or displayed due to this truncation, the issue will have further implications for possible input data and output data fields that may also need to accommodate the increased field length.

It is our hope that this, too, can in the future be handled by the conversion utility. Making such fixes by hand in an instrument consisting of dozens of blocks would be time-consuming.

Keyboard navigation

Unlike in Blaise 4, "out of the box" default layouts in Blaise 5 are geared toward a keyboard-free environment. However, our field representatives and telephone center operators still use laptops and desktops, even as touch-driven devices are being phased in. During the process of converting the NCVS instrument, we explored some ways in which keyboard functionality can be either replicated or emulated in Blaise 5.

Blaise4Layout

One of the samples provided with the Blaise 5 system is the Blaise4Layout resource database. This resource database includes templates that have the effect of imitating the look and feel of a Blaise 4 instrument. Here is an NCVS screen rendered using this resource database:

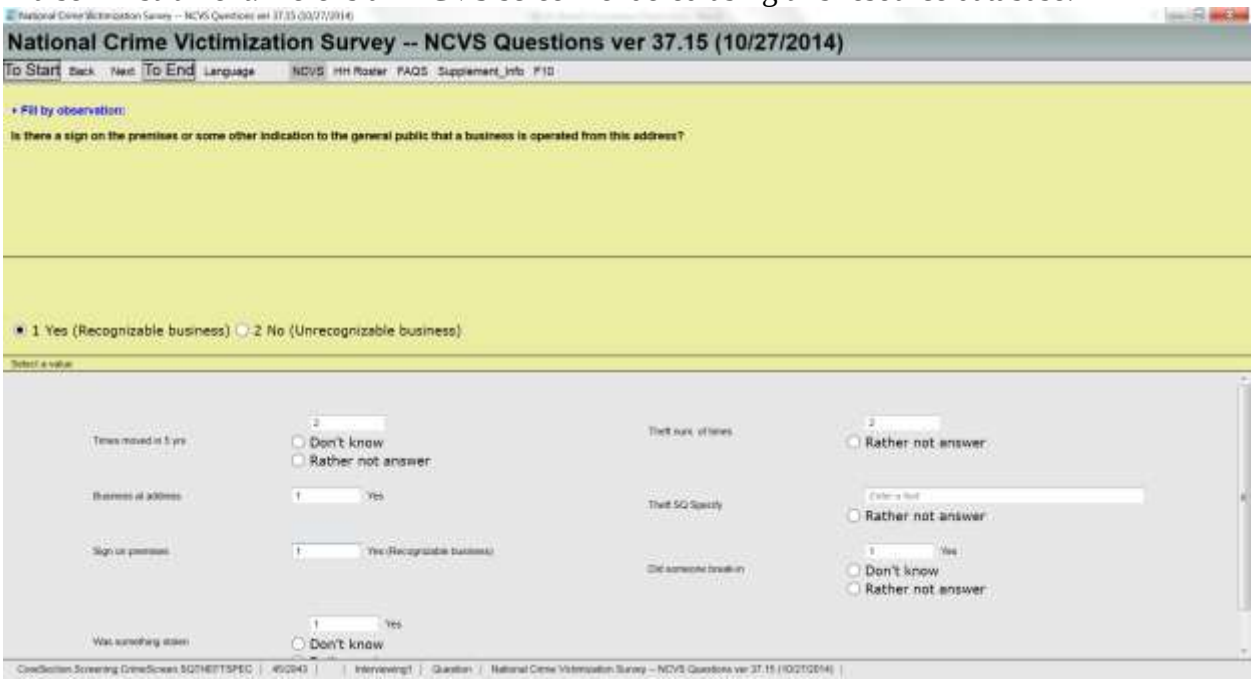

Figure 9. Using a Blaise4-like layout under Blaise 5.

Note the top control strip that includes some navigation and language selection buttons, as well as buttons for accessing parallel tabs.

This resource database employs a combination of control templates that allow for a keyboard entry of category values (see figure 10). So, Response Value Templates "SetTextBox" and "EnumerationTextBox" allow for keyboard entry of category numbers, while Category Templates "SetCodeText" and "EnumerationCodeText" display the answer category numbers next to the category text.

Figure 10. SetTextBox control

Blaise4Layout and its controls are a promising start for integrating keyboard navigation into Blaise 5 surveys, even though it has not yet been fully implemented.

Keyboard shortcuts

Other features of the Blaise 4 DEP that are still missing are the keyboard shortcuts and menu actions. The need for a menu is somewhat obviated by the ability to associate events with buttons. For example, the "Home", "End", "PageUp", and "Page Down" actions can all be associated to button events. But keyboard shortcuts for these buttons are still lacking.

Especially missed is the ability to enter Don't Know (Ctrl+D) and Refuse (Ctrl+R) responses with a keyboard, and the ability to suppress a signal and call help (F1).

In addition, radio button categories, once checked, cannot be emptied without keyboard input.

Checks and Signals

The handling of checks and signals has changed considerably in Blaise 5. In this section, we will describe some of the changes we have encountered.

Checks involving more than one field / Involving

One of the most useful features of Blaise edit checks has been the "Involving" feature, enabling the user to jump to any of the fields involved in a check in order to correct problem entries. The implementation of the INVOLVING feature in Blaise 5 is lacking. In the following screenshot, fields *Start* and *IncSAM* are "involved" in a CHECK.

Figure 11. Multiple Involved fields on one page

As the screenshot shows, this results in the error appearing next to each of the involved fields. This is okay when all of the involved fields are laid out on one page, but presents problems when involved fields are on separate pages. This is because the interviewer does not know to navigate to the previous pages to correct the error. Even if the field names were included in the error text, there is no simple way to select the involved field and jump to it. Correcting an inconsistency between multiple fields is therefore quite difficult in the current implementation of Blaise 5.

ERROR keyword

An "ERROR" keyword can be used to trigger error checks – both soft and hard – in Blaise 4. It is impossible to know just by looking at this line of Blaise 4 code whether the resultant check will be Hard or Soft:

```
ERROR "@FPlease correct your entry@F"
```

The type of check that will be displayed is determined by the presence of either the SIGNAL or CHECK keyword somewhere in the code prior to the appearance of the ERROR keyword. SIGNAL/CHECK act as a kind of toggle.

We found that in Blaise 5 the behavior of the ERROR keyword was not consistent with Blaise 4 when the SIGNAL/CHECK toggle was not near the ERROR keyword – soft errors in Blaise 4 appeared as hard errors in Blaise 5 in a number of places. Perhaps this is caused by differences in the order in which the Blaise 5 parser reads through the blocks and encounters the toggles. This type of problem can be difficult to notice during testing. It may be good advice not to use the ERROR keyword to trigger checks, unless it is immediately preceded by the toggle, as in "`SIGNAL ERROR…"`

Piping and EditType

Often it is necessary to capture the "Suppressed" value of a signal. In our survey, a suppressed value of a check at the screen ROSTERREVIEW triggers the emptying of the ROSTERREVIEW screen:

```
SIGNAL CK_HHR | (hhr_error = 0)
        if CK_HHR = Suppressed then
           ROSTERREVIEW := empty
           endif
```

In this code, CK_HHR is an EDITTYPE that, via piping, receives the value "Suppressed", and we then use this value in an *if* statement. The problem is that, in Blaise 5, this syntax gets us stuck in a loop, whereas the value of ROSTERREVIEW keeps getting emptied. A simple modification explicitly empties CK_HHR, and resolves the issue:

```
SIGNAL CK_HHR | (hhr_error = 0)
         if CK_HHR = Suppressed then
            ROSTERREVIEW := empty
            CK_HHR := empty
            endif
```

EditType error text

Additionally, in the same piece of code, *edittype* field CK_HHR contains the error text that, in Blaise 4, displays as the error message:

```
CK_HHR "Please correct this error": EDITTYPE
```

However, in Blaise 5, only the literal condition is displayed, and the text in the EDITTYPE field is ignored:

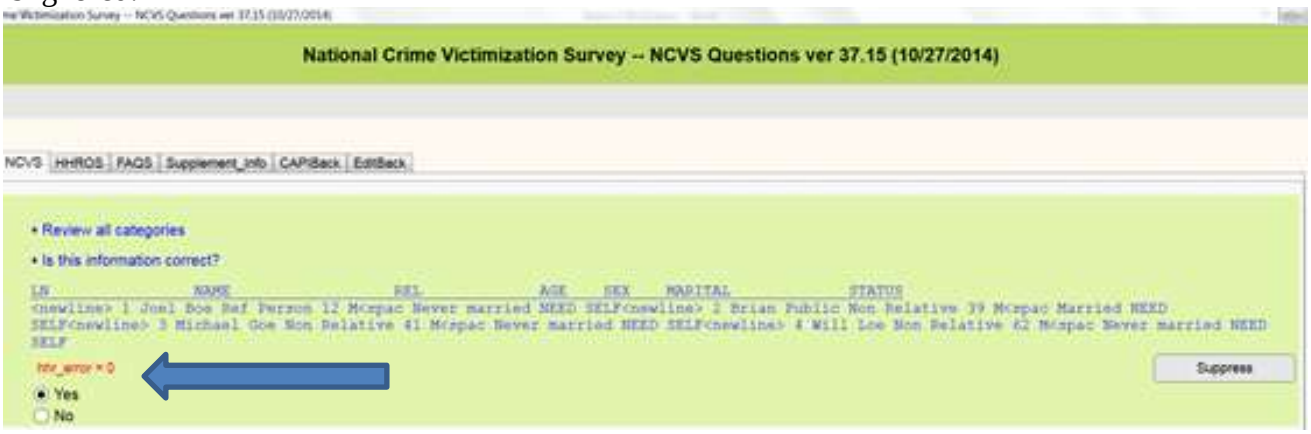

Figure 12. Error Text is not appearing correctly

As a result, we need to define the text inline, as follows:

```
SIGNAL CK_HHR | (hhr_error = 0) "MY TEXT IS DEFINED HERE"
```

Suppressed Checks

There is currently a bug where the switching of tabs "re-awakens" all of the previously suppressed checks. This is perhaps related to the following Blaise language modification:

> *When a field involved in a suppressed signal is changed by the rules after the signal was triggered, the suppress is removed.*

Parallel Tabs

Tab Text

When first converted, the parallel tab text used block names rather than descriptive titles:

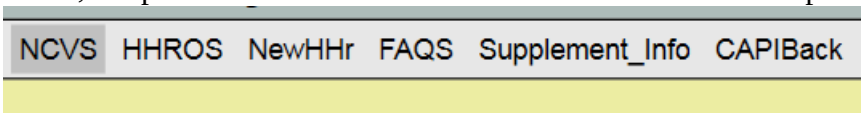

Figure 13. Parallel tab text not displaying correctly

In Blaise 4, this text is set in the Datamodel Properties > Short Text.
In Blaise 5, this needed to be done in 2 steps:

1) Specify the text in the FIELDS section of the source code:

```
HHROS    / "HH Roster"     : BlkHHROS
```

2) Then, in the Resource Editor > Master Page template > Parallel tabs control > TextSource = TextOrName.
3) (Alternatively, if the page template uses Parallel Templates, then go to Page Area > Parallel Templates > [Default] > Parallel Button > Text Source > TextOrName

This is the result:

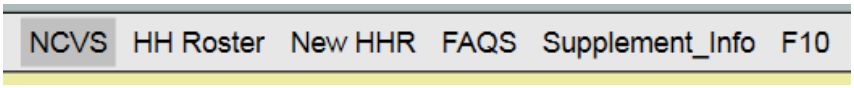

Figure 14. Corrected parallel tab text

Tab visibility

Some parallel tabs should only be visible when a certain condition is met. In our survey, we only want to show the tab "EditBack" when flag WEBEC = 1. In Blaise 4, this is accomplished with the following code:

```
IF WEBEC = 1 THEN
    EditBack
ELSE
    EditBack.KEEP
ENDIF
```

In Blaise 5, the KEEP was causing the EditBack tab to always remain visible, and the statement needed to be commented out. If the KEEP is necessary to preserve the data in that block, a different workaround will be needed, perhaps via the "Toggle Visibility" event.

Navigating at End of Parallel

There is currently no way to specify instrument behavior at the end of a parallel block. One implication of this is discussed below in the section “Exiting the DEP”.

Tables and groups

One of the most important concepts in Blaise 4 is a “TABLE”. This is going away in Blaise 5 and is being replaced by a GROUP construct. Fortunately, this change does not require very big changes to the instrument code.

The GROUP concept is described very well in the Blaise 5 Help documentation. In essence, the block defining the table rows remains unchanged, but the “shell” of the table – the code controlling the routing of the rows is no longer called a TABLE. It is renamed as a BLOCK, and a section called GROUP is added to this block. The GROUP section contains the code (typically a *for...do* loop) for displaying the table rows. The group name is then used in the RULES section to place the entire table on route.

The most difficult part of the process is applying the table template to the group, and modifying that template to accommodate the data types that are used in the table. Additional work is needed in this area to create an optimal layout for large tables with multiple data types, long question texts, and vertical as well as horizontal scrolling. Below is one implementation:

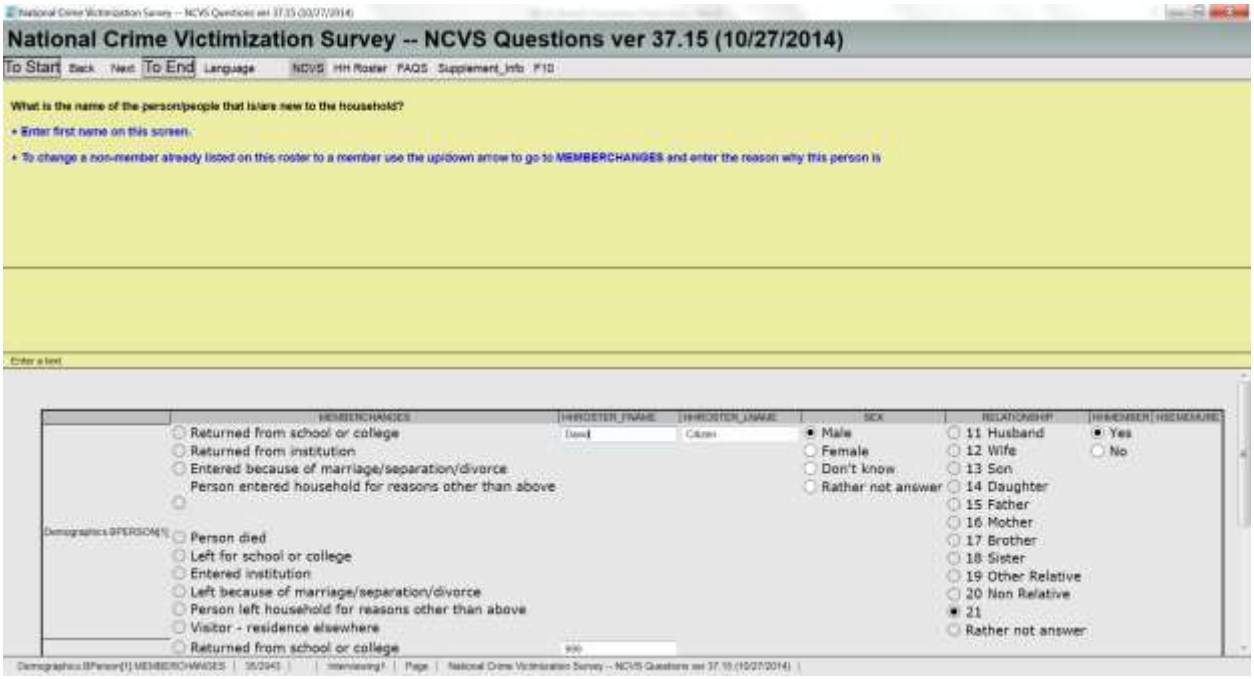

Figure 15. Attempt at a Blaise4-style table in Blaise 5.

Critical Fields / Re-execution

Critical fields vs. Server contact option

The layout view contains an option to "Generate Critical Fields". This option requests server contact as soon as a value is entered into the "critical field" allowing for the refreshing of values and routing prior to leaving the page. This option is important for web surveys, but it is not relevant for stand-alone CAPI survey applications.

Instead, a project-level setting must be changed in the project's .blax file: select the **Settings tab > Advanced > Server Contact > check "*After leaving a changed question*".**

Run Time Performance Delay

When first running the instrument, we noticed a significant re-execution delay in Blaise 5 for our large survey, with performance being inferior to the Blaise 4 instrument running side-by-side on the same machine.

Specifically, there was no delay whatsoever in navigating between most fields in Blaise 4. In Blaise 5, when running the NCVS datamodel, there was a delay of 1.5 to 2.5 seconds in updating a fill, or in changing which field comes on route based on a value entered on the same page.

This issue was improved by passing a "ThickClient" parameter at runtime. The delay went down to about half a second for fields without any rule dependencies outside their local block. The delay is longer when there are such dependencies.

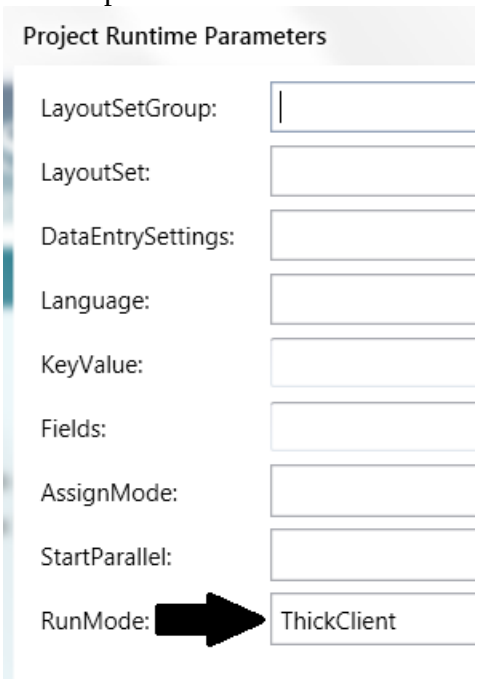

Figure 16. Runtime parameter ThickClient speeds up performance.

This was a substantial improvement, but it still does not yield the type of instantaneous smooth flow that one sees in a Blaise 4 instrument. More than just a delay in going from one screen to

another is first landing on a field that will then be taken off route when the rules engine catches up half a second later. This has the effect of being occasionally disorienting, as the active field suddenly switches or a check fires for a field that has already been passed. If the question text consists of a fill that is half-a-second late in recalculating this means the interviewer will begin reading the wrong text that will then change mid-sentence.

In summary, we believe there may be two separate issues involved. Firstly, there is a greater processing delay than in Blaise 4. Secondly, unlike in Blaise 4, the application allows the focus to progress to the next field even as the prior field is still being evaluated in the background, resulting in disorienting behavior.

Data storage and saving

Data and metadata from the open, incomplete, data entry session is continuously being saved by the Data Entry Service into a database called *RuntimeSessionData2.db*. This includes not only the values of all the fields, but also metadata such as the active field and so on. This database is not the same as the permanent instrument database. Field values are moved into the instrument database only after the last field on route in the form is reached – or via a Save action.

This session data is used to back up a partial case, and enables the user to re-enter the case in the same state in which it was last left.

If the session is terminated before the end of the survey is reached, the permanent database will not be updated.

One of our key business requirements is the ability to retrieve data from a partial case in the field. During our Blaise 4 field operations, we often handle transmissions of partial cases for various purposes, and being able to easily load these partial cases between laptops, or to view them is very important. Sessions sometimes crash mid-interview, or a laptop may experience a power failure. Sometimes a partial case may need to be re-assigned to a different interviewer. We are not sure whether it is possible to extract and copy data for an individual case from the session database, but if it is it will require a third program accessing it through the API.

In order to actually store the data from a partial case into a permanent case database, an explicit *Save* action needs to be added to instrument controls. For example, one can add the *Save* action to the OnClick event of “Next” or “Back” buttons, so that the data is saved at each page switch:

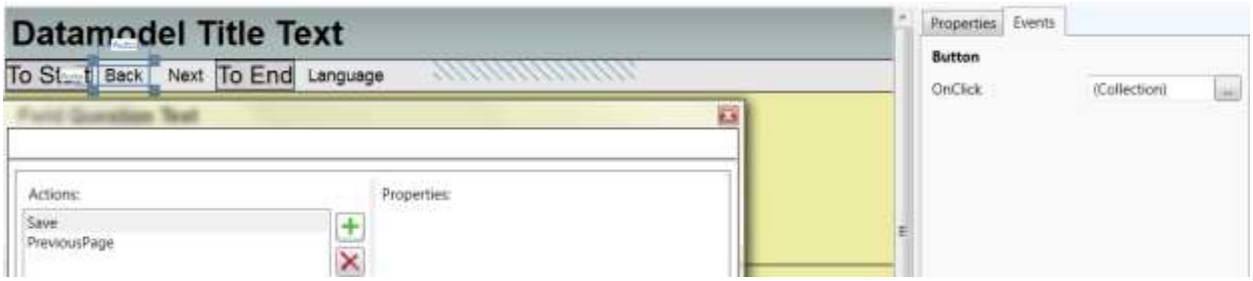

Figure 17. Adding an event to buttons

Similarly, the Save action can be added to parallel tabs, all other buttons, radiobuttons and checkboxes. It seems to us that this would be good practice, though of course it would have to be weighed against potential performance costs. It must be said that unlike in Blaise 4, I did not notice any delay in Blaise 5 associated with the Save action. It is not possible to add events to text boxes associated with integer and string fields; it is not possible to request that data be "auto-saved" based on a set time interval (a feature that we utilize in Blaise 4); and it is not possible to associate a Save event with the closing of the data entry program.

The new runtime session database with a more robust implementation of the Save action promises to be a good way to deal with partial and abruptly terminated interviews.

Exiting the DEP

Perhaps the most important functionality in this regard is preventing interviewers from exiting the instrument without first going through the "Back" parallel block in which they gracefully wrap up the interview and at the end of which the data is saved and the form is closed. It is currently not possible to disable the Windows "X" button on the DEP form in order to prevent the user from immediately exiting the survey session. This is perhaps due to the multi-platform target of Blaise 5 – different platforms have different rules for shutting down applications. Still, controlling how an application is shut down is an important element of the CAPI survey environment. It would be very useful to have the ability to disable the X, or to trap and modify the "close" event of the application.

Control Center and DEP environments

The Control Center environment is a big step forward from Blaise 4. Real-time parsing, Intellisense editor, device emulator, and the WISYWIG layout designer are welcome improvements.

The Resource Editor is a complex tool that was at first intimidating and cumbersome to use. However, once an understanding of how templates work together was attained, the editor became much easier and more intuitive to use. Thus, appropriate training is essential.

The watch window and database browser are not yet fully implemented.

Speed

The speed of the control center and layout designer has improved dramatically since earlier iterations of the Blaise 5 software. It is now practically seamless. Certain particularly intensive operations (such as generation and removal of critical fields throughout the whole instrument) still take a very long time, but on the whole the control center operations are quite smooth.

One notable issue is the solution build. When new pages need to be generated, it takes 10 to 12 minutes for our survey to generate all of the pages. By contrast, the same survey in Blaise 4 took about 10 seconds to prepare on the same machine.

```
15:34:13 :   Generating layout
15:34:13 : WARNING:    Not all field references of the
15:34:13 :   Layout generated
15:34:13 :  Generating meta file
15:34:15 :   Generating interview pages
15:35:27 :   Generated page 100
15:35:39 :   Generated page 200
15:35:58 :   Generated page 300
15:36:19 :   Generated page 400
15:36:41 :   Generated page 500
15:37:06 :   Generated page 600
15:37:29 :   Generated page 700
15:37:50 :   Generated page 800
15:38:13 :   Generated page 900
15:38:36 :   Generated page 1000
15:38:57 :   Generated page 1100
15:39:19 :   Generated page 1200
15:39:40 :   Generated page 1300
15:40:01 :   Generated page 1400
15:40:22 :   Generated page 1500
15:40:44 :   Generated page 1600
15:41:07 :   Generated page 1700
15:41:33 :   Generated page 1800
15:42:00 :   Generated page 1900
15:42:26 :   Generated page 2000
15:42:52 :   Generated page 2100
15:43:17 :   Generated page 2200
15:43:43 :   Generated page 2300
15:44:10 :   Generated page 2400
15:44:36 :   Generated page 2500
15:45:02 :   Generated page 2600
15:45:24 :   Generated page 2700
15:45:47 :   Generated page 2800
15:46:11 :   Generated page 2900
15:46:26 :   Generated page 3000
15:46:27 :   Generated 3011 pages
15:46:27 :   Generating datamodel definition
15:46:27 :   Generating check definitions
15:46:27 :   Generating classification types
15:46:27 :   Generating custom pages
15:46:29 :   Meta file generated
15:46:29 : Build successfully completed (217 warnings
```

Figure 18. Generating pages during a Build takes 12 minutes.

There is an important option in the control center to "*Build without generating pages*" -- this means that pages will be generated at runtime. You can speed up the build process even more by un-checking the option to *GenerateAllSections* in the "Properties" menu of your blax's Layout view. (However, if you have layouts that apply to specific sections, such as blocks or groups, then you need to set this option to "True" in order to generate accurate section indexes for your Layout tasks.) It seems that if you are not working on section layouts you can uncheck this option.

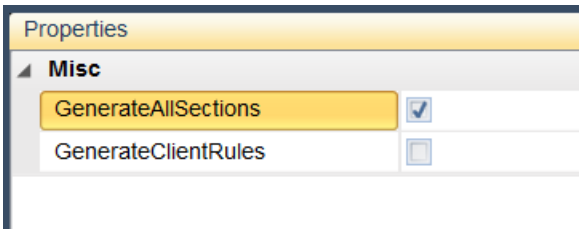

Figure 19. This option generates Section Indexes.

Blaise documentation warns of a run-time performance cost to this build option. We have not noticed the cost penalty thus far in our CAPI standalone environment, and it certainly makes the build process much more manageable. If we do start noticing a cost penalty, then we may decide to use the option during development only, but do a full build with pages when preparing for a release.

System Errors

During our testing we sporadically encountered the following errors in the DEP. It is not clear what exactly triggers these errors. More information about the source of the errors – and perhaps a facility to report the errors and the call stack to the developers would be helpful.

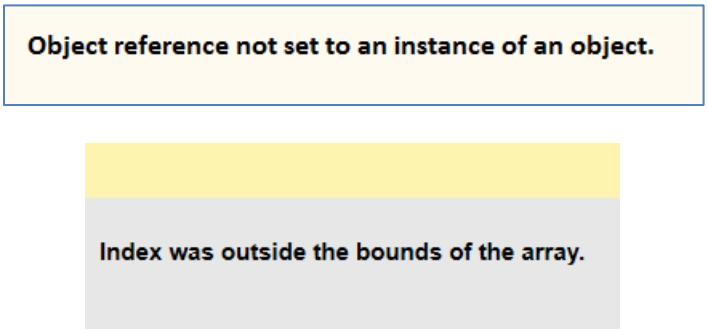

Figure 20. DEP system errors

Additionally, when a data entry session is left inactive for too long – about 30 minutes? -- the following error screen appears at the next page switch, and the session is automatically terminated:

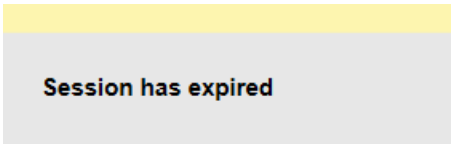

Figure 21. "Session has expired" error message in the DEP

Terminating a session due to inactivity is a questionable way to end an interview session and can potentially be a source of interviewer frustration and data loss. If there are environments under which a session expiration limit is preferred (such as due to a security requirement in a web

survey), then perhaps there should be a survey-level option to disable the expiration timer in other environments, or at least to save the data in the permanent database before terminating the session.

Item-level Help screens

Item-level Help screens were accessed in Blaise 4 via an F1 function key. In Blaise 5, item-level help can be accessed by defining a Help Role for a field, and attaching an event to a Help button.

First one needs to define a "Help" role for the datamodel, and map it to the Help Role defined in the Layout Designer:

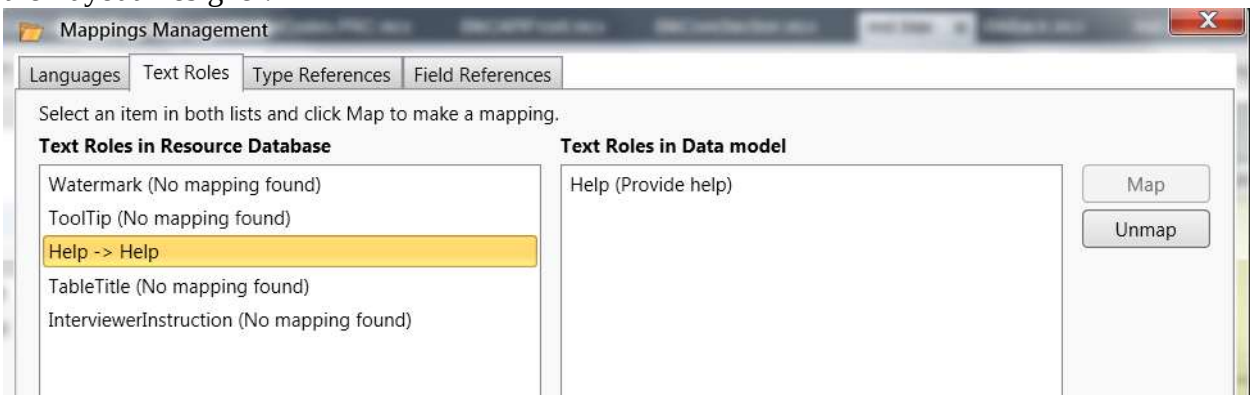

Figure 22. Mapping the Help Role

Then, the HELP role tag is defined for fields which require item-level help:

```
START_CP ("START_CP")
     ENG "<newline><L>  Some question text….</L>"
     HELP "Help text here..."/ "Start CAPI Interview" : TStart_CP, NODK,
NORF, NOEMPTY
```

Lastly, an event is attached to the "Help button" in the Resource Database. That event can be a *GoToURI* , such as *file:///C:\NCVS Blaise5\NCVSver37_15\FunctionKeys.htm*

The Help Button has a regular expression associated with its Visibility property – it is only visible on fields for which a Help Role is defined:

```
IF LEN(Field.GetRoleText('Help')) = 0 THEN 'Collapsed' ELSE 'Visible' ENDIF
```

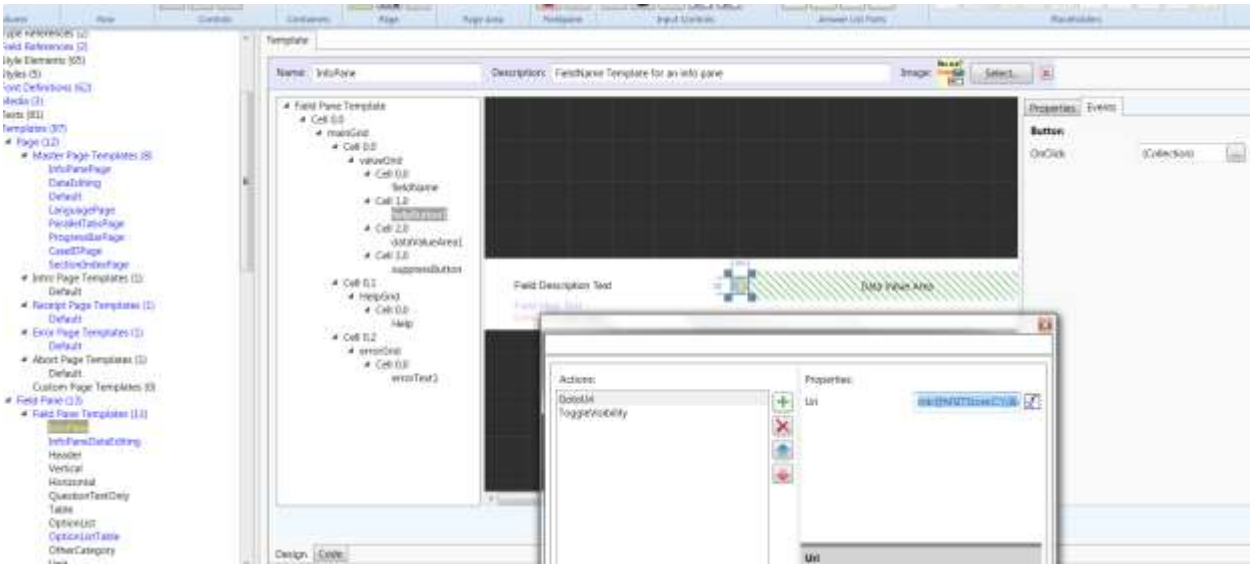

Figure 23. GoToURI action

To access a page in compiled HTML help (.CHM), one can use the following syntax:

```
mk:@MSITStore:C:\\Blaise4
surveys\\ACS\\m201501\\help\\ACS_WITH_TOC.chm::/H_WKL_PRS.htm
```

This will bring open a new window with the HTML Help screen.

Now we can create a Regular Expression so that item-specific help pages can be brought up. In our case, help pages are named in a standardized format ***H_<field name>.htm***. Thus, the regular expression can look like this:

```
'mk:@MSITStore:C:\\NCVS\\m201501\\help\\NCVS_WITH_TOC.chm::/H_' +
                    Field.LocalName + '.htm'
```

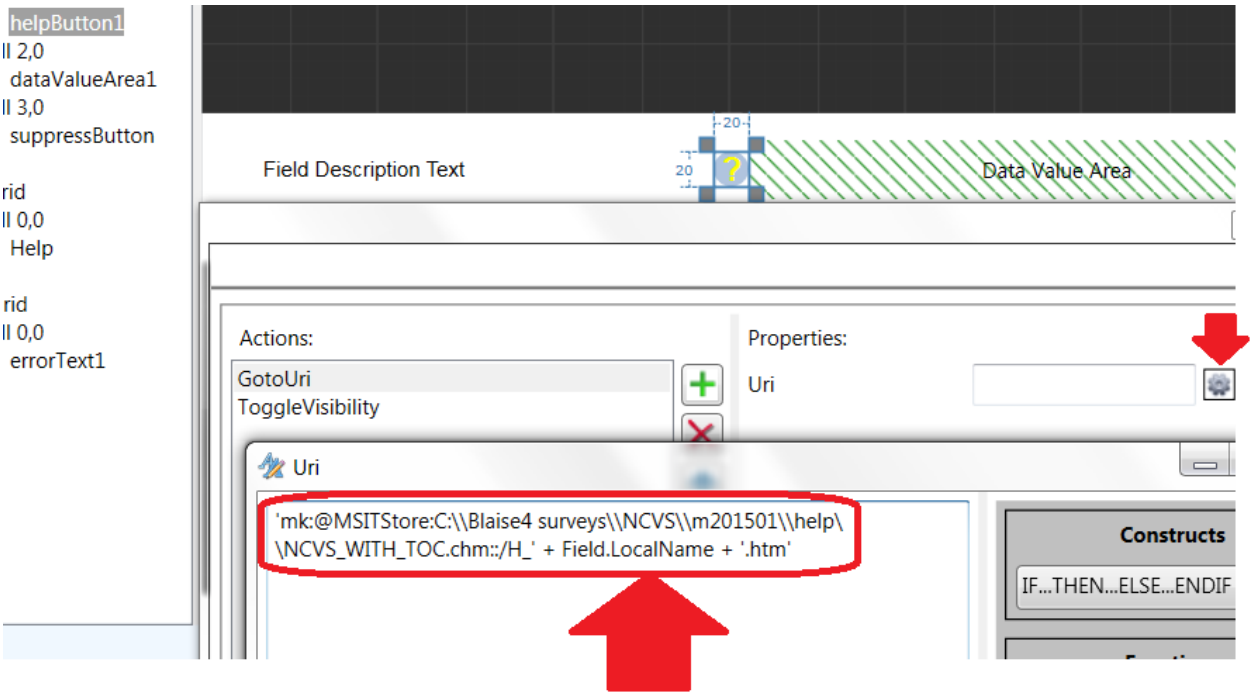

Figure 24. Expression to create dynamic item-level help based on the name of the current field.

As a result, the help screen opens in a new browser window, as illustrated below.

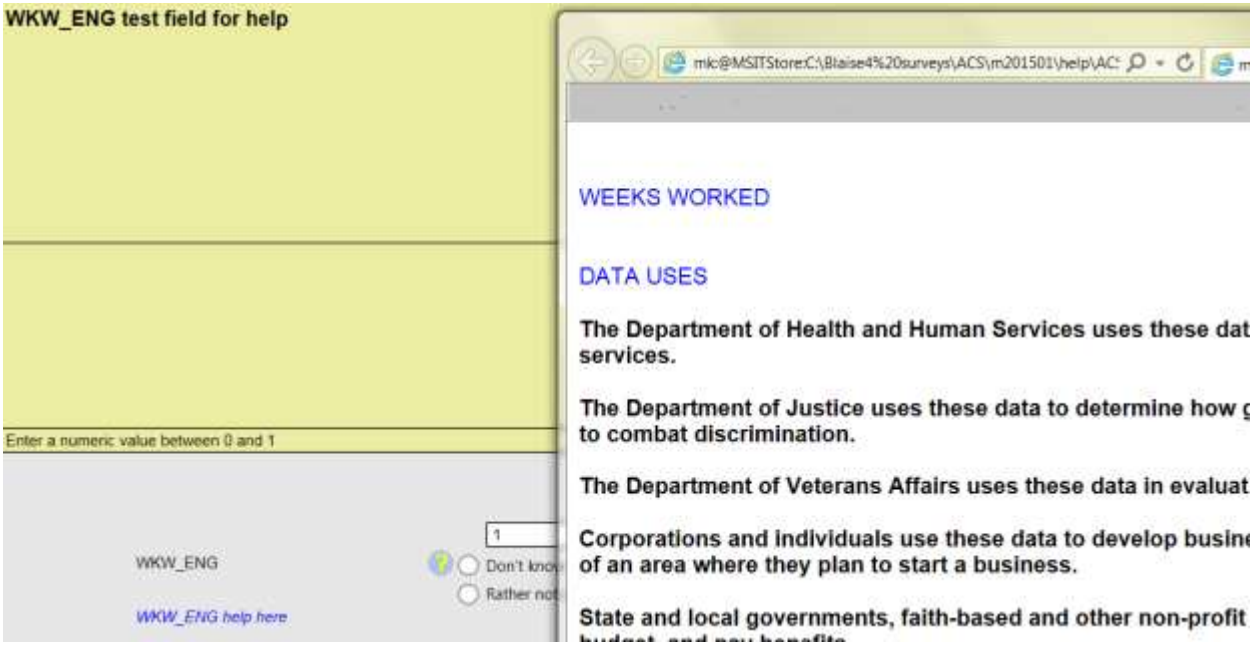

Figure 25. Item-specific help displayed in a browser.

Data Values

NCVS uses a number of storage fields that are never asked or displayed, but that retain important calculated data that is used either to route fields during the interview, or as output. During our testing we keyed identical complete cases in Blaise 4 and Blaise 5, and then compared values of 20 such fields. All of the values matched. In addition, we tested each of the possible outcome codes (i.e. codes for a complete interview, partial interview, sufficient partial interview, refusal, no such address, etc.), and in all cases Blaise 5 data matched Blaise 4 – all of the outcome codes were achieved.

Audit trail access

Audit trail data is stored in a SQLite database called *AuditTrailData.db* – a database shared across all installed surveys. This is different from Blaise 4, where audit trails are stored in a plain text file, and a separate audit trail file is associated with its own case. As such, one cannot view the audit trail without an SQLite browser, and even inside the SQLite browser it is difficult to inspect the content of the audit trail.

Here is a screenshot of the audit trail data inside a third-party SQLite browser:

	SessionId	InstrumentId	TimeStamp	Content
	Filter	Filter	Filter	Filter
1	{662c1384-f...	{bf4b0c63-...	2014-12-24 1...	<StartSessionEvent Width="1536" Height="864" Device="WindowsDesktop" Language="ENG" Platform="Windows" />
2	{662c1384-f...	{bf4b0c63-...	2014-12-24 1...	<UpdatePageEvent LayoutSetName="Interviewing1" ParallelName="PRIMARY" PageIndex="1" />
3	{662c1384-f...	{bf4b0c63-...	2014-12-24 1...	<EnterFieldEvent FieldName="CaseID" AnswerStatus="Empty" />
4	{662c1384-f...	{bf4b0c63-...	2014-12-24 1...	<KeyboardEvent KeyStrokes="00000004{BACK}1" />
5	{662c1384-f...	{bf4b0c63-...	2014-12-24 1...	<LeaveFieldEvent FieldName="CaseID" Value="00000001" AnswerStatus="Response" />
6	{662c1384-f...	{bf4b0c63-...	2014-12-24 1...	<ActionEvent Action="Save()" />
7	{662c1384-f...	{bf4b0c63-...	2014-12-24 1...	<EndQuestionnaireEvent />

Figure 26. Content of AuditTrailData.db EventData table

This view includes all sessions from all cases from all installed instruments. A separate API utility will most likely need to be written to enable case-level access to audit trails.

Effort

Initial conversion of a large instrument from Blaise 4 to Blaise 5 involves significant effort. This effort will certainly be much reduced over time, as the organization settles on standard approaches to Blaise 5 layout, deployment, data storage and transmission; as programmers become more familiar with working on Blaise 5; and as Blaise 5 itself matures.

Still, thorough testing of all aspects of instrument functionality is required, and one should not assume that the converted instrument will perform as it did under Blaise 4.

At present time, we have been able to convert into Blaise 5 much – but not all – of the functionality of our Blaise 4 NCVS instrument. As mentioned earlier, emptying of blocks is a key feature for NCVS and it is not yet implemented. This limitation, together with the lack of Manipula implementation, keeps the Blaise 5 NCVS instrument from being fully functional.

In addition, we still need to find the optimal layout approach that takes advantage of the new touch-driven technology while also retaining the functionality achieved in Blaise 4 via keyboard shortcuts and menus. We also need to better understand how the new Blaise data storage approach can work with our data transmission processes.

We still have a ways to go in order to put this all together.

Summary

Blaise 5 promises to be a robust data collection system. It builds upon the capabilities of Blaise 4 with a much enhanced, modern Control Center, multi-platform capabilities, and a complex layout design capabilities.

Perhaps the most significant area of concern is the performance speed in the DEP. Processing speed will need to improve in order to smoothly run large household surveys under Blaise 5.

Key positives and negatives we encountered during this conversion exercise are summarized in the table below.

Features that convert well	Features that need work
Overall Blaise Rules performance	CAPI DEP Performance speed
Help	Graceful exit from a case
Parallel tabs	Watch Window
Control Center environment	Viewing Audit Trails
Tables / Groups	Handling of checks
Fonts	Keyboard-driven navigation / DK / RF

Table 1. Summary of impressions

This paper focused primarily on replicating Blaise 4 functionality under Blaise 5, and we noted a number of features that may be lacking at this stage in development. Left unexplored are many new features and capabilities of Blaise 5 that may offer entirely new possibilities for future survey design.

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