DEP Unlimited: Solving Complex Run-Time Problems with Manipula

Rick Dulaney and Peter Stegehuis, Westat, United States

1. Abstract

The Blaise 4.8 Data Entry Program has the capability to shell out to Manipula, which allows us to solve some difficult problems during data collection. We will show three examples: (1) When selecting dates, it can be desirable to use a graphical date-picker rather than the Blaise Date type. We use Manipula to establish the allowable date range, to display and manage the date-picker, and to store the resulting data in the active datamodel. (2) There are times when it is desirable to add "off-path" data during the course of an interview. We describe some advantages of using Manipula rather than parallel blocks. (3) The visible components and controls available in Manipula can provide a very effective tool for managing lists or rosters. In addition to displaying and selecting items from the list, we can use Manipula to allow additions and modifications to the list as well.

2. Introduction

Blaise 4.8 introduced significant new capabilities in Manipula. The traditional use for Manipula prior to version 4.8 was for batch manipulation of data between Blaise datamodels, or for extract-transform-load operations between Blaise and external software systems. In version 4.8, Manipula functionality is part of the DEP executable, which means Manipula can operate on the DEP record in memory, with simultaneous access to other datamodels and external data files. Manipula now also has direct access to the Blaise metadata, complementing and sometimes replacing Cameleon programming.

As we noted in our 2013 IBUC paper, one approach to solving complex routing or data access problems while the DEP is running is to shell out to Manipula. A Blaise datamodel can invoke Manipula at run-time in several ways, including menu selection, buttons programmed to appear (or not) on the task ribbon, or even as part of selected response using data types. Manipula programs can show dialog boxes, read and write data using Blaise datamodels, read and write data from non-Blaise sources, and even update the record in memory using the datamodel which called Manipula and is still active. There are new language keywords to help manage the return from Manipula to the active Blaise datamodel, including GETACTIVEFIELD (get the full name of the field from which the Manipula procedure was started), SETACTIVEFIELD (the full name of the field to return to after the procedure has been completed) and SETALIENROUTERACTION (to specify the desired action after the Manipula procedure has finished).

The basic approach is to invoke Manipula from the Blaise datamodel, while the DEP is running, using one of the methods above. We can then use Manipula to read, display, update and store data in and from a variety of data sources. Finally, the Manipula program can return to the DEP either at the place we departed or at some other point determined algorithmically. This relatively simple sequence can be used to solve some very complex problems. We discuss three of these challenges below - for each, we first describe the challenge and then provide a solution using Manipula during run-time.

3. Challenge 1: Managing Rosters

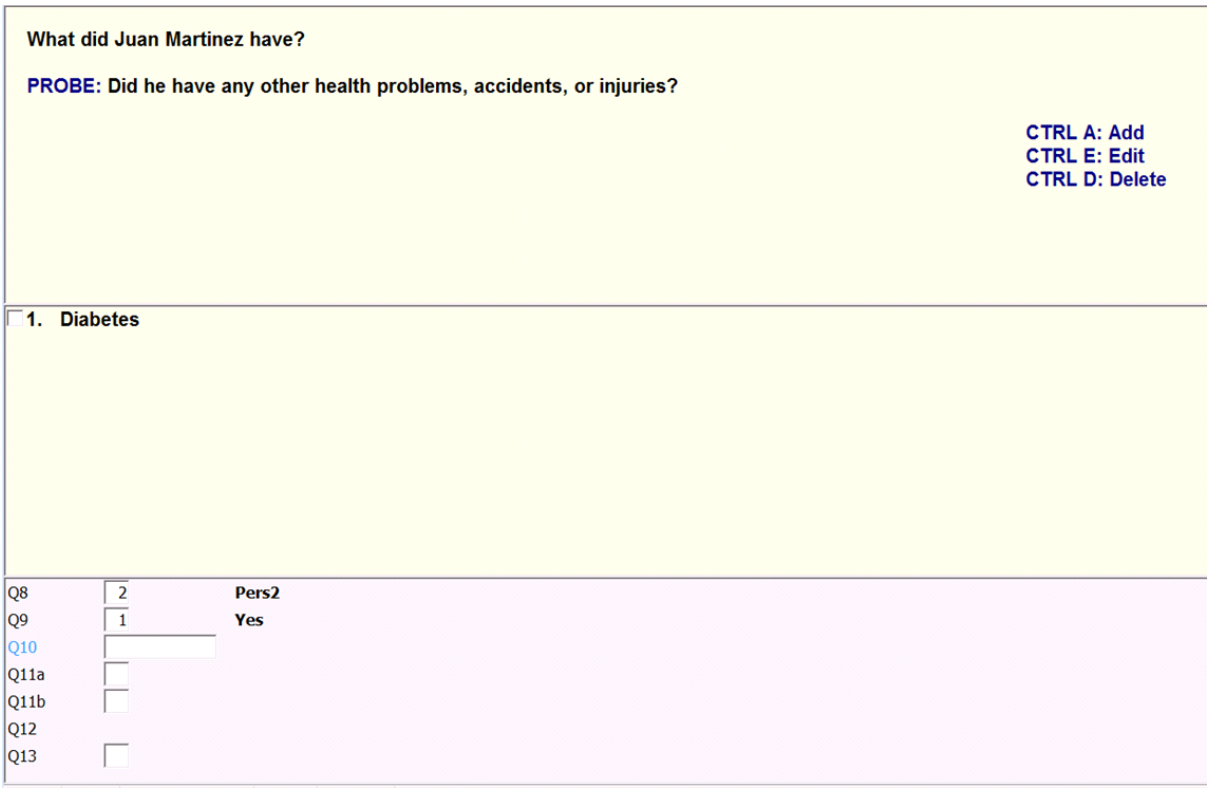
A roster is a list of items. Typical rosters include persons (such as a household enumeration) medical events or conditions, jobs held, etc. A nested roster is a list of items collected for each entry in a different list, such as the medical events for each person in the household. In some cases, entries in one list may be linked to another list, such as storing a medical provider for each medical event for

each person. In this case it is most efficient to keep a separate list of providers and store the index number identifying the provider on medical event entry.

In general, Blaise provides question types that allow for collection and management of rosters. One normally uses grids to collect the list, such as a list of persons in a household, and stores these in a person array. It is also relatively straightforward to display those persons as potential responses to a subsequent question, such as who helps the most with a particular activity. If necessary, those roster entries may also become the possible responses to a SET question to allow multiple responses. It is convenient in Blaise to store these lists in arrays where they are accessible and extensible.

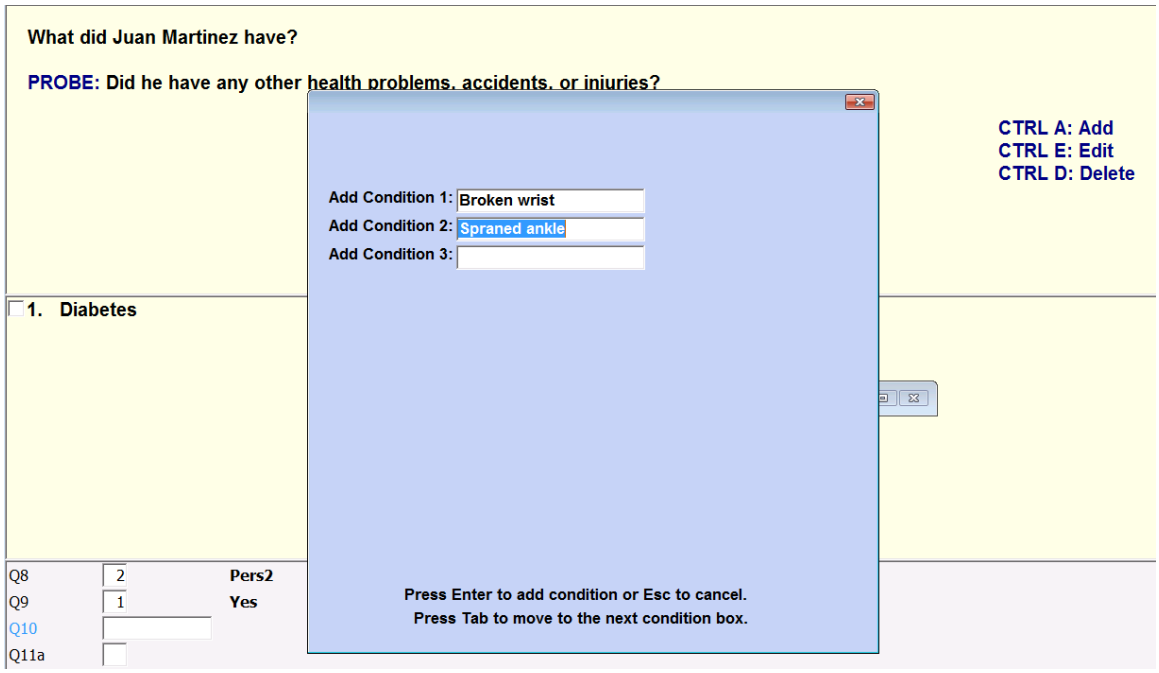
The challenges arise when interviewers are asked to manage and select list entries simultaneously. In an ideal world, an interviewer presented with a list could select one or more entries, or could add, delete, or modify those entries. In addition, it would be nice if interviewers were prevented from introducing error, such as deleting persons who had been selected on previous questions. Using DEP only, this can be difficult, and may require additional screens in order to separate the functions. However, we can use Manipula to allow list modification from the same screen that interviewers are using to select responses in the DEP. Manipula scripts can read from virtually any data source, , can display information and collect responses using dialog boxes, and can write back to any data source, including the record in memory.

Figure 1. Medical Condition Roster – Normal SET Question



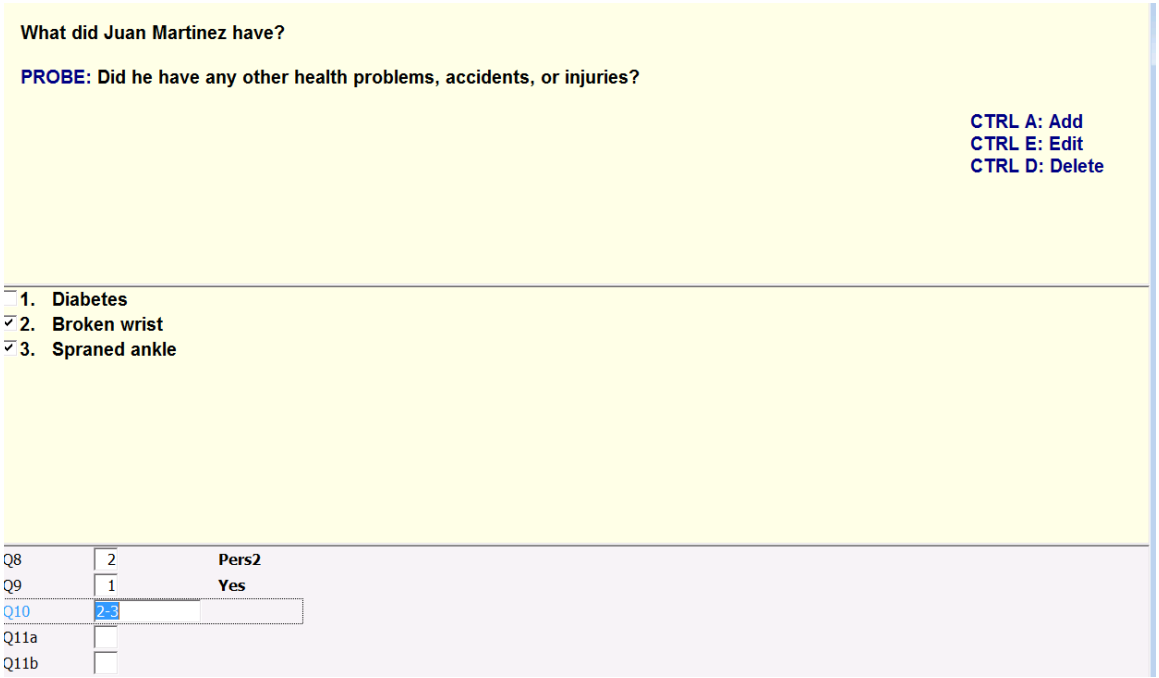
Here is an example of such a roster, in this case one for health conditions. The basic type is a set question; the answer options are fills working directly off an array with health conditions for this specific person. In this example there is one condition prefilled when we first get to this question. After pressing Ctrl-A, for Add, we get a Manipula dialog:

Figure 2. Manipula Add Dialog



We can add one or more health conditions as reported and either press <Esc> to cancel or <Enter> to add the new conditions and return to the roster question. After pressing <Enter> we'll see that the conditions have been added and selected at the original question:

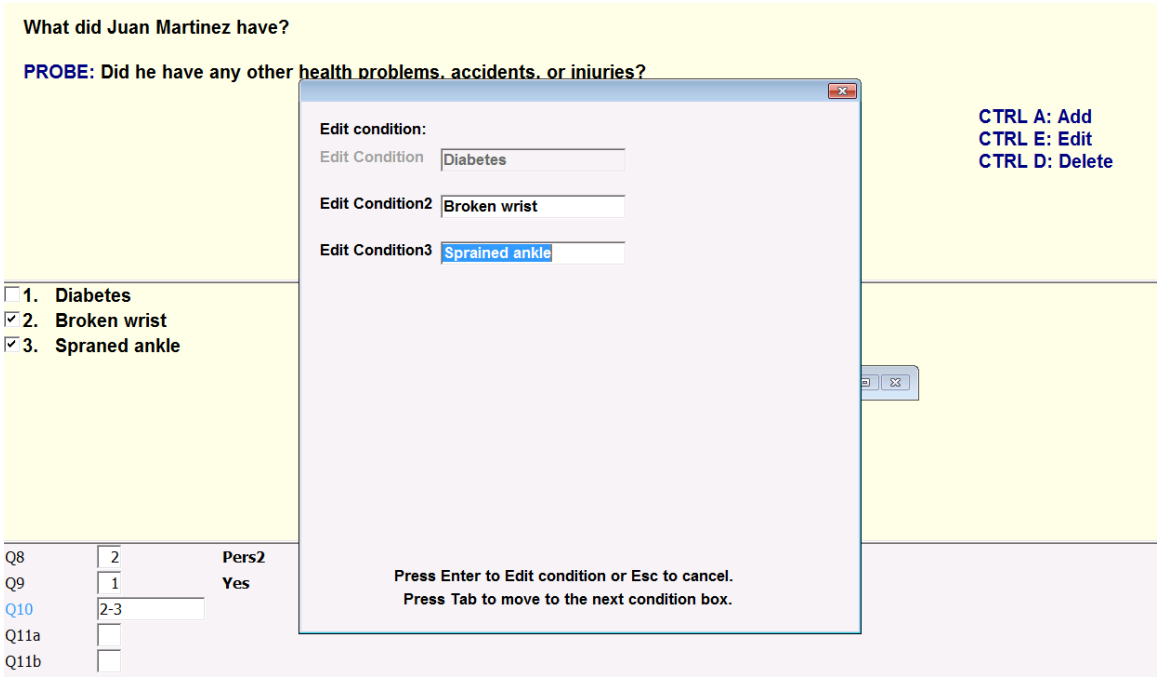
Figure 3. Updated Conditions Roster



Manipula code added the new conditions and included them in the set question answer, after pressing <Enter> in the dialog and before returning control to the DEP screen.

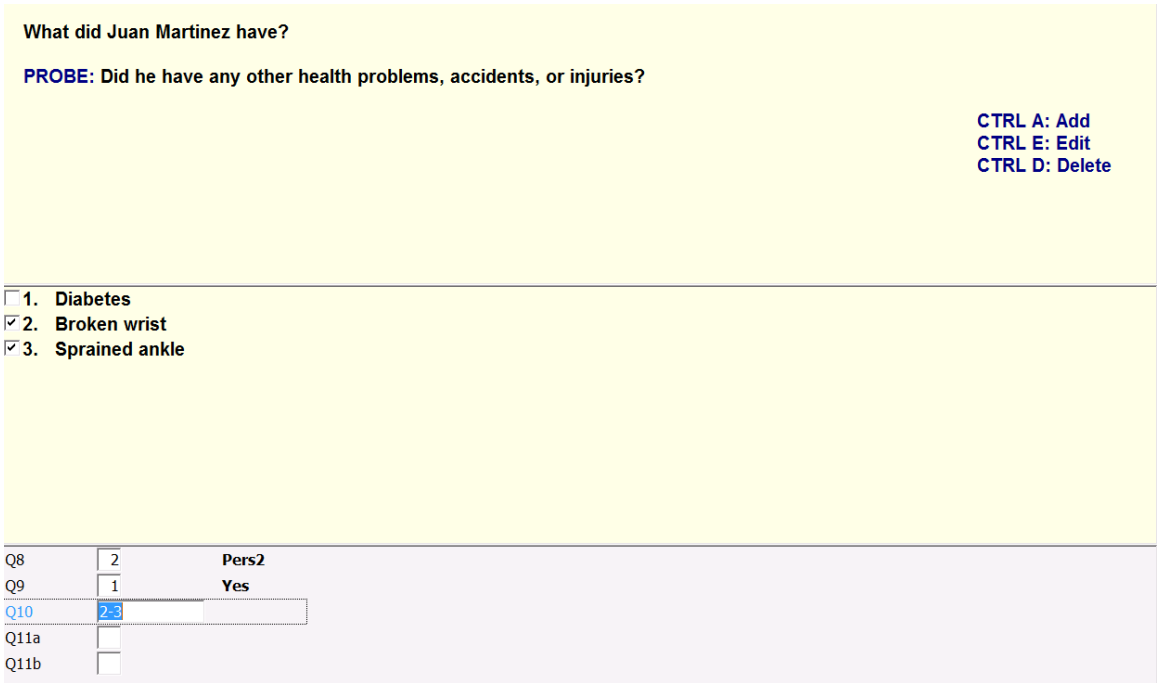
Now note the typo in one of the added conditions, 'spraned' instead of 'sprained'. To correct that we press Ctrl-E for Edit:

Figure 4. Edit Dialog



And after pressing <Enter>:

Figure 5. Fixed Typo



The same principle applies for deletion.

Note that in the dialog box for editing the first condition, Diabetes, is grayed out. It shows that we can control per screen which roster elements can and which ones can no longer be edited. The rules for editing can be set as desired, but it is useful and often important to be able to have such rules.

4. Challenge 2: Managing off-path data

Respondents sometimes provide data at a time other than when it is asked. In simple cases, the interviewer can simply back up to the appropriate item and correct the previous response. However, there are situations where this may not be the ideal solution:

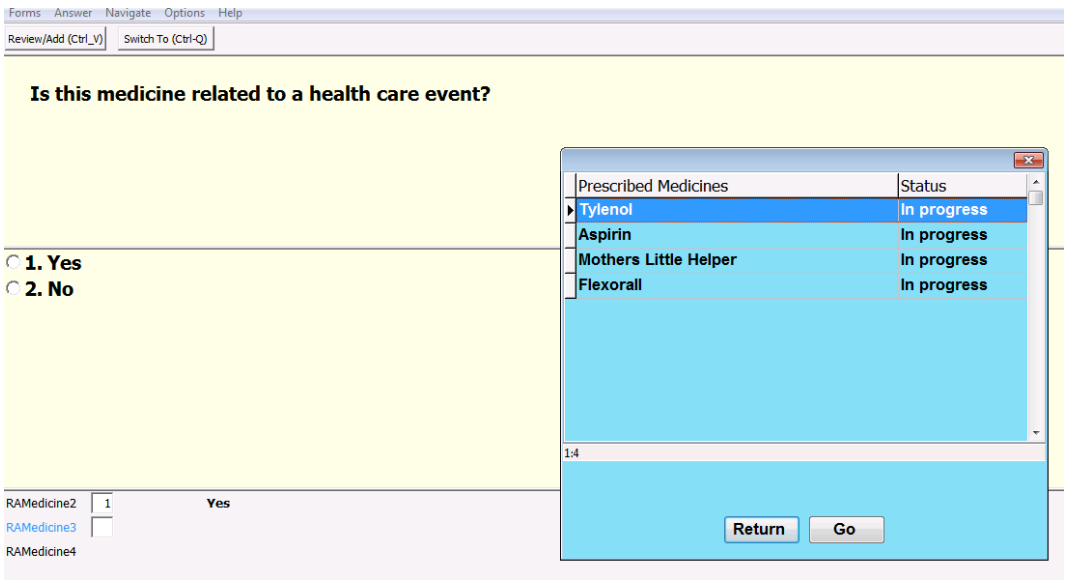
- It may be that the interviewer cannot back up because the data has been intentionally protected from corrections through the use of a wall or similar technique.
- Perhaps the information has been used to direct other main branches of the survey and the decision has been made not to allow changes. For instance, if the survey collects information about a school, then samples a few children, then collects information about each of those children, it may not be desirable to allow the interviewer to re-sample.
- In some cases the interviewer may require more flexibility, such as the need to pause a particular section, collect some other data possibly from a different respondent, and then return to the original point of departure. This can often occur in establishment surveys, where interviewers may need to speak with several respondents such as a principal, an administrator, and one or more teachers in a school.

Blaise does offer the use of parallel tabs, and these are useful in certain instances, but Manipula offers substantially more control. One recent project had a default path: first enumerate a household and then for each person, collect medical conditions, collect medical events and prescribed medicines, and finally collect jobs and insurance policies. However, interviewers required the flexibility to move to other sections as needed and collect that information, usually if one respondent needed to leave soon. Our solution was to use a menu button which led interviewers to the list of items that could be reviewed and updated. This single point of departure is organized and presented using Manipula.

This design has several advantages:

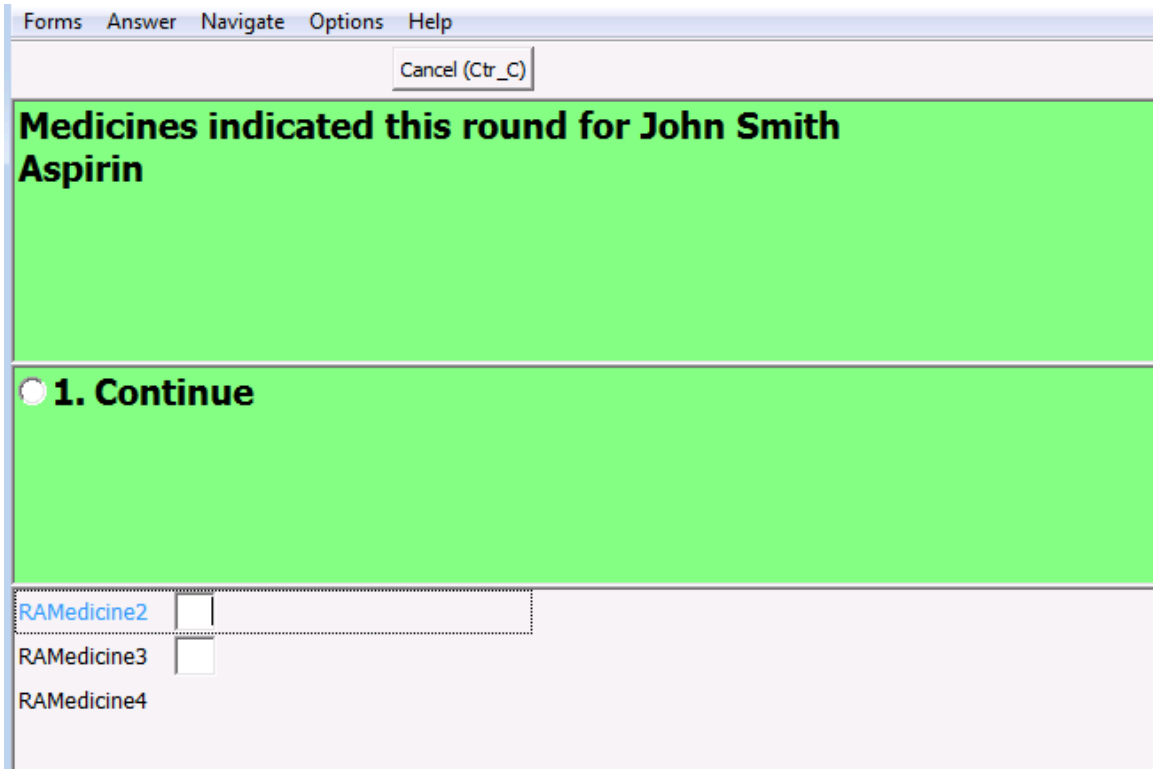
- We can collect data using dialog boxes, separate DEP datamodels, or external programs as appropriate.
- We can control where the interviewer goes. It may be appropriate not to allow the interviewers to enter sections, for instance, if the section has dependencies on another section that must be completed first.
- We can provide indications to help the interviewer. For instance, in the example below, the screens collecting detail for an entity added using off-path navigation have a green background so the interviewer expects to return to the original point of departure in the interview.

Figure 6. Status and 'Switch To' Popup



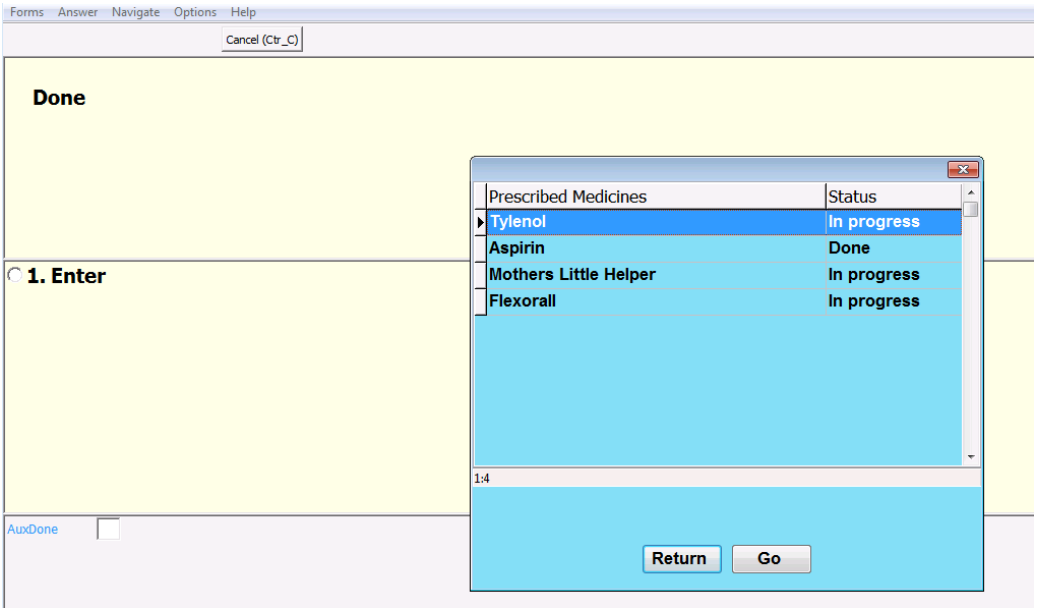
Here we have follow-up questions on every medication that has been collected earlier in the interview. Generally you have to go through these in order, but the above popup screen serves both as a status screen and as a way to deviate from the standard order.
When we select Aspirin, the second medication in this list, and then press Go:

Figure 7. Off-Path Questions



The green background indicates that the interviewer has departed from the normal route, so it's hard to miss that we're off the beaten path here. Note also that the two previously available buttons are no longer showing and only a Cancel button is available. Pressing the Cancel button just gets you back to where you left off. The other buttons were removed temporarily, so as not to confuse the interviewer with 'options within options'.
After answering the questions for Aspirin we return, first to the overview screen to show the new status, and then back to where we left off:

Figure 8. Updated Status and 'Switch To' Popup



Note the 'Done' status for Aspirin. The options now are to choose another row, or simply Return to go back to the questions for the first medication, Tylenol:

After answering the questions for Tylenol we get, via the normal routing, to the questions for Aspirin, which have been answered already. We can change the answers or simply press <End> to go to the questions for the next medicine.

This is a very simple example, but this principle can be applied in many different ways: with much bigger sections, or even allowing to choose selections on multiple levels (e.g. allowing to select a person to talk about all his/her medications, but also allowing to select one specific medication for person 2, and then return to person 1).

Alternatively, once the interview has gone too far beyond the normal spot to ask a section, you could allow a subset of questions when a respondent volunteers that they forgot to mention a job, a medication, a doctor's visit or even a member of the household.

5. Challenge 3: Using External Programs to Collect and Manage Data

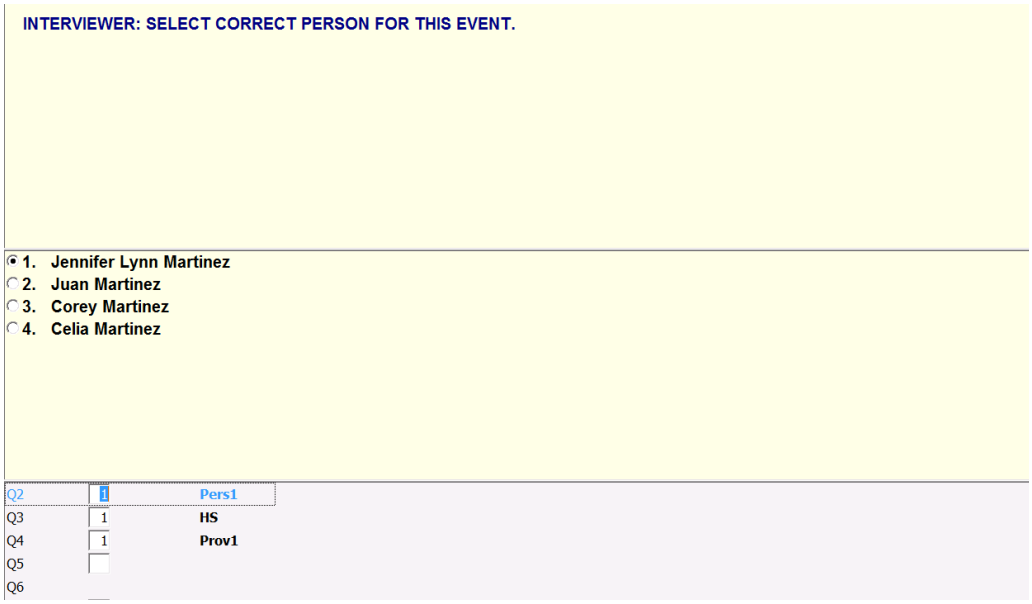
There are design requirements which sometimes call for input options other than Blaise data types. A common example is the use of a graphic element to collect data, such as a graphical date picker that lets the interviewer navigate to the correct date by selecting the correct day on a calendar rather than entering numeric date values. It is relatively straightforward to locate a third-party date picker and invoke it from Blaise using the ALIEN calls. However, we recently encountered a project with a more sophisticated set of requirements for collecting medical event dates:

- Bound the legitimate dates. In our example, we have a reference period and event dates must fall within that period.
- Allow begin and end dates for certain event types to contain an interval. For instance, a doctor visit occurs on a single date, but a hospital stay may begin and end on different dates.
- Select multiple event dates on the same screen. We did not want a time-consuming loop between the date picker and a question such as "Any more dates?". The requirement is that the interviewer collect all of the dates that respond to the question prompt at once.
- Allow the same date to be selected for more than one event. An individual may see more than one doctor on the same date.
- Allow events to be deselected if they were added by mistake on this screen.

- Enable interviewers to enter recurring events (such as dialysis) in the most efficient way possible.
- Store all events in an array in the current Blaise record in memory.

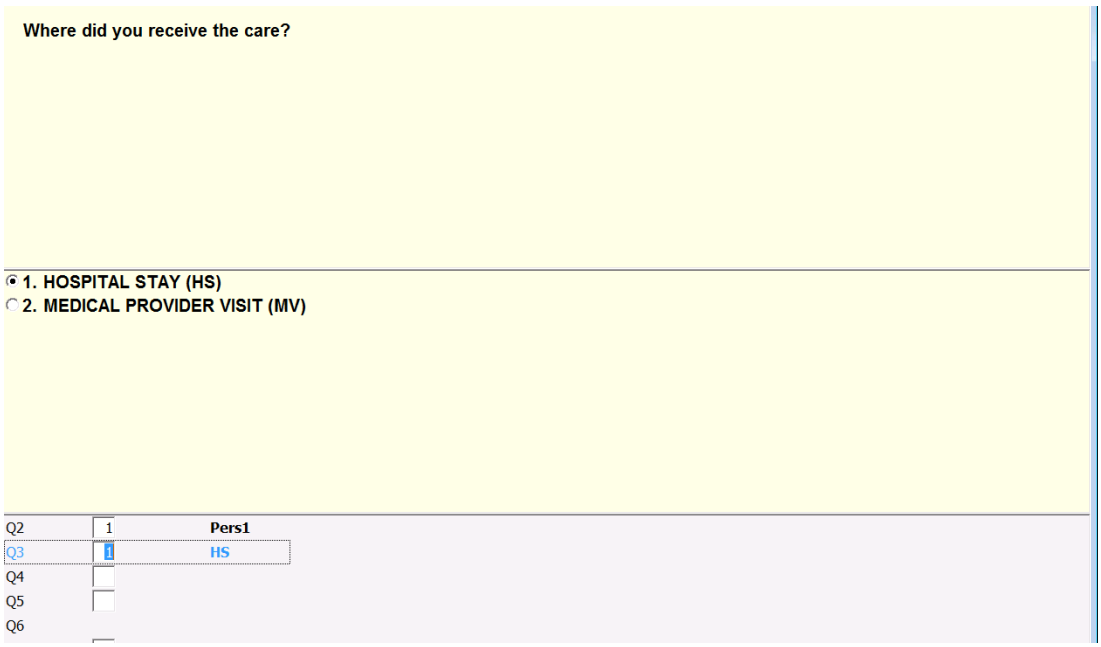
Our solution uses an external application written at Westat in C# using open source Microsoft .Net components. Interviewers may either select "Add an Event" in response to the appropriate question, or they may use the button described above to review and update events. Manipula computes all necessary parameters – reference dates, event type, previously-entered events, etc. – and sends them to the .Net application at call time. When the interviewer leaves the date picker, Manipula accesses all the new events and writes them into the medical event array in the Blaise record in memory.

Figure 9. Select Person



Here is an initial screen, just to select the correct person for a medical event.
Next we select the type of event –

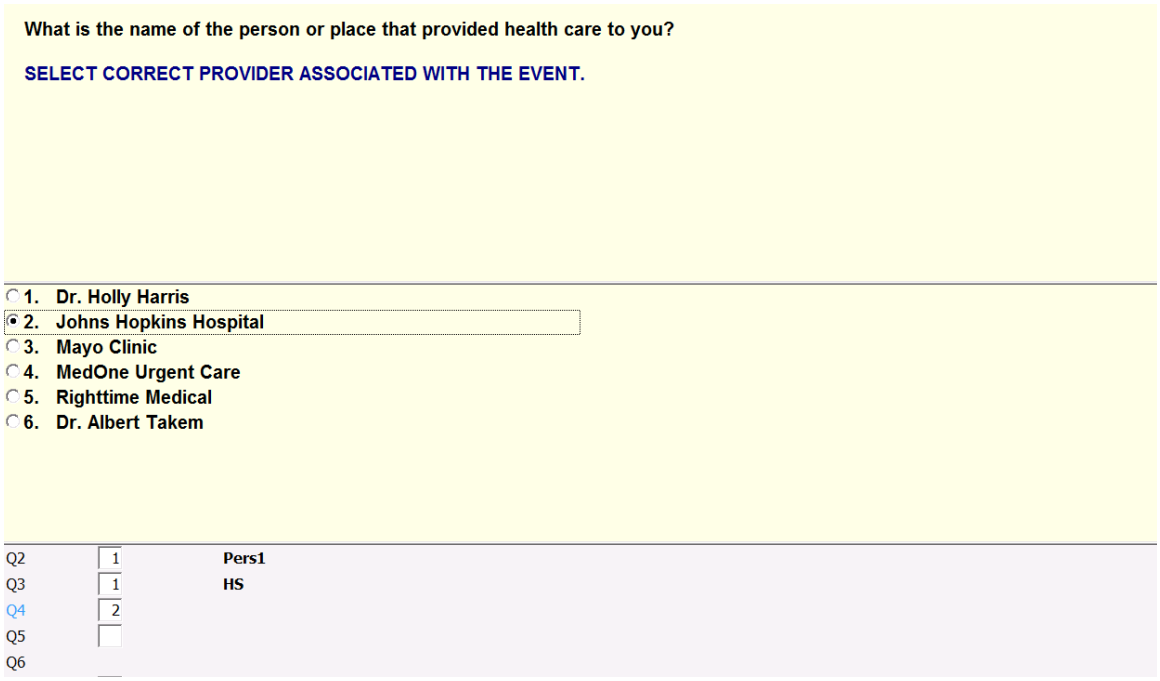
Figure 10. Select Event Type



In this small example there are only two options, as each has its own ‘calendar behavior’.

Next we select the medical provider for the event:

Figure 11. Select Medical Provider



Then we get to the screen that starts the external executable:

Figure 12. Screen to Launch External Executable

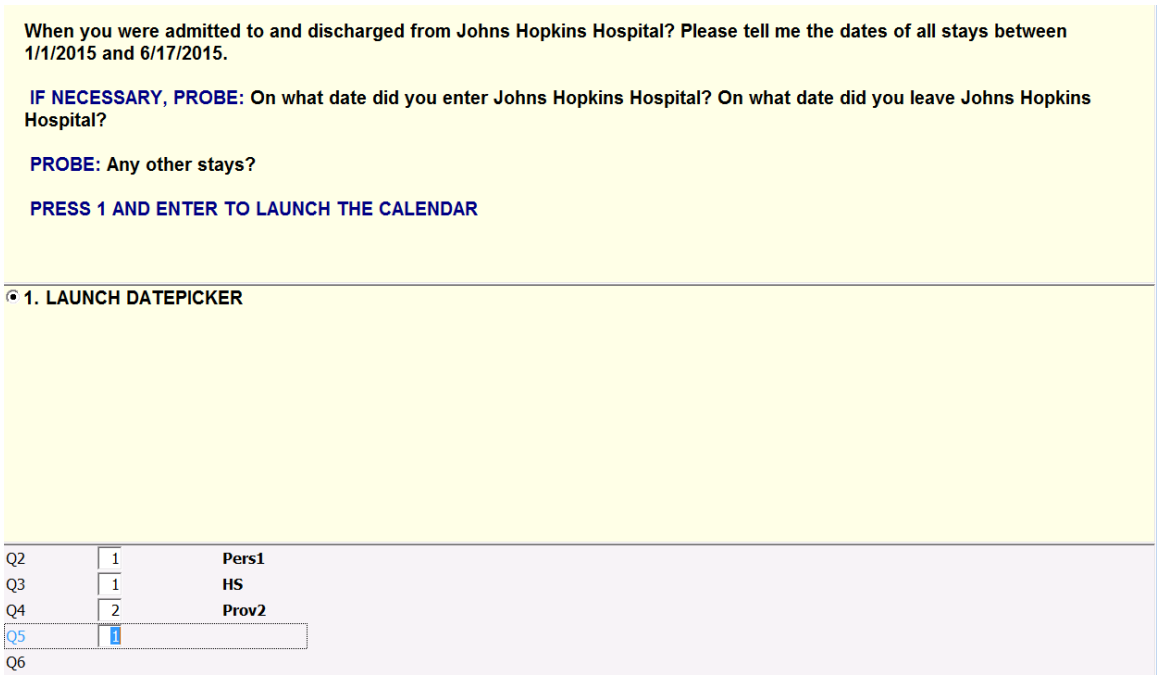
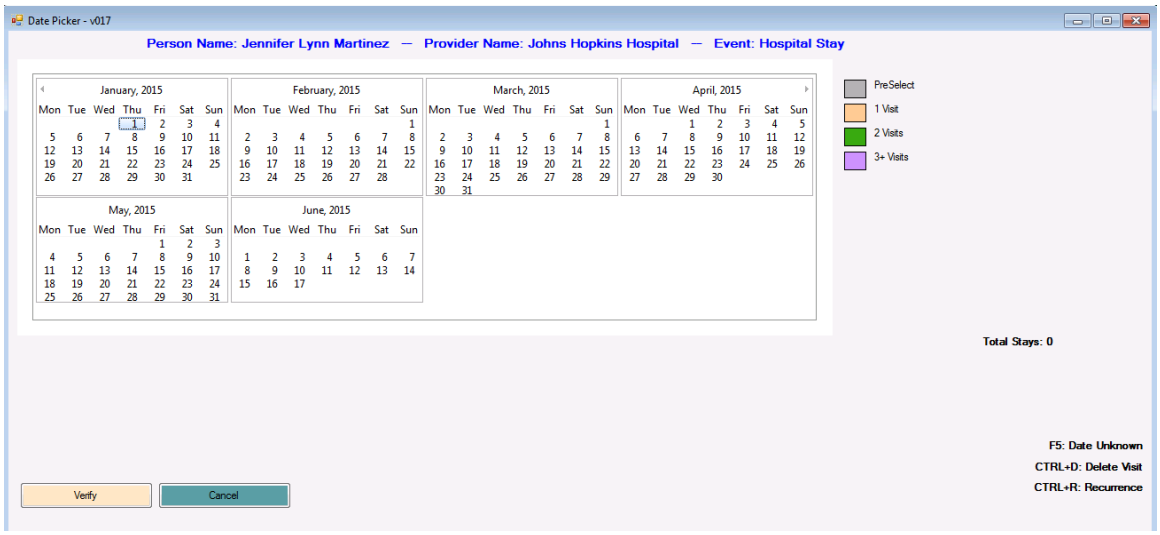
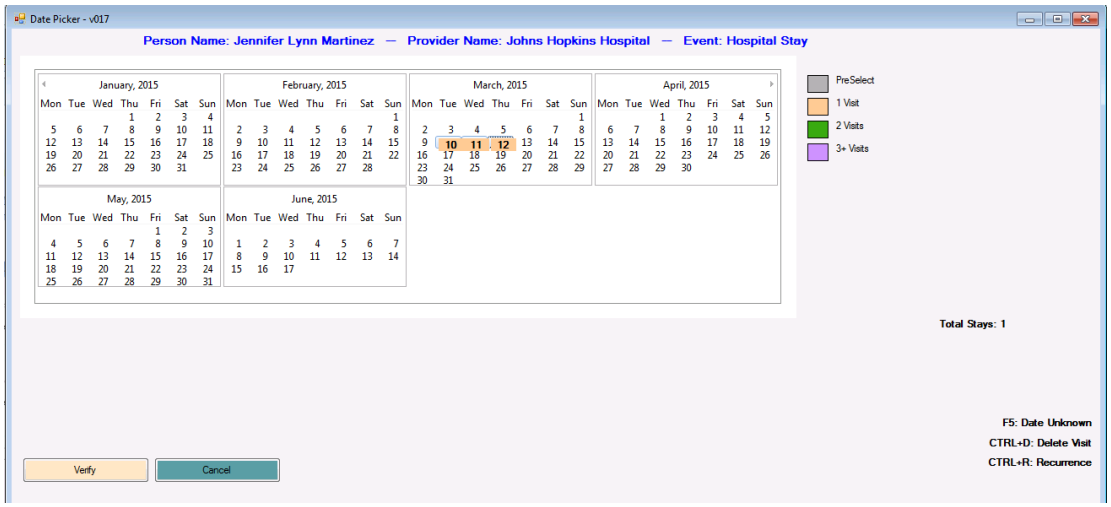


Figure 13. Date Picker



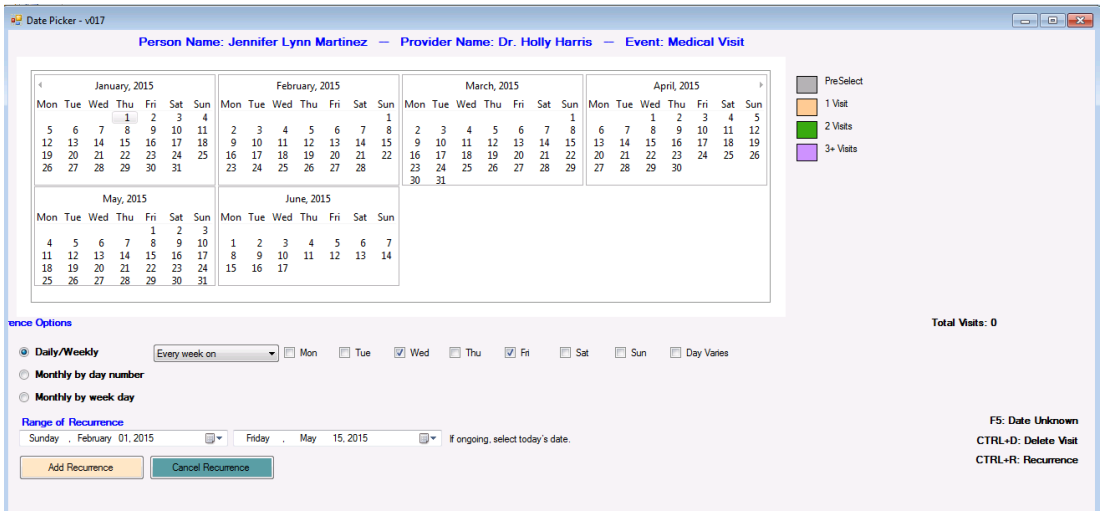
Note the 'context header' line at the top, and that the complete reference period for this person shows up at once. It makes it easy to select the correct dates, while automatically safeguarding against entry of dates that are outside of the reference period and therefore not allowed.
Both mouse and keyboard can be used for selecting dates. Here is an example of one hospital stay as selected in the calendar:

Figure 14. Sample Hospital Stay



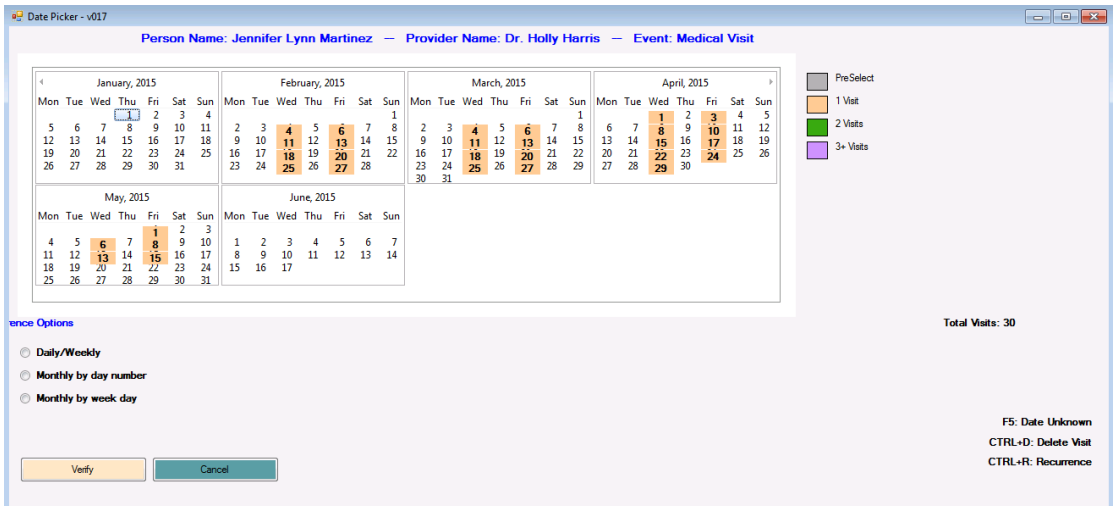
Here's an example of the recurrence feature for single date events, adding many events at once instead of one at a time:

Figure 15. Recurrence Feature for Single Date Events



After pressing the Add Recurrence button:

Figure 16. Add Recurrence

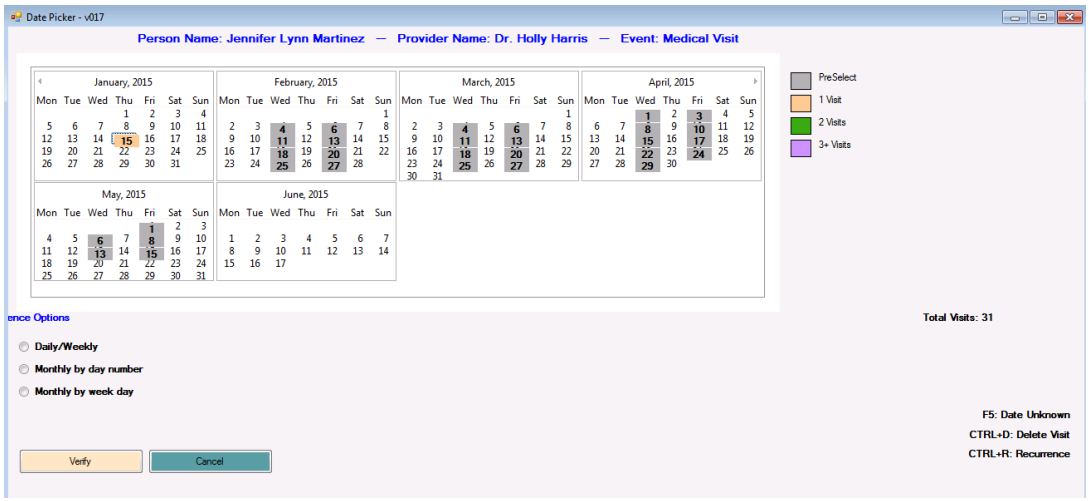


As is shown with the counter on the right side of the screen, this added 30 medical visits at once.

Just like with the example of roster popups earlier, Manipula procedures are doing the work behind the scenes, passing to the Calendar executable any info it needs and writing the selected dates to an array in the main datamodel.

To show that the dates are written correctly and that rules can be enforced for previously entered event dates, here's the Calendar screen for the same person-event type-provider combination, for a second time, showing the previously entered dates, now non-editable:

Figure 17. Non-Editable Dates



6. Conclusion

There are some considerations to be aware of when using these capabilities as described above. First, data collected or edited using Manipula procedures are not subject to the selective checking mechanism that governs fields collected directly in Blaise. This blend of techniques – selective checking for DEP and traditional batch processing for Manipula routines – requires the careful development of good design and programming practices. Second, to the extent that the Blaise datamodel is used to document or direct additional survey activities beyond data collection, the use of Manipula routines may have significant implications. Examples include the use of the datamodel to support data editing, metadata systems, or other downstream processes.

The examples we have shown are the tip of the iceberg – the ability to use Manipula to access the DEP record in memory along with any other data source puts solutions to many challenges within reach.