Replaying Blaise Audit Trail Files for Data Verification

Jason Langfahl, Survey Research Organization Technical Services Group, Institute for Social Research, University of Michigan

1. Introduction

In survey research, a lot of effort goes into ensuring the quality of the data that is gathered. After all, it is the data that researchers use and refer to when writing up papers for grants, proposals, theories, and a host of other technical documents for their fields. If the data is of poor quality, then the foundation on which their work is based is also of poor quality by extension.

This paper will concern itself with a quality assurance utility that was devised to take an audit trail file that is generated by an interview and using it to "play back" a case on an in-house reviewer's computer while listening to a tape. The goal is to approximate what the interviewer sees while entering the data. This allows the reviewer to assess the quality of the entered data and the interviewer's performance, especially in areas of adherence to standardized interviewing techniques (for example, correctly handling probes or inputting data into an open-ended comment as relayed by the respondent verbatim) and accuracy in entering data.

2. Why Playback?

One concern that comes from data gathered by surveys is whether the interviewer is conducting interviews correctly. It does no good to have data gathered that was erroneously inputted by an interviewer during the course of an interview. So, something is needed to be able to check data that has been gathered and verify its soundness.

A standard method that has been used to achieve this is "read back". Essentially, cases sent into the central data-gathering location (in our case, Ann Arbor, Michigan) are re-opened in a Blaise data-entry session by a reviewer. The reviewer then examines the case compared to any information on hand, typically a tape recording of the interview, to determine the accuracy of the case and evaluate how the interviewer handles situations that arise during the course of the interview.

The Health and Retirement Study (HRS), however, could not use this method of evaluation because the logic employed in the Blaise instrument itself removed question fields from the route upon completion of certain portions of the interview. This sort of protection keeps interviewers from going back to change data that might impact future questions in the course of the interview, but it also means that once the case returns to Ann Arbor for evaluation, those questions stay unavailable and cannot be evaluated. In HRS's case, this would be unacceptable because so many portions of the interview are blocked off in this manner. Another way had to be devised to see the data as the interviewer had inputted it.

The result is "Playback," a tool that was developed for the purpose of taking audit trail files generated by Blaise and using the data in those files to re-create the case as the interviewer saw it on screen.

3. History

The origins of Playback go back to the prior wave of HRS in 2002. As noted before, HRS has special portions of its interviewing process whereby questions get "locked away" after completion, preventing any chance of changing data once those questions are answered, thus negating any benefit of just opening up the case line in a Blaise data-entry session and reviewing the data, since so much of it would not be visible.

Enter Playback. The idea was to utilize the audit trail file generated by a case to "play back" a case based on the keystrokes entered during the course of the interview. Since the case being entered was "untouched" as it were, there were no sections that were locked away from view since no data had been entered yet. Playback would parse the audit trail file and send keystroke information to the data entry session to mimic the interviewer inputting information during the course of the interview.

The initial version that was developed for HRS 2002 was only partially successful, however. It encountered numerous problems ranging from being unable to share preload between respondents as the original interview did, thus precluding an entire group of people from ever getting reviewed, to simple Blaise mechanics that weren't fully programmed for. In addition, the 2002 version, while intended for HRS initially, was expected to be a general-purpose tool eventually, so there were complicated interfaces developed to try to meet this need that hampered the review process.

For 2004, case reviewers still needed a tool to handle HRS. Other methods, such as digitally recording interviews were considered, but ultimately the decision was made to improve Playback.

The most significant improvement didn't actually have to be done to Playback itself. As mentioned before, a significant set of interviews, those pertaining to the second respondent in a two-respondent household, could not be reviewed at all in 2002 because the preload that Playback tried to run with was always from a first-respondent perspective. Any data that the first respondent gave that might impact how the interview was to be handled for the second respondent was lost because there was no repository that contained these changes. To overcome this problem for the 2004 wave, it was agreed that programming was to be added to SurveyTrak (our in-house field survey management system) that preserved the preload string for each case. With preload preservation, the retained preload string could be fed into an ASCII-to-Blaise Manipula as it is done in the field and the preloaded database would be created. This allowed second respondents to be reviewed on-screen for the first time.

Another significant change came with the agreement that when cases were completed in the field and returned to Ann Arbor, the ADK files would be kept in separate folders identified by the date of the datamodel that recorded them. Previously, in all other Blaise-developed studies by ISR, the audit trail files were all kept in the same folder for review. By breaking them down by datamodel (and extending this breakdown to the new preload string files being retained) it allowed Playback the luxury of knowing which datamodel to use for review without having to resort to going through ADK files looking for the correct one.

In addition to retaining the preload string and breaking down the ADK and preload files by datamodel, there were several other changes made to streamline and improve the program. The resulting program is at present a simple program that caters to the needs of reviewers who are looking at cases completed in HRS.

It is this present version of the program that will be discussed in detail.

4. Development Environment

Playback had to be designed to not only provide the functionality of mimicking the interviewing process, but it also had to present it to users in a relatively basic, easy-to-use fashion. The target users for this utility were in-house reviewers. These reviewers, while generally proficient with the Windows operating system, are not as technically skilled as computer programmers, so the program has to be something that they can grasp quickly and easily, without having to guess what this symbol or that button means.

Visual Basic 6, with its ability to develop windowing programs quickly and easily and its ability to parse text files without much programming complexity, was the development environment of choice. There is also a ready-made function, "SendKeys" that is the primary function used to send keystroke information from the Playback program to the DEP session, eliminating any need to program a new such function for our purposes. When this paper describes the Playback tool as "sending keystrokes" or "inputting keystrokes" or other similar phrases, this function is the primary mechanism for doing it.

5. Implementation

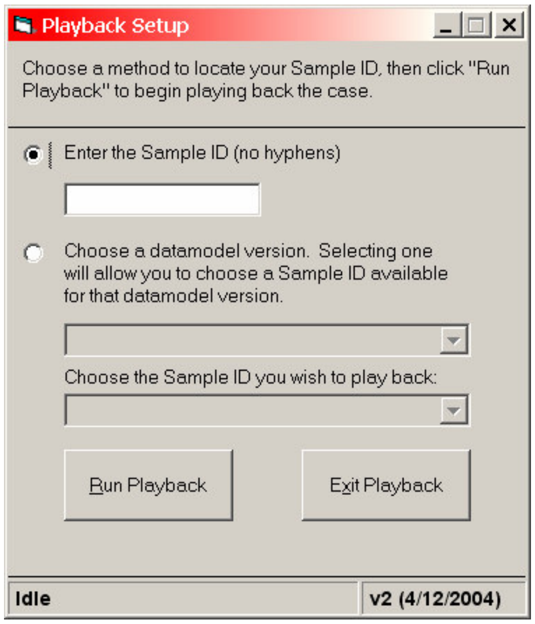
This section will detail step-by-step how Playback is designed graphically and programmatically to play back a case based on audit trail output.

5.1 – Selecting a Case to Review

Playback was designed for use by reviewers to look at how cases were being handled in the field. A typical reviewer isn't as technically proficient with computers as a programmer is, so the interfaces needed to be clean, simple, and obvious to avoid confusion.

In the original 2002 version, this wasn't necessarily so. A reviewer was at first confronted with a large window that requested they fill out several text boxes that identified various command line parameters like menu files, configuration files, and so forth. Needless to say, this sort of thing wasn't very easy to use for those not familiar with all the things that go in to running a Blaise data-entry session.

With this version being written specifically to handle HRS cases, it also gave a chance to redress some of these interface issues. Since the project of choice is hardwired into the program, there was no need anymore to have a large, complicated screen to fill out in preparation of lauching the DEP session for case reviews. Instead, all that was necessary to know was what sample ID to review. The new starting interface is shown in Figure 1.



Choose a method to locate your Sample ID, then click "Run Playback" to begin playing back the case.

[✓ Enter the Sample ID (no hyphens)]

Choose a datamodel version. Selecting one will allow you to choose a Sample ID available for that datamodel version.

Choose the Sample ID you wish to play back:

Idle | v2 (4/12/2004)

Figure 1 – Case selection interface.

In the new interface, the reviewer needs only to supply a sample ID (usually written on the tape he/she is about to listen to). Optionally, the reviewer can choose a datamodel and then browse the sample IDs available for that datamodel version to find the one he/she wants to review.

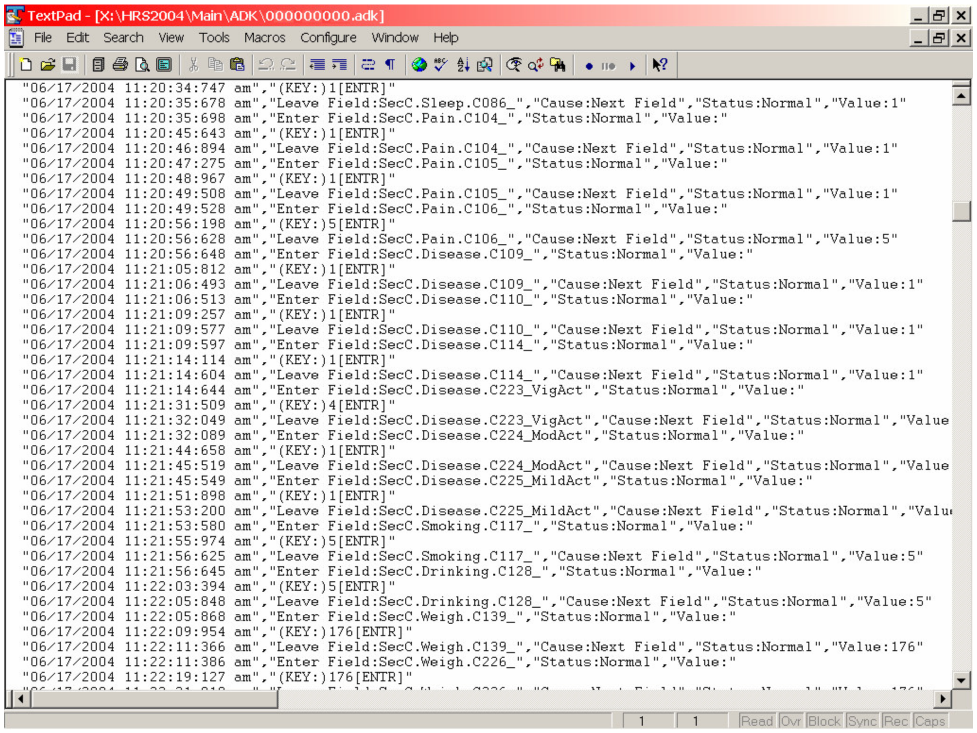
With the case now selected for review, the reviewer clicks "Run Playback" to begin the actual process of playing back a case.

At this point, Playback goes out to a network location where all the necessary files (ADK, preload, datamodel files) are stored and copies them to the reviewer's local machine. (NOTE: this network data is itself a copy of the actual data that our data operations staff uses to create reports to HRS staff, so there is no actual involvement of source files at any point in this reviewing process, just copies.)

With all files now brought down to the local machine, Playback's next step is to parse the audit trail file.

5.2 – Audit Trail Parsing

The most necessary component for Playback to work is the audit trail file, or ADK, as we call it. This file is generated by an in-house version of the auditkey.dll that Blaise supplies. A normal ADK file that is captured in the field looks something like figure 2.



TextPad - [X:\HRS2004\Main\ADK\000000000.adk]
File Edit Search View Tools Macros Configure Window Help

```
"06/17/2004 11:20:34:747 am","(KEY:)1[ENTR]"
"06/17/2004 11:20:35:678 am","Leave Field:SecC.Sleep.C086_","Cause:Next Field","Status:Normal","Value:1"
"06/17/2004 11:20:35:698 am","Enter Field:SecC.Pain.C104_","Status:Normal","Value:"
"06/17/2004 11:20:45:643 am","(KEY:)1[ENTR]"
"06/17/2004 11:20:46:894 am","Leave Field:SecC.Pain.C104_","Cause:Next Field","Status:Normal","Value:1"
"06/17/2004 11:20:47:275 am","Enter Field:SecC.Pain.C105_","Status:Normal","Value:"
"06/17/2004 11:20:48:967 am","(KEY:)1[ENTR]"
"06/17/2004 11:20:49:508 am","Leave Field:SecC.Pain.C105_","Cause:Next Field","Status:Normal","Value:1"
"06/17/2004 11:20:49:528 am","Enter Field:SecC.Pain.C106_","Status:Normal","Value:"
"06/17/2004 11:20:56:198 am","(KEY:)5[ENTR]"
"06/17/2004 11:20:56:628 am","Leave Field:SecC.Pain.C106_","Cause:Next Field","Status:Normal","Value:5"
"06/17/2004 11:20:56:648 am","Enter Field:SecC.Disease.C109_","Status:Normal","Value:"
"06/17/2004 11:21:05:812 am","(KEY:)1[ENTR]"
"06/17/2004 11:21:06:493 am","Leave Field:SecC.Disease.C109_","Cause:Next Field","Status:Normal","Value:1"
"06/17/2004 11:21:06:513 am","Enter Field:SecC.Disease.C110_","Status:Normal","Value:"
"06/17/2004 11:21:09:257 am","(KEY:)1[ENTR]"
"06/17/2004 11:21:09:577 am","Leave Field:SecC.Disease.C110_","Cause:Next Field","Status:Normal","Value:1"
"06/17/2004 11:21:09:597 am","Enter Field:SecC.Disease.C114_","Status:Normal","Value:"
"06/17/2004 11:21:14:114 am","(KEY:)1[ENTR]"
"06/17/2004 11:21:14:604 am","Leave Field:SecC.Disease.C114_","Cause:Next Field","Status:Normal","Value:1"
"06/17/2004 11:21:14:644 am","Enter Field:SecC.Disease.C223_VigAct","Status:Normal","Value:"
"06/17/2004 11:21:31:509 am","(KEY:)4[ENTR]"
"06/17/2004 11:21:32:049 am","Leave Field:SecC.Disease.C223_VigAct","Cause:Next Field","Status:Normal","Value
"06/17/2004 11:21:32:089 am","Enter Field:SecC.Disease.C224_ModAct","Status:Normal","Value:"
"06/17/2004 11:21:44:658 am","(KEY:)1[ENTR]"
"06/17/2004 11:21:45:519 am","Leave Field:SecC.Disease.C224_ModAct","Cause:Next Field","Status:Normal","Value
"06/17/2004 11:21:45:549 am","Enter Field:SecC.Disease.C225_MildAct","Status:Normal","Value:"
"06/17/2004 11:21:51:898 am","(KEY:)1[ENTR]"
"06/17/2004 11:21:53:200 am","Leave Field:SecC.Disease.C225_MildAct","Cause:Next Field","Status:Normal","Valu
"06/17/2004 11:21:53:580 am","Enter Field:SecC.Smoking.C117_","Status:Normal","Value:"
"06/17/2004 11:21:55:974 am","(KEY:)5[ENTR]"
"06/17/2004 11:21:56:625 am","Leave Field:SecC.Smoking.C117_","Cause:Next Field","Status:Normal","Value:5"
"06/17/2004 11:21:56:645 am","Enter Field:SecC.Drinking.C128_","Status:Normal","Value:"
"06/17/2004 11:22:03:394 am","(KEY:)5[ENTR]"
"06/17/2004 11:22:05:848 am","Leave Field:SecC.Drinking.C128_","Cause:Next Field","Status:Normal","Value:5"
"06/17/2004 11:22:05:868 am","Enter Field:SecC.Weigh.C139_","Status:Normal","Value:"
"06/17/2004 11:22:09:954 am","(KEY:)176[ENTR]"
"06/17/2004 11:22:11:366 am","Leave Field:SecC.Weigh.C139_","Cause:Next Field","Status:Normal","Value:176"
"06/17/2004 11:22:11:386 am","Enter Field:SecC.Weigh.C226_","Status:Normal","Value:"
"06/17/2004 11:22:19:127 am","(KEY:)176[ENTR]"
```

Figure 2 – An example of an ADK file.

A lot of information is gathered in these ADK files, and Playback must sift through the information to find what it needs to replicate the case. Chiefly, it parses through this file looking for references to "Enter Field" and "Leave Field". These references tell Playback which field data is being inputted into. More importantly, in between each set of "Enter" and "Leave" lines that say which field the interview is recording in is the line that states "(KEY:)". The data that appears after these "KEY" references are the actual keystrokes that are inputted into the interview.

Playback goes through the ADK looking for these markers and in turn stores these in an ADODB record set, which has two fields. One field contains the name of the field, the other the keystrokes as recorded. It should be noted that normal alphanumeric characters are identified separately from function keys like "Enter" or the space bar. Any string of alphanumeric characters makes up one record in the recordset. A function key such as "Enter" gets its own record in the record set. This enables Playback to accurately mimic the interviewer hitting things like "Backspace" and the reviewer is able to see that reflected on the screen, rather than having things occur in a blur if all the keystrokes for a field were played out at once. Conceptually, then, this is what Playback breaks down an ADK to:

FIELD	KEYSTROKES
Sec_C.Pain.C104_	1
Sec_C.Pain.C104_	[ENTR]
Sec_C.Pain.C105_	1
Sec_C.Pain.C105_	[ENTR]
Sec_C.Pain.C106_	5
Sec_C.Pain.C106_	[ENTR]

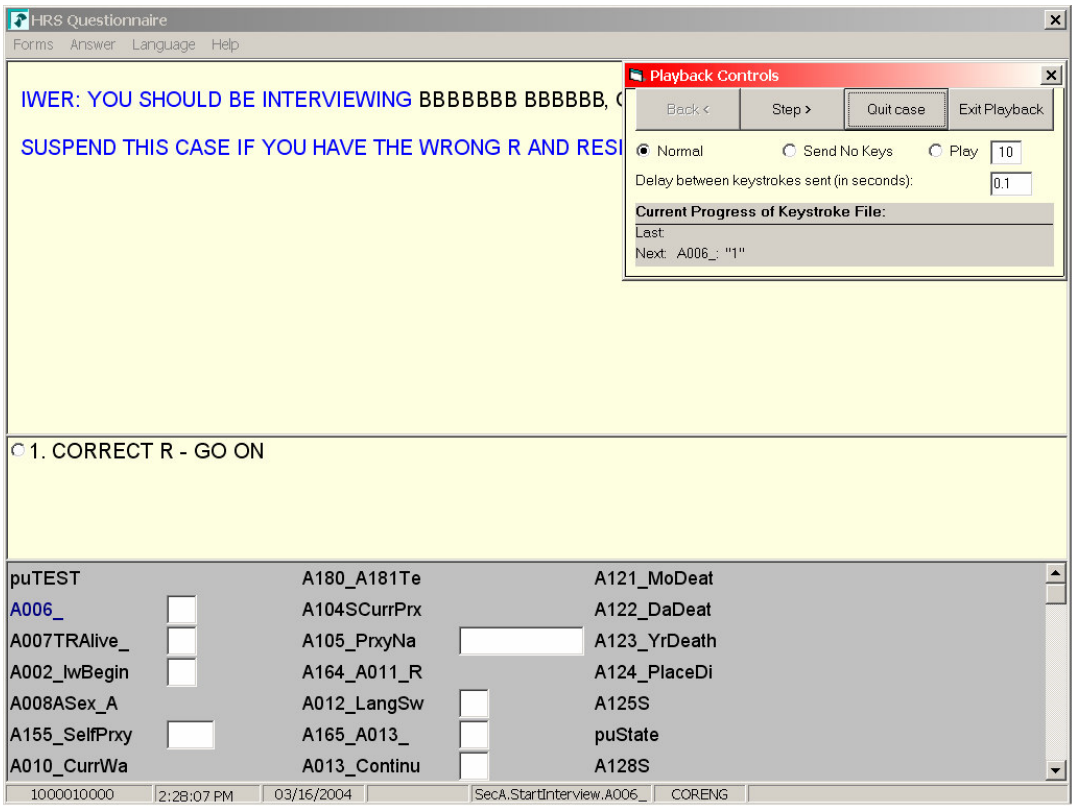
However, since Playback is sending keystrokes via Visual Basic to the DEP, the program must change function keys into something that can be understood by the SendKeys function in Visual

Basic. So, Playback also has been programmed to change "[ENTR]" as given by the ADK file to "{ENTER}" as SendKeys could understand and handle. This sort of manipulation is especially needed in case the interviewer has to put in a DK or RF response to a question. These combination keys shows up as "[ALT]d" or "[ALT]r" in the keystroke file. So, Playback has to infer that when [ALT] is pressed and it is followed by a "d" or "r" that it must mean the interviewer pressed both together at the same time, since combination keystrokes are not actually noted in the ADK file. As a result, Playback replaces these "ALT" keystrokes with the appropriate "^%" code that Visual Basic uses to represent a combination keystroke using the ALT key.

There are several other manipulations of the data that comes from the ADK file that allow for Playback to do its job better. We won't go into details on all of them, but suffice to say a lot of conversion between ADK output and Visual Basic input occurs.

5.3 – The Playback of the Case

Once the ADK has been parsed, Playback then takes the preload.asc file it has copied and runs it through a Manipula ASCII-to-Blaise script that creates a preloaded database for a starting point to the case. With that accomplished the program then calls the DEP session into existence. A screen similar to the one in figure 3 appears for the reviewer:



IWER: YOU SHOULD BE INTERVIEWING BBBBBBB BBBBBB, (

SUSPEND THIS CASE IF YOU HAVE THE WRONG R AND RESI

[✓ Normal] Send No Keys Play 10

Delay between keystrokes sent (in seconds): 0.1

Current Progress of Keystroke File:
Last:
Next: A006_: "1"

1. CORRECT R - GO ON

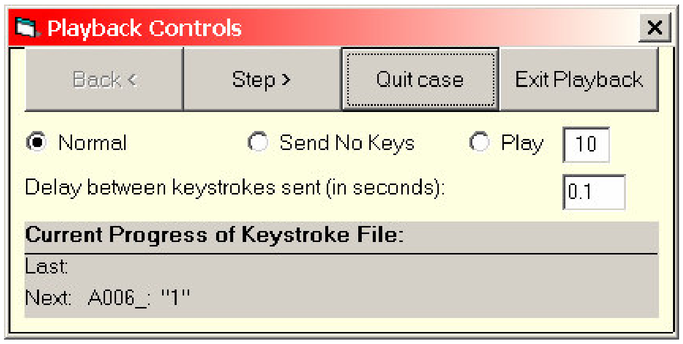
puTEST	A180_A181Te	A121_MoDeat
A006_	A104SCurrPrx	A122_DaDeat
A007TRAlive_	A105_PrxyNa	A123_YrDeath
A002_IwBegin	A164_A011_R	A124_PlaceDi
A008ASex_A	A012_LangSw	A125S
A155_SelfPrxy	A165_A013_	puState
A010_CurrWa	A013_Continu	A128S

1000010000 | 2:28:07 PM | 03/16/2004 | SecA.StartInterview.A006_ | CORENG

Figure 3 – Playback of a case at startup.

In the upper right is the actual control that manipulates the playback of the case. This interface has also gone under a significant change since the 2002 wave. In that version, this control consisted of VCR-like buttons for "play", "stop", and "pause" (pause was necessary because the original version allowed for a continuous play of keystrokes). In addition there were up and down arrows and a stop sign labeled "EXIT". The interface was somewhat confusing to use because it wasn't immediately clear whether "Stop" or "EXIT" in a stop sign did the same thing or completely different things. In addition, with something like continuous play, a reviewer would have to constantly pause the playback and resume it to keep in time with what he/she was hearing on the tape. There was no mechanism to back up if something should go wrong with the playback of the case, which can and did happen frequently as Playback wasn't robust enough to handle all the little things that go on in a Blaise instrument, like out-of-range boxes. The only information the reviewer had about where he/she was in the audit trail of the case was a couple of lines describing the last field that had a keystroke inputted and which keystroke it was.

With the new version, the interface was re-worked to make it simpler and more informative for the case reviewer. Gone are the VCR buttons. The option for continous play has been removed. Instead Playback operates on a step-by-step process, going one field at a time, at the pace the reviewer wants based on the case being listened to. Figure 4 shows this new interface.



● Normal ○ Send No Keys ○ Play 10

Delay between keystrokes sent (in seconds): 0.1

Current Progress of Keystroke File:

Last:

Next: A006_: "1"

Figure 4 – The Playback control interface.

There are several parts to the Playback control interface.
*Back: This allows the reviewer to step backwards through the keystroke file. Do note that this does not alter the present state of the data entry session. The cursor in the DEP window will remain at the field the reviewer was at before pressing the back button. It does not attempt to "undo" the keystrokes that were played in the course of stepping backwards. (It would be difficult to reverse keystrokes, especially ones related to remark files, since the keystroke that opens one would occur before any text put into a remark.)

To resume Playback properly after using the Back button, move the cursor in the data entry session to the field listed in the "Next" portion of the "Current Progress of Keystroke File" part of the window.

*Step: This is the driving portion of the program. Each time "Step" is clicked to send keystrokes to the DEP session the program consults where it is in the record set, plays the keystrokes for the current record, and moves to the next one. If that record has the same field name as the prior one, it goes ahead and plays the keystrokes associated with the record it is now currently looking at. It continues in this process until it reaches a record where the prior record has a field name different from the current record. This allows Playback to put in all the keystrokes for a field without having the reviewer spend the time to click through each portion of the keystrokes for a given field. In addition, for remark files, Playback will delay closing the remark box for a number of seconds to allow the reviewer to read what was inputted. If it isn't enough time, the reviewer may return the cursor to that field and open the remark again. Once done the reviewer can move the cursor back to the field listed in the "Next" area of the playback control to resume normal playback. Playback also recognizes when checks or signals occur in the keystroke file and can close the box if the error was suppressed or exited or will attempt to display to the reviewer where the instrument moved to if the interviewer elected to correct a mistake by moving to a different field.

Beneath the row of playback buttons, the reviewer can further control what the "Step" button does. The reviewer can do one of three things: (A) Normal, which means the program will send all of the keystrokes for one field at one time and then stop again, (B) Send No Keys, which essentially is like Back, only the reviewer is now stepping forward through the keystroke file without changing the data entry session's state, or (C) Play, which has a box next to it in which the user can put in a number of fields Playback should attempt to go through each time Step is clicked. In addition, there is an option below these choices to determine how quickly or slowly the user wants keystrokes fed to the data entry session, the larger the value, the slower the progression through the individual keystrokes. This allows the reviewer better ability to track what's being inputted for each field, but results in greater amounts of time spent reviewing the case.

*Quit Case: This option allows the reviewer to terminate the case being reviewed and return to the startup screen to select another case to review.

*Exit Playback: The reviewer can click this button to terminate the application entirely without returning to the startup screen.

At the bottom of the control window is displayed the current position of Playback within the set of keystrokes that make up the case. "Last" lists the last field that received a keystroke and the keystroke that was sent to the data entry session. "Next" normally lists the field the data entry session is found in and the upcoming keystroke ready to be sent to the session. This portion of the control acts as a check for the user to make certain that the keystroke file is in sync with the data entry session, since there is no programmed logic in the program that controls this aspect.

Once a case has been completed or the user terminates the control by clicking on the "Quit Case" or "Exit Playback" buttons, Playback will attempt to do garbage collection by deleting all the files it copied over from the network location, thus removing any personal data from the reviewer's machine after completion of case review.

6. Issues

Playback as it stands today is a solid utility that helps reviewers get through cases that they could not look at otherwise. This is not to say there are no flaws with the program, however. Playback still has many limitations. Chief among them is the fact that Playback is an unintelligent system. That is to say there is no way for the Playback control to understand what is going on in the DEP session. If, for example, an out-of-range error occurred in the DEP session, the Playback control has no awareness of it since audit trail files do not record interviewer actions in these external boxes. The reviewer must manually close this error box before continuing. This sort of bump is especially problematic for instances of continuous play, since the Playback program will continue to feed keystrokes to the DEP even if the DEP is no longer the focus because of the out-of-range error box. The only way to correct such a mistake is to use the "Back" button to move the keystroke file record back to the field that generated the error and then resume playback from the point of the correction.

Another point of weakness is the handling of alien routers. In HRS these routers produce word lists and state lists similar to lookups in Blaise. However, since these routers launch windows that exist outside the DEP, any keystrokes are not recorded and the ADK file actually has no "Leave Field" line denoting the interview moving on to the next field. This caused serious problems for the parser. A workaround was made by having the parser recognize the specific fields that have alien router lookups (hardcoded) and adding the necessary information to the ADK file to put a DK response into that field, since normal keystrokes wouldn't work in those fields. However, the problem with this sort of solution is if the instrument's route depends on an answer given in the alien router call, it will throw the program off-track without any way to recover since the reviewer cannot input the data needed to send it down the correct path.

An issue that only comes up rarely, yet is significant is if the interviewer suspends a case. At present, Playback isn't programmed to resume a suspended case at all. Furthermore, there isn't any consideration for if the interviewer is issued a new datamodel and the case is therefore migrated to the new version. In that instance, more code would have to be programmed to handle a migration and resumption. None of this is there now.

Aside from those issues (and, of course, the occasional bug), Playback has been able to satisfactorily meet the needs of the reviewers involved in looking at HRS cases.

7. The Future

At the present time, Playback is geared specifically to handle HRS cases only. It was necessary due to the short development time available to get the product ready for cases coming in for the 2004 wave of HRS. However, if time and resources are made available, it is probable that this program will be rebuilt from the ground up into a general-purpose reviewing program that will handle a broad range of projects.

8. Conclusion

In conclusion, Playback has largely been able to meet the demands placed on it by those who use it most, the reviewers. The advances the program has made since the 2002 wave of HRS have been able to increase considerably the number of cases that reviewers can review and have been able to review. There are limitations and places for improvement, of course, but it has largely been a successful utility.