Migrating from Blaise 4 to Blaise 5 – The AHS experience


1. Introduction

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

2. Overview of Blaise 4 Instrument – AHS

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

3. Conversion of AHS to Blaise 5

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

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

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

Another requirement given was minimizing the changes in the instrument; if possible, retain the structure of the instrument and the functionality for which it was built. This particular requirement was difficult to retain since B5 is a different system and the presentation in B4 may no longer be possible in B5; therefore, we had to be careful and opt for some new solutions or alternatives.
3.2 Blaise 4 to Blaise 5 Conversion process

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

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

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

3.3 Using the Conversion Tool provided in Blaise 5

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

- In the Data model:
  - **Warnings** - We experienced more than 300 warnings during the conversion process.
    - Not all field references of the Resource Database are mapped to Fields of the Datamodel.
    - The meaning of this expression differs from Blaise 4,
      - Self-Reference warnings.
      - KEEP statements were generated
      - String constant exceeding assigned variables
o **Lots of layout removal** (B4 legacy layout). In earlier version of Blaise 5, a warning message was provided that pointed out that LAYOUT statements would eventually have to be removed. In later versions, it transitioned from a warning to an error, forcing us to go through dozens of modules to remove layout code before a clean compile could be achieved.

o **Adjustments to the external files used in B4**, that is rebuild them again.

- In the Manipula scripts:
  While all the scripts were converted, the Setup script was the most important script to convert so that the input data could be loaded and generate the case databases. Some adjustments were made to this script in order to load the cases as in B4.
  o Early in this migration we received the error “Element 'leftHandeSize' was not found. Line 1, position 1” and the number 385917. It was temporarily commented out. These errors had to do with SET OF questions. Later versions of B5 corrected this issue and the code was reinstated.

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

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

### 3.4.1 Problematic Blaise 4 Code

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

- **Redundancy in the code**, repetitive code through the instrument. The use of procedures to simplify the code is always encouraged, however not always applied.

- **Different styles in the code**, best programming practices had not been enforced. This resulted in convoluted code which was very hard to maintain and convert. Contributing to this is that the instrument used for our research is 14 years old and at least 6 programmers worked on this over the years, who each brought in a different level of programming best practices and Blaise expertise.

- **Deviates from Standards** - Instrument did not follow some of the standard collection practices used by many other surveys, disregarding a good solid structure of the instrument. For example, format of phone number or method to collect address information was done differently.

- **Data Processing for output within Instrument** - Many Sponsors have the tendency to request that the instrument massage data to help simplify their output process. Not only does the instrument collect data, it also calculates variables for output. This “data processing” part of the B4 instrument added complexity to the conversion process to create a B5 instrument that would perform exactly like the B4 instrument.

- **Parameterization** - Another item found during this process was the fact that there was no parameterization across the blocks in the AHS instrument; consequently, it was hard to dissect all
pieces for analysis during module testing, when issues were identified, such as fills not showing up or skipping variables.

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

### 3.4.2 Coding Modification Suggestions when migrating to B5

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>From this in the rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IF</strong> (No_prior_survey_interv_experience IN PRIOR_SIPP) THEN</td>
</tr>
<tr>
<td><strong>ERROR INVOLVING</strong>(PRIOR_SIPP)&quot;&lt;newline&gt;&lt;Z&gt;s&lt;/Z&gt;&lt;L&gt;&quot;&quot;No Prior Survey Interviewing Experience' cannot be selected with any other answer category. Please go back and correct your answer. &lt;/L&gt;&quot;</td>
</tr>
<tr>
<td><strong>ENDIF</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Add a Check Section and...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHECKS</strong></td>
</tr>
<tr>
<td>CK_PRIOR_SIPP &quot;&lt;/R&gt; [Err 01] 'No Prior Survey Interviewing Experience' cannot be selected with any other answer category. Please go back and correct your answer. &lt;/R&gt;&quot; =</td>
</tr>
<tr>
<td><strong>NOT</strong> (No_prior_survey_interv_experience IN PRIOR_SIPP) <strong>INVOLVING</strong> (PRIOR_SIPP)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In the Rules :</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IF</strong> PRIOR_SIPP,CARDINAL &gt; 1 THEN</td>
</tr>
<tr>
<td>CK_PRIOR_SIPP</td>
</tr>
<tr>
<td><strong>ENDIF</strong></td>
</tr>
</tbody>
</table>
The need of parametrization across blocks moving into B5, is important to isolate issues during modular testing. In addition, it adds simplicity for maintainability. Furthermore, we noticed that when “reaching” for variables across the instrument, the information needed is not always available right away.

The use of functions and procedures are strongly encouraged, as their use simplifies the code for readability and maintainability.

Decline any data output manipulation in the instruments within B5; especially if these create performance issue. However, these may be ok for small things, such as:

- Setting a value based on a condition

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

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

### 3.4.3 Utilizing Blaise 5 Functions

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

Figure 3.4.1 – Example of function created in Blaise 5

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

How it is used in the code: 

```plaintext
aGenderFill := fGenderToNSStringFormat (Gender.ord)
```
3.5 LAYOUT Modification Required in Blaise 5

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

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

3.5.1 Master template

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

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

  Figure 3.5.1.1 – Example of Header created for AHS Blaise 5

- **Buttons** - We experimented with different styles of buttons. We anticipate this is one area that will be addressed when we establish our Standard Instrument Master Template, in order to have a consistent look and feel for the instruments between projects (see Figure 3.5.1.2).

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

- **Bottom Status Bar** – Our Blaise 4 standards displays a status bar at the bottom of the screen. At a minimum, the Case ID (primary Key) and Field Name are contained in this area, along with project specific fields. We added this to the Master Template for consistency (see Figure 3.5.1.2).

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

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

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

Figure 3.5.2.1 – Example – Phone Masking for AHS Blaise 5
However, in AHS and in B4, phone numbers are collected using separate fields or into a single field. Depending in the questions, extension or phone type were necessary, but not always: (Area+Prefix+Suffix+Extension+Phonetype).

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

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

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

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

Figure 3.5.2.2 – Example – Watermark example for Phone Masking

*Source Code:*

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

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

RULES
Phone_Area
Phone_Prefix
Phone_Suffix
NOPHONEEXT
NPHONETYPE

ENDGROUP
```

*Resulting*

- The WATERMARK role provides a hint to the user as to what input elements are used for.
3.5.3 Mailing & Physical Address Templates

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

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

Figure 3.5.3.1 – Example – Blaise 5 vs. Blaise 4 address templates
The team created another template for the same purpose but this time the intention is to display both addresses on one screen and allow editing of the address that is not correct. We still needed to see the address that was correct as a show and gray background. The advantage of this template was to have both information side by side (see Figure 3.5.3.2).

Figure 3.5.3.2 – Another example of Blaise 5 address template
3.5.4 Tables and Groups Templates

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

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

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

<table>
<thead>
<tr>
<th>One way in B4</th>
<th>Another way in B4</th>
<th>New approach in B5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Block</strong> bRoster</td>
<td><strong>Table tblRoster</strong></td>
<td><strong>Block</strong> bRoster</td>
</tr>
<tr>
<td>…</td>
<td><strong>Block</strong> bRoster</td>
<td>…</td>
</tr>
<tr>
<td><strong>Endblock</strong></td>
<td><strong>Block</strong> bRoster</td>
<td><strong>Endblock</strong></td>
</tr>
<tr>
<td><strong>Table tblRoster</strong></td>
<td><strong>Table tblRoster</strong></td>
<td><strong>Group grpRoster</strong></td>
</tr>
<tr>
<td>Roster: array [] of BRoster</td>
<td>Roster: array [] of BRoster</td>
<td>Roster: array [] of BRoster</td>
</tr>
<tr>
<td><strong>EndTable</strong></td>
<td><strong>EndTable</strong></td>
<td><strong>Endgroup</strong></td>
</tr>
</tbody>
</table>

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

Figure 3.5.4.2 – Example of templates created under Blaise 5

<table>
<thead>
<tr>
<th>B5 Code Without Changes</th>
<th>Result is All questions on route (Only one question should be)</th>
</tr>
</thead>
</table>
| IF IPHONENUM = EMPTY THEN TELHH = YES THEN TELAV.ASK ENDIF IF TELHH = YES OR IPHONENUM <> EMPTY THEN TELAV.ASK ENDIF IF TELHH = NO OR TELHH = NONRESPONSE THEN TELAV.ASK ENDIF … more code | IF we need to contact you by telephone to followup on missing data, is there a telephone in this home?  
Yes
No
If there is a land-line telephone in this home?  
Yes
No
Do not count cellular phones, or any phone line that is used only for a computer or fax machine.  
Yes
No
If we need to contact you, is a telephone interview acceptable?  
Yes
No
Is there a telephone elsewhere on which people in this household can be contacted?  
Yes
No
Where is the phone located?  
basement
How many adults (age 18 and over) in this household have a cell phone for personal use?  
[ ] |
3.5.5 Buttons vs Radio buttons

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

Figure 3.5.5.1 –Example of buttons inside group table
3.5.6 Dropdown list

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

Figure 3.5.6.1 – Dropdown table

3.5.7 Dropdown list for SET OF

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

Figure 3.5.7.1 – Example Dropdown for SET OF
4. Challenges Encountered

4.1 Manipula Issues

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

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

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

4.2 Blaise 5 Bugs

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

4.3 Blaise 5 and In-House App

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

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

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

Independently, we tried using the Visual studio 2017 community, we were able to build the solution, however we still encountered some issues. The picture below shows the error messages while building the project (see Figure 4.3.1).
The errors above seem to be related to an incompatible Blaise API “statNet.Blaise.API.Adroid.dll” and the “schemas.android.com” link referenced in some of “.axml” files in the resources/layout folder. An attempt to debug the errors, we revealed that:

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

We hope to continue experimenting with the Blaise 5 app.

### 4.4 Exporting the Audit Trail Database to XML

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

Our recommended SQL statement:

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

In the Audit Trail Setting, select "Answer" level to provide the level of granularity that includes the values entered by the user.
When one looks at the "Content" field where data is entered in XML format, there are UNBALANCED QUOTES, e.g.;

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

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

The proper format for the attribute LayoutSetName should be:
- **LayoutSetName="FRLaptop"** and not
- **LayoutSetName=FRLaptop**" (a quote should be before the "F" in FRLaptop)

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

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

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

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

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

### 4.5 Dealing with Alien Routers and Services

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

- **Checking the HP Services using a .NET solution.**
  
  //In the "HPServices" .NET solution, this works fine.

  ```c
  PROCEDURE LockSection
  PARAMETERS
    IMPORT value: INTEGER
    EXPORT lockId: INTEGER
    ALIEN('http://localhost:8733/Design_Time_Addresses/WCF/LockSectionService', 'ILockSection', 'LockSection')
  ENDPROCEDURE
  ```

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

- **Function Overload** - The service has a function overload "decimal LockSection(decimal importValue)" and "decimal LockSection(decimal importValOne, decimal importValueTwo)". Note that both have different attribute names, the first overload is "[OperationContract(Name="LockSection")]") while the second one is "[OperationContract(Name = "LockSection_Two")]."
Blaise either doesn't call it (because it's a function overload), or it can't process the service's return value. This is not a problem in .NET consumers though.

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

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

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

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

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

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

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

- Alien Router #1 works fine, but Alien Router #2 does not. Does Blaise 5 have a problem with overloaded functions? A C# test project for the Service functions as expected and can handle the overloaded functions, so this doesn't make sense.

- NOTE: The Blaise 5 IDE also shows a blue squiggly line underneath the "LockSection_Two" in the line "PROCEDURE LockSection_Two". We are unsure what does this blue line mean.
5. Next Steps / Future Plans

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

6. Conclusion

6.1 Achievements

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

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

6.2 Blaise 5 Learning Curve

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

6.3 Convert or Rebuild from Scratch?

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

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

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