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

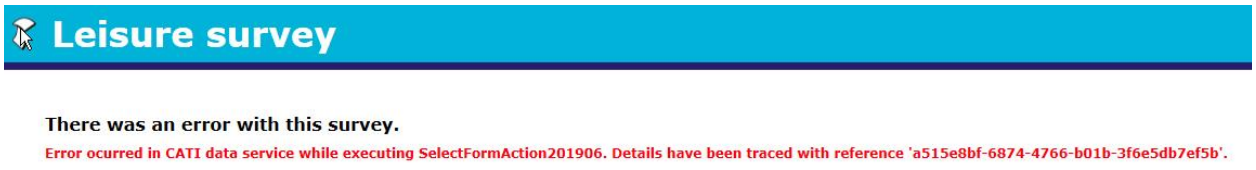
Departmental regulations have taken ways many of our rights and abilities to do our own installation. We download the latest version of the software and have the latest build installed by our IT service desk on our testing server and on a developer's individual virtual machine. By internal departmental 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.



Leisure survey

There was an error with this survey.

Error ocurred in CATI data service while executing SelectFormAction201906. Details have been traced with reference 'a515e8bf-6874-4766-b01b-3f6e5db7ef5b'.

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”.

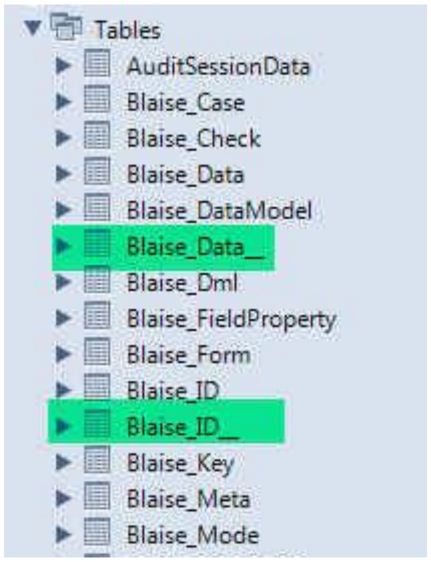
6.2.1 Harmful Changes – Our Data is Gone

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.

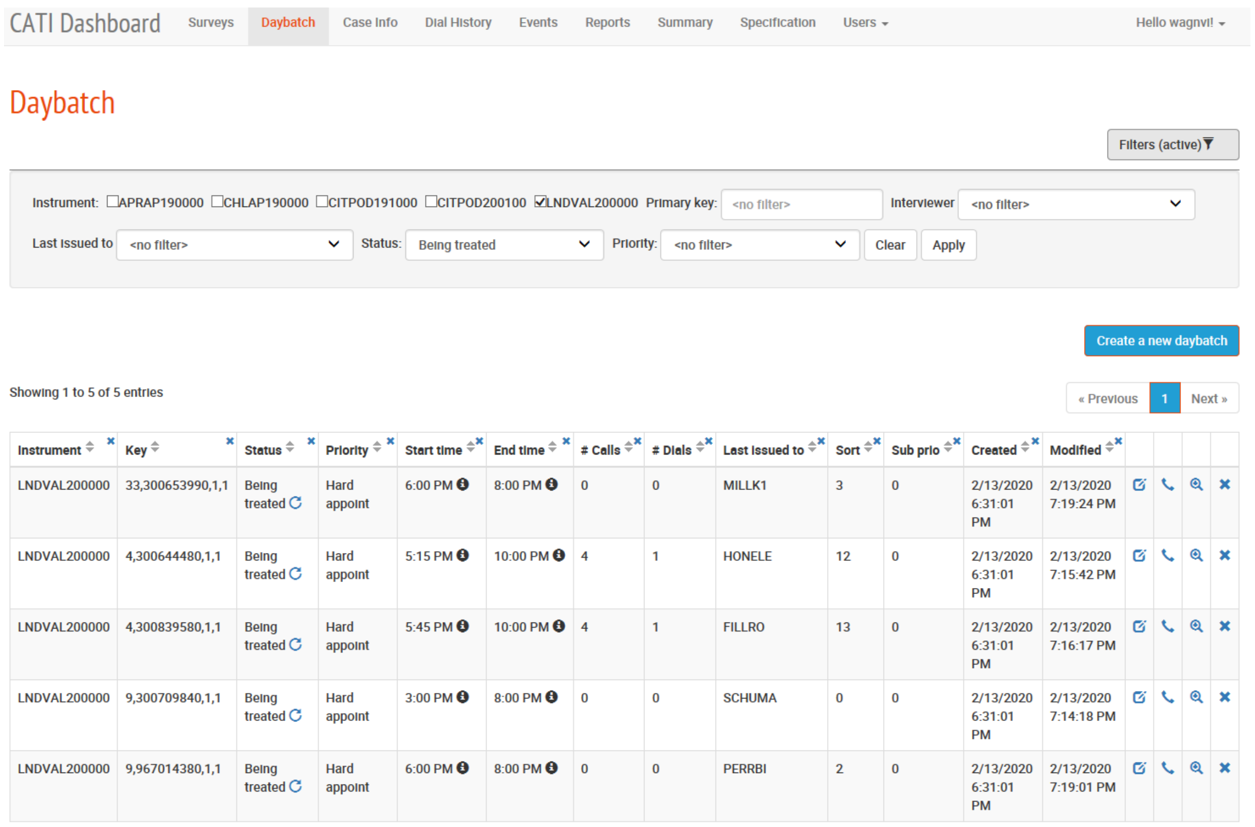
It may be desirable to see all information for all installed CATI projects on the same dashboard. Finding the particular sets of records you would like to explore can typically be done through the available filters that only differ slightly from tab to tab. However, if a poor development choice of the 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.



Daybatch

Instrument: ☐APRAP190000 ☐CHLAP190000 ☐CITPOD191000 ☐CITPOD200100 ☑LNDVAL200000 Primary key: <no filter> Interviewer <no filter>

Last issued to <no filter> Status: Being treated Priority: <no filter>

Showing 1 to 5 of 5 entries

Instrument	Key	Status	Priority	Start time	End time	# Calls	# Dials	Last issued to	Sort	Sub prio	Created	Modified
LNDVAL200000	33,300653990,1,1	Being treated	Hard appoint	6:00 PM	8:00 PM	0	0	MILLK1	3	0	2/13/2020 6:31:01 PM	2/13/2020 7:19:24 PM
LNDVAL200000	4,300644480,1,1	Being treated	Hard appoint	5:15 PM	10:00 PM	4	1	HONELE	12	0	2/13/2020 6:31:01 PM	2/13/2020 7:15:42 PM
LNDVAL200000	4,300839580,1,1	Being treated	Hard appoint	5:45 PM	10:00 PM	4	1	FILLRO	13	0	2/13/2020 6:31:01 PM	2/13/2020 7:16:17 PM
LNDVAL200000	9,300709840,1,1	Being treated	Hard appoint	3:00 PM	8:00 PM	0	0	SCHUMA	0	0	2/13/2020 6:31:01 PM	2/13/2020 7:14:18 PM
LNDVAL200000	9,967014380,1,1	Being treated	Hard appoint	6:00 PM	8:00 PM	0	0	PERRBI	2	0	2/13/2020 6:31:01 PM	2/13/2020 7:19:01 PM

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.