Experiences Using an Event History Calendar in the Panel Study of Income Dynamics

April Beaulé, Eva Leissou and Youhong Liu, University of Michigan

1. Introduction

The Panel Study of Income Dynamics (PSID) is a nationally representative longitudinal study of approximately 8000 U.S. families. The Board of Directors of the PSID continually search for ways to innovate the data collection instrument. For the 36th wave of the PSID, a major new component was added. A childhood health calendar was programmed to collect information on the most common childhood medical conditions including asthma, diabetes and allergies. Since this childhood health component is retrospective focusing on the time from the respondent's birth to age seventeen, the design team anticipated that using a calendar module would help trigger the respondent's memory. The PSID had prior experience using an electronic calendar for employment data outside of the main Blaise interview module. This separate employment calendar presented many obstacles, the most significant of which was the processing of separate file structures: Blaise and Access databases. In order to alleviate some of these processing issues, the decision was made to incorporate a Visual Basic (VB) calendar that would be called from the Blaise Data Entry Program (DEP). The data from the VB calendar application was written directly back to the Blaise bdb using a Dynamic Link Library thus avoiding the creation of multiple datasets.

In addition to data structure and extraction issues, the calendar mode versus the standard question list presented us with a number of challenges during interviewer training. This paper describes the challenges that we faced when fielding this calendar including data structure, extraction, and interviewer training as well as the strategies used to mitigate these issues.

2. Background

The impetus for the use of the Event History Calendar method of interviewing surfaced for the PSID when budgetary constraints forced the project to move from an annual interviewing cycle to an every other year interviewing cycle. The primary concern was whether respondents would be able to provide the same level of reporting accuracy for the two calendar years prior to the interviewing year. Based on the cognitive recall work of Robert Belli, the PSID conducted an experiment in 1998 on the use of an Event History Calendar to collect data in the core domains used in the PSID including housing moves and employment spells. Half of the cases were randomly assigned to each condition: (1) Standard question-list format and (2) Event History Calendar format. The results of the experiment showed that the EHC (Event Calendar History) method of interviewing provided more accurate autobiographical retrospective reporting in comparison with the standard question list method. The experiment also showed that there was no significant difference in the amount of time it took to conduct each of the interviewing treatments.[1]

[1] From PSID website see overview of Calendar Methods Study
http://psidonline.isr.umich.edu/Data/documentation/ehc/PSIDcalendarMethodsStudy.html

Based on the results of the experiment, the leadership of the PSID decided that the core questions regarding employment spells, time off and housing moves would move to an electronic calendar instead of traditional question list. The PSID application was overhauled in 2003 with the two major changes being a move from Surveycraft to Blaise and creating and EHC for the job sections. Since Blaise did not offer a method for programming a calendar type grid, an EHC for the employment domains was programmed in Visual Basic with a Microsoft Access database as the backend. At the time the 2003 application was programmed, there were significant limitations in using Dynamic Link Libraries the most critical of which was stability. The University of Michigan programmers experienced several problems trying to implement alien routers and procedures in large applications such as the PSID.[2] As a result of these limitations, the PSID application was split into three separate Blaise applications and two Visual Basic calendars. An in-house "Multiple Application Interface (MAI)" was developed to handle moving data from one application to the other.

3. Prior Experience with EHC Data Collection

The data collection using the EHC seemed to go smoothly with the exception of minor technical difficulties for some cases using the utility that passed preload to each instrument. The interviewer feedback indicated that the EHC seemed to aid respondents with date recall as the calendar allowed data entry to be flexible. The grid format allowed the movement from one domain and back again depending on how the respondent was able to recall date information. However, the processing and release of data in the first wave using the EHC was difficult and time consuming. Because the PSID is a panel survey, the need for consistency over time is crucial. The employment data was collected in a different format from all previous waves but the final output variables needed to line up with prior waves.

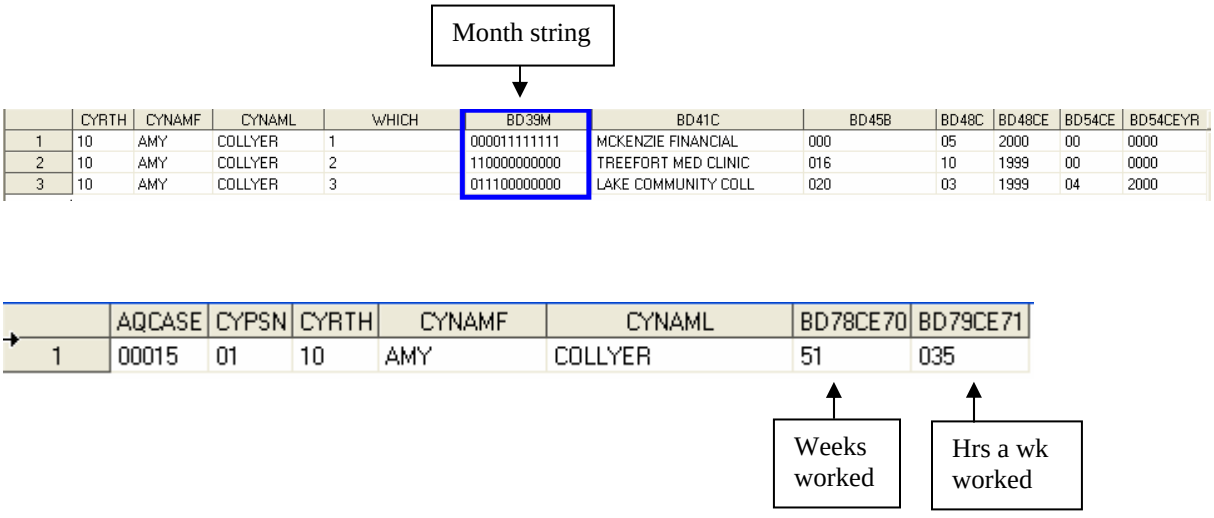
Below is an example of the differences between data collection techniques and output variables. The screen shots show a comparison of how work weeks were asked between the question list and the EHC:

[2] See 2003 IBUC paper by Hagerman and Kannan, University of Michigan

Figure 1: Number of weeks and hours worked in Question-List Format

In the question-list structure the interviewer marks which months the respondent has worked. This translates into a month string field (Figure 2), one of our key employment variables used during editing. We also ask "So how many weeks out of the year did you actually work at this job 2006?" and "On average, how many hours a week did you work on this job in 2006?" (Figure 1)

Figure 2: Month String, Weeks Worked and Hours Worked- SAS dataset-List Format



In the EHC structure for 2003, we no longer asked the questions in the same way. We asked when each job started and stopped and the interviewer marked each job on the calendar.

Figure 3: Number of weeks worked in EHC Format

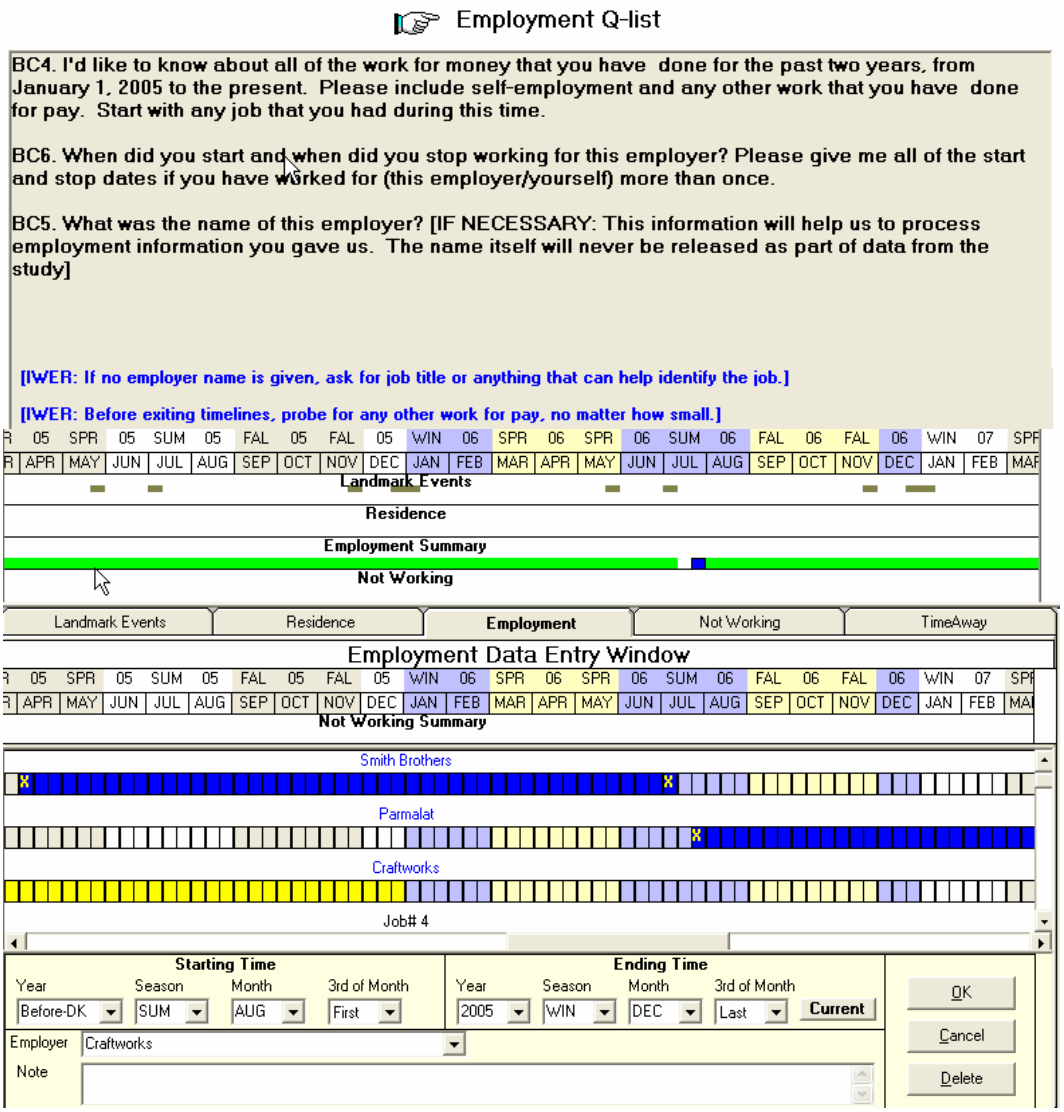


Figure 4: EHC Table Structure in Access

SampleID	JobNu	cEDate	SYear	SSeason	i_sMonth	SMonth	Sby3rd	EYear	ESeason	i_eMon	EMonth	Eby3rd	Employer	Currrent
0004003	1	07/01/2006	2005	SPR	4	APR	First	2006	SUM	7	JUL	First	Smith Brothers	0
0004003	2	04/21/2007	2006	SUM	7	JUL	Last	2007	SPR	4	APR	Last	Parmalat	1
0004003	3	12/21/2005	B-DK	SUM	8	AUG	First	2005	WIN	12	DEC	Last	Craftworks	0

In the MS Access database (Figure 4) we captured start and stop dates for each employment spell. We had to construct month strings. For respondents with jobs that were seasonal, it was not clear from the start and stop dates alone which months they were actually working.

We also had to construct number of weeks worked and this was tricky for several reasons. The calendar is divided into thirds of months and it was often difficult to tell if a job ended in the second third of June what that meant. Perhaps it meant June 15th or perhaps the 3rd week of June. When jobs overlapped, calculating total weeks and hours worked was even more problematic. The result of the new data collection technique required several weeks of hand corrections by editors in order to construct variables that were comparable to those asked in the question-list format in prior waves.

The incorporation of the EHC also proved a challenge in other areas. Visual Basic is not a robust survey software and thus we were unable to enforce the type of constraints that are available in software such as Blaise. Consistency checks were written for the calendar but for some cases we still ended up with missing data. The in-house application that passed preload from one application to the other also failed at times, which required us to do callbacks on a small number of cases.

One of the major implications of incorporating the EHC was that we were forced to break up the PSID application into five different applications and five different datasets. From the interviewing perspective, this was a significant drawback because once each application was complete; the interviewer could not back up to that section of the interview to make a correction.

For processing and documentation, separate datasets in different formats required us to have different utilities to import data to SAS. The audit trail on the calendar was not as sophisticated as the audit trail in Blaise. While we could generate detailed timing reports from Blaise data, we were not able to replicate these for the calendar data. Automated tools for documentation such as the Michigan Questionnaire Documentation System could not be used on the EHC files. In addition, the EHC files did not contain the same meta-data information that we routinely extract from the Blaise files.

4. Training Interviewers on the use of an EHC

Interviewer trainings start with an introduction to the General Interviewing Techniques (GIT) that teach interviewers core interviewing skills including standard interviewing protocols for all interview components. The teaching and use of standard interviewing protocols is meant to promote consistency across data collectors and is easy to use when the questionnaire format is a question list. The interviewer reads the questions verbatim and records answers, either close or open ended. The EHC sequence of the interview administration has some deviations from this standard because not every question or probe is scripted, and recording answers is done using both mouse and keyboard entry. In addition, the interviewer has to navigate through the various tabs where the scripted questions are or where they record answers.

The EHC administration has presented us with two challenges which we had to consider when developing the training protocols and material: 1) use of non-scripted questions or probes, and 2) recording answers using a mouse and keyboard. For the first challenge we had to teach interviewers that the kind of questions or probes that are allowed based on the fields they have to fill out. For example, in the "Residence Domain", we ask for residential moves done in the last two years, but do not include every question asking for start and end time of each residence move reported. The interviewer uses their own simple phrases or questions to get the year or month of the move. In addition, if the respondent does not know the exact move date, the interviewer probes for the season

when the move occurred. For data entry in Blaise, interviewers are trained to use the keyboard only, but within EHC they are required to switch from keyboard to mouse. Training using multiple data entry modes has been a time consuming task for those interviewers who are less adaptable.

The EHC training has two modes of presentation: home study and in-person training. The goal of presenting the material in different modes is to reach every type of learner, those who are visual and those who need more hands on practice. Interviewers are given a home study packet which includes a study manual, and a DVD. The study manual covers in detail the goals of the EHC questionnaire and an overview of the various tabs, using screenshots to demonstrate the various tasks interviewers will be doing. This manual is a training tool and user's manual. The DVD also provides an introduction to the basic concepts covered in EHC as well as a demo on data entry.

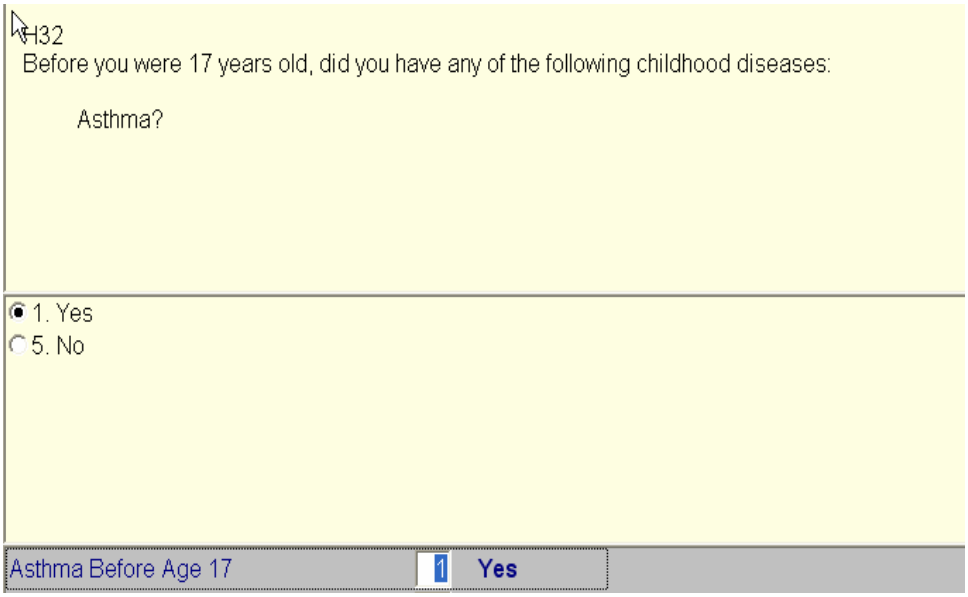
At the in-person training the same material, basic questionnaire concepts and data entry, are covered in more detail. This is followed by practice sessions using a stand-alone module. The stand-alone application allows interviewers to practice the EHC without having to go through the full questionnaire. This module is used for extra practice during the training sessions, the extra help sessions during the in-person trainings, and during the production phase when remedial training may be necessary for some staff.

The interviewer certification protocol includes scoring performance in EHC administration. The certifiers make observations and score specifically the ability of interviewers to do accurate data entries, maneuver with efficiency through the various tabs, and the use of appropriate probes to clarify responses in order to capture a complete and accurate answer.

5. Experimenting with EHC using Dynamic Link Libraries

In spite of the technical difficulties incorporating the calendar, PSID senior staff recognized the value of using an Event History Calendar for certain types of question series. For the 2007 data collection wave, a new retrospective health history sequence was planned. The questions focused on childhood health conditions. This retrospective health history data collection was a prime candidate for a calendar format since the question series focused on a period of the respondent's life from birth to age seventeen. In order to overcome some of the past problems with EHC collection and processing, a DLL call was used that passed control to the subroutine (Visual Basic calendar) if the respondent confirmed they had any of the childhood health conditions listed. If any of the conditions were endorsed in the screening section (Figure 5), then control would pass from Blaise to the EHC where follow up questions were asked. If no conditions were endorsed by the respondents Blaise would continue on to the next field on route.

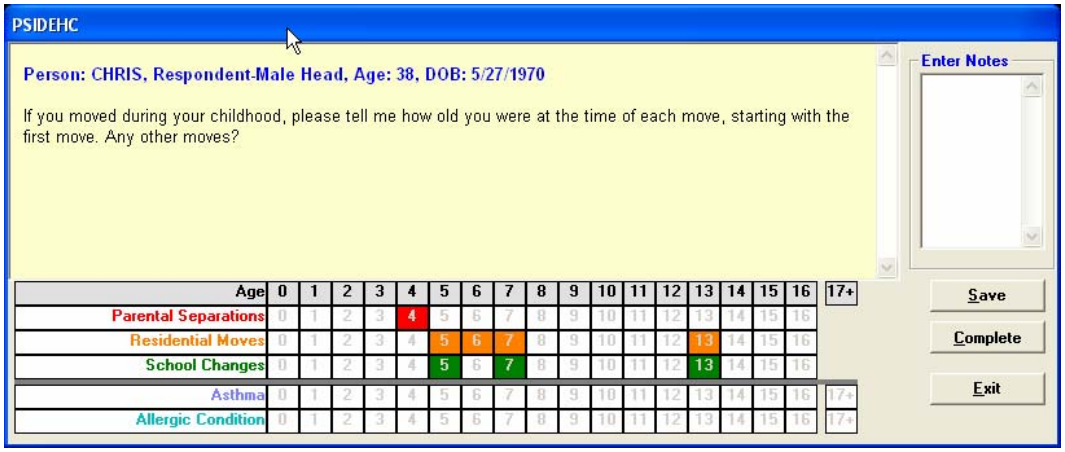
Figure 5: Screening Question in Blaise for the EHC Childhood Health Calendar



The data collected in the calendar was then sent back to the Blaise bdb via the DLL. By incorporating the calendar in this fashion, we were able to overcome several issues. Because communication between Blaise and the calendar is dynamic, interviewers were able to back up to make corrections, then move forward and have those changes be reflected in the calendar. For the data processing team, instead of having to process different datasets, all calendar data was written back to bdb. The use of the DLL was significantly more stable than passing preload using our in-house application interface software. We had no reports from the field of the DLL link to the calendar failing.

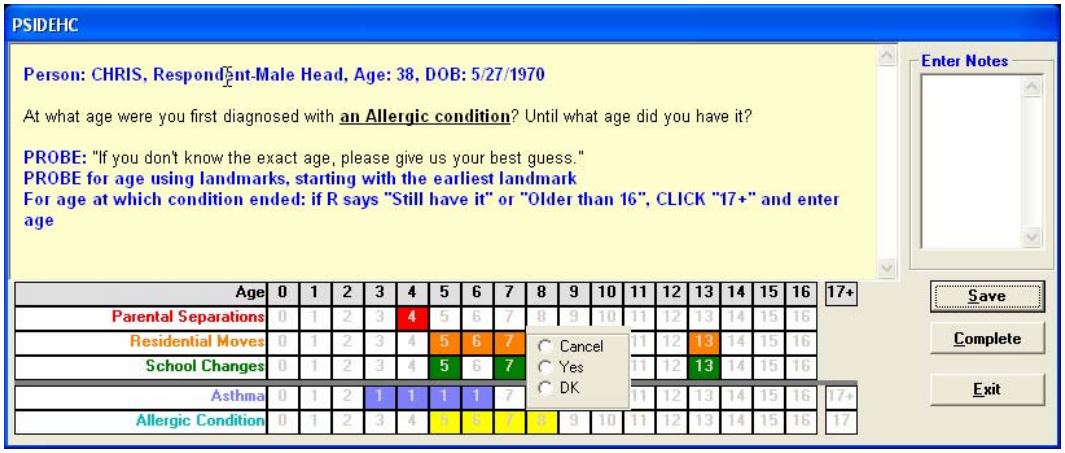
Given our past experience with employment calendar data, the structure of the health history calendar data was designed for ease of processing. Instead of having start and stop dates like the employment section, we programmed a grid that included a cell for each age from birth to age seventeen. The interviewer began by asking a short series of "landmark" questions about other childhood events. The respondent's answers could then be used to anchor the follow-up health questions (Figure 6).

Figure 6: Landmarks EHC Childhood Health Calendar



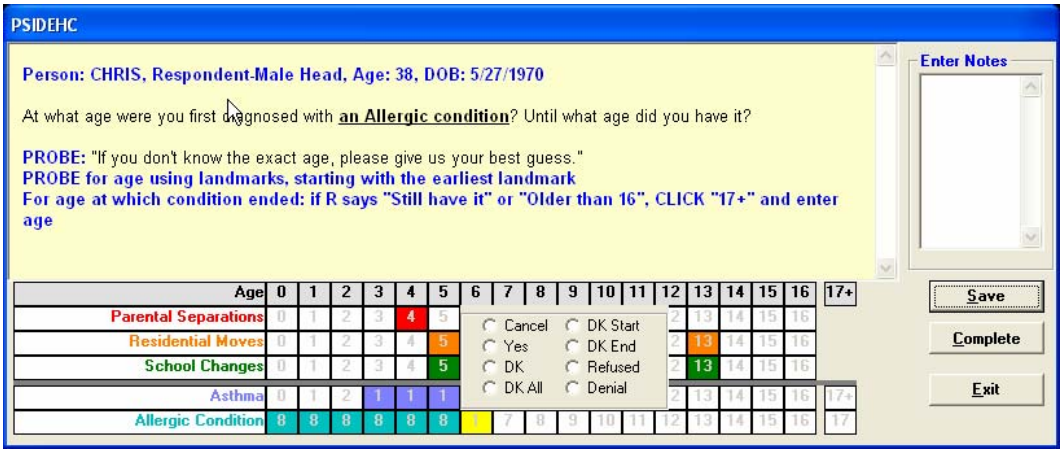
Next the interviewer asked follow up questions about the conditions endorsed in the previous screening section (Figure 5). When the interviewer clicked the start and stop cells and then 'Yes', the ages included in that range would be marked '1' meaning 'Yes' (Figure 7).

Figure 7: Simple Data Entry Childhood Health Calendar Conditions



We also allowed for more complex coding if the respondent could not remember exact start or stop ages. For example, if a respondent said that he or she had an allergic condition that stopped at age six but could not remember when it started the interviewer could right click on age six and choose 'DK Start' (Figure 8). It terms of data, we record '8' meaning 'Don't Know' for all the ages from birth to age five and then '1' meaning 'Yes' for age six. This method of using one cell to represent each year of childhood made the processing of these variables clean and consistent. We can easily generate start and stop ages for various conditions or we can leave the construction of those variables to the user.

Figure 8: Complex Data Entry Childhood Health Calendar Conditions



6. Programming Logistics

The first step in programming the interface was to create the data structure in Blaise to work with the grid in the Visual Basic form. Blaise code was then written to call the Childhood Heath Calendar (Figure 9).

Figure 9: Design the Data Structure in Blaise

```
AsthmaEHC : ARRAY[0..17] OF 1..7      {Corresponding to a row in the VB form}
AsthmaEHCAgeGR17 : 17..120            {Store age if grid 17+ is checked}
 IF HadAsthma = YES THEN
        ConditionCount := ConditionCount + 1
 ELSE
{This code is necessary here, in case the condition is changed from yes to no.
The DLL will not clean up the data.)
   FOR I:= 0 to 17 DO
      EHC.OtherProbEHC[I] :=EMPTY
   ENDDO
  EHC.OtherProbEHCAgeGR17 := EMPTY
 ENDIF
IF ConditionCount > 0 THEN {At least one problem is checked "yes"}
  EHC.PSIDEHC (xDOB , 'Jone Smith')
ENDIF
```

The Visual Basic form is programmed to work with the Blaise data and bdb structure.

Figure 10: Design the Visual Basic Form

The childhood health conditions are loaded dynamically based on the yes/no screening questions in the prior Blaise DEP sequence. For example, in Figure 10, the respondent endorsed two conditions: asthma and a respiratory disorder. Screening questions for all the other conditions were not endorsed and therefore they do not appear on the grid.

Once the interviewer has completed the grid and has clicked the "Save" button, the data are saved back to the bdb (Figure 11).

Figure 11: Save Data in the Visual Basic Form back to the Blaise bdb

```
If Controls(lblstring)(i).BackColor = Controls(lblstring + "txt").ForeColor Then
        If Controls(lblstring)(i).Caption = "1" Or _
            Controls(lblstring)(i).Caption = "7" Then
           dbp.Field(FieldBasename + "[" + CStr(i) + "]").Text = Controls(lblstring)(i).Caption
           xmlTS.WriteLine "<Field name=""" + _
                FieldBasename + "[" + CStr(i) + "]" _
                + """ value=""" + Controls(lblstring)(i).Caption + """ />"
        ElseIf Controls(lblstring)(i).Caption = "8" Then
           dbp.Field(FieldBasename + "[" + CStr(i) + "]").Status = blfsDontKnow
           xmlTS.WriteLine "<Field name=""" + _
                FieldBasename + "[" + CStr(i) + "]" _
                + """ value=""DK"" />"
        ElseIf Controls(lblstring)(i).Caption = "9" Then
           dbp.Field(FieldBasename + "[" + CStr(i) + "]").Status = blfsRefusal
           xmlTS.WriteLine "<Field name=""" + _
                FieldBasename + "[" + CStr(i) + "]" _
                + """ value=""RF"" />"
        Else
           dbp.Field(FieldBasename + "[" + CStr(i) + "]").Text = ""
        End If
```

In addition to saving the data to the bdb, this code (Figure 11) also saves the data to an xml file for backup. The xml file is essential because the Blaise audit trail function does not capture any data collected in an external application such as the VB form.

Figure 12: Load data from bdb to Visual Basic Form

```
If dbp.Field(FieldBasename + "[" + CStr(i) + "]").Status = blfsResponse Then
        Me.Controls(lblstring)(i).BackColor = Controls(lblstring + "txt").ForeColor
        Me.Controls(lblstring)(i).Caption = dbp.Field(FieldBasename + "[" + CStr(i) + "]").Text '1 or 7
    ElseIf dbp.Field(FieldBasename + "[" + CStr(i) + "]").Status = blfsDontKnow Then
        Me.Controls(lblstring)(i).BackColor = Controls(lblstring + "txt").ForeColor
        Me.Controls(lblstring)(i).Caption = "8"
    ElseIf dbp.Field(FieldBasename + "[" + CStr(i) + "]").Status = blfsRefusal Then
        Me.Controls(lblstring)(i).BackColor = Controls(lblstring + "txt").ForeColor
        Me.Controls(lblstring)(i).Caption = "9"
End If
```

When the calendar is invoked for the first time, all cells in the VB form are empty. However, if the interviewer backs up and makes a correction and revisits the field that launches the calendar, the VB form will reopen for additional edits. The code above (Figure 12) loads the bdb values to the VB cells so that any data collected previously will not be lost.

Figure 13: Consistency Check when Exiting Calendar

```
CHECK    HR4aCheck  < 18 AND
   HR4bCheck < 18 AND {HR4bCheck = 18 means all items for Asthma array = EMPTY}
   HR4cCheck < 18 AND
   HR4dCheck < 18 AND
   HR4eCheck < 18 AND
   HR4fCheck < 18 AND
   HR4gCheck < 18  AND
   HR5aCheck < 18 AND
   HR5bCheck < 18 AND
   HR5cCheck < 18 AND
   HR5dCheck < 18 AND
   HR5fCheck < 18 AND
   HR5gCheck < 18 AND
   HR5hCheck < 18
INVOLVING (HR12_INTRO)
   "Calendar not Complete!"
```

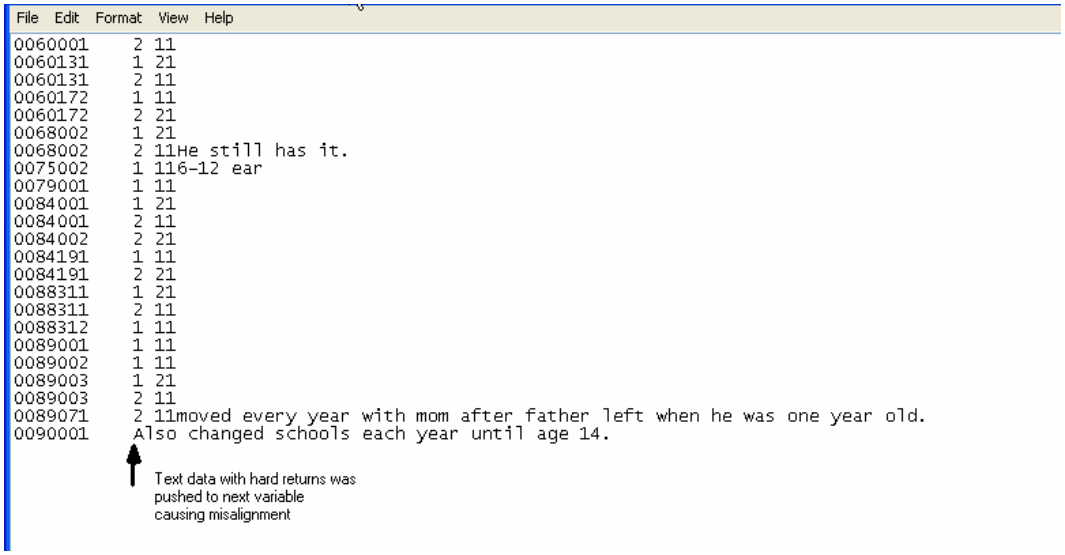
Once the interviewer has indicated that the calendar is complete, a final consistency check (Figure 13) is performed to ensure that all the conditions endorsed have at least one cell filled out indicating when the respondent had that condition. If the interviewer fails to record a value for any condition they are prompted by a hard check indicating the calendar is not complete.

7. Calendar Data Processing

Prior to production, the health history calendar was pretested and we did process and review all pretest data without incident. However, in early data dumps during production, we began seeing some problems in the calendar variables. We researched the cases involved and found that the bdb data looked correct but that our SAS data was not

lining up correctly. We narrowed the problem down to our Blaise to SAS extraction routine. Normally the first step in our routine from Blaise to SAS is to generate an ASCII delimited file with each case representing one row (Figure 14).

Figure 14: Blaise to ACSII delimited file



We incorporated a note field in the EHC health history calendar for the interviewer to provide us any clarification on the calendar (Figure 7- top right corner). Unlike data entry in Blaise, the note field in the Visual Basic calendar allowed interviewers to enter notes that contained hard returns. In Blaise if a hard return is entered in a string field the interviewer is pushed to the next question on the route. Our extraction routine didn't anticipate hard returns in string fields and interpreted the presence of hard returns as an indicator to move the remaining text data to the next variable. For those cases that had hard returns, each instance would push the remaining text data to the next variable and thus all the variables for that point forward were misaligned. Since we hadn't had any cases with hard returns in the note field during pretest, we did not find this problem at that stage.

Our solution was to alter the DLL to remove all the hard returns before writing them back to the Blaise bdb. The new DLL was then sent out to all the interviewer laptops to correct all future cases and was also run on the master file to resolve all of our older cases.

8. Conclusion and Summary

The PSID use of a DLL routine to call an external program such as the EHC was much more successful that our earlier experiences with EHC data collection using a separate application. There are many advantages to having all the data in one format and in one database. The DLL performance was extremely stable in moving data to the external application and back to the Blaise bdb. Having one dataset also means that we can take advantage of other applications that can read Blaise files such as the Michigan Questionnaire Documentation System (MQDS).

Because of our experience with EHC for employment questions, we were able to construct the EHC for the health history calendar with a more concise data structure. It required the use of more variables but reduced ambiguity for start and stop dates. Because of the resulting data structure, users can now easily construct their own variables without PSID staff assistance.

We did find however, that because the EHC is written in another programming language, we need to be careful about how we write back that information to the Blaise bdb. We need to consider the type of restrictions and requirements that are associated with Blaise fields we are writing to and apply the same restrictions and requirements to the fields in the corresponding external application.

9. References

From PSID website see overview of Calendar Methods Study. Retrieved May 31st 2007 from World Wide Web:
http://psidonline.isr.umich.edu/Data/documentation/ehc/PSIDcalendarMethodsStudy.html

Hagerman, J. and Kannan, H. (2003): The "Multiple Application Interface" with Blaise and Visual Basic, Proceedings of the 8th International Blaise Users Conference, Copenhagen, Denmark, May 2003.