Building a Data-Driven Computer-Assisted Interview Using SAS/AF®

Derek Morgan, Covidien

ABSTRACT

Creating a respondent-usable survey using the SAS® System may require significant developer time, using traditional SAS System methods of data capture with an end-user interface, since the effort needed is directly linked to the size of the survey. In such an environment, each survey has to be custom built and coded. We developed the method described in this paper to bypass those and other obstacles related to end-user data capture. This method uses SAS datasets to implement program control and provide questions and responses. A fixed series of SAS/AF frames interact with the user, yielding a consistent user interface. The frame code does not change between surveys—the datasets do. However, if the user needs to create a custom frame to handle the administration of a particular question, it is easily integrated into the existing method.

This method was initially developed to administer the diet history questionnaire (DHQ). The DHQ is a food frequency questionnaire developed by staff at the Risk Factor Monitoring and Methods Branch of the National Institutes of Health. It consists of 124 food items and includes both portion size and dietary supplement questions. The 124 food items comprise 404 questions with extensive skip patterns. However, there was no instrument available other than a paper form to administer this questionnaire. Traditional data entry methods using SAS/FSP® or SAS/AF would have required a disproportionate amount of programming resources. This paper will use the prototype application to illustrate the development of the method and its basic features.

DISADVANTAGES OF THE TRADITIONAL SAS/FSP METHOD

Think about creating either an FSEDIT or a PROGRAM screen with 407 fields. Think about typing the full text of each question, along with the possible responses for each question into the screen. The FSP application would have one labeled section of code for each field in addition to the INIT, MAIN, and TERM sections in a full-text, tightly controlled SAS/FSP application. The SCL code would be at least a thousand lines. A single SAS/AF FRAME hardly improves the situation. All of these methods can require a great deal of programmer time during maintenance, too. What happens if text in the questionnaire changes or a question is added or deleted? All of this adds up to a maintenance nightmare for something that already has a high development cost in terms of programmer hours. While the traditional method[1] is still a valuable tool for small projects and questionnaires, the parameters of the DHQ clearly required an alternative method.

Another disadvantage is that FSEDIT screens are not re-usable. They may be recycled, but if anything changes from project to project, the screen will require some amount of reprogramming or redesign. Again, more time is required for development. Part of the issue is rapid deployment of a series of questionnaires. It does not take very long to modify one questionnaire or even build a short one from scratch using FSEDIT. However, if you have a series of forms for data entry, or one of those forms requires a great deal of coding, FSEDIT begins to lose its attractiveness as a data capture tool.

DEVELOPING THE METHOD: SYSTEM ANALYSIS

The first thing we had to do was look very carefully at the questionnaire to see what we were trying to emulate on the computer. When you go through the DHQ, the first thing you notice is the survey questions fall into distinct categories: "How often did you eat or drink {food item}?", "Each time you ate/drank {food item} how much did you eat/drink?" and "Which of the following {food items} did you eat/drink? (Select all that apply)." In addition, most of the questions begin with the phrase, "Over the past 12 months..." This commonality allowed us to look a little closer at the types of responses that are expected.

Every response to each food question falls into one of two categories: single-response (SR), or multiple-response (MR). However, one set of SR questions requires a different type of flow of control than the rest of the survey. The only other questions asked in the DHQ are three demographic questions and, of course, a variable will be needed to keep track of unique respondent identifiers. When you look at the survey in detail, you see that many questions use the same set of possible responses. We did a careful check, and discovered that there are 126 unique sets of SR choices. There are seven unique sets of MR choices, with the maximum number of responses for any multiple-choice question being 24.

The questions flow in a distinct order, with skip logic to bypass the quantity questions when foods are not consumed. The re-use of response categories and the similarities between questions allow us to factor the common elements, and that leads to the possibility of using metadata for both questions and responses. Although the previously noted exception requires special handling for its flow of control, it does use a common set of SR choices and therefore can still be driven by response metadata.

Both questions and responses are contained in their own tables. The question dataset (QDS) is what drives the application. It needs to contain all the information necessary to dictate flow of control, the set of possible responses associated with each question and data capture information. The response dataset (RDS) is only responsible for the response text of each item and the values associated with that text.

METADATA TABLE ONE: THE QUESTION DATASET

Beyond the obvious question text and corresponding response set, what did we need in this table to drive an application? First and foremost we needed an order, since the DHQ is an ordered questionnaire. While the question dataset has an ordering variable, it is not responsible for the survey's flow of control. In production, the ordering variable is only used to sort the dataset for production of hardcopies of the metadata in question order. By not using this ordering variable for the survey control, we allowed for the insertion/deletion of questions without regard to their physical location in the table.

There is skip logic in the DHQ, so the application needed to know where it is going and what makes it jump. That means it also needed to know where it is, and where it would go if it the skip pattern was not invoked. We created a variable for the question number to identify each question, separately from its order. If we were to use the order variable, changing the questionnaire would require altering the order variable for all records subsequent to the added/deleted record, and re-sorting the QDS. By using this question number variable for survey control, the only records that needed changing were those prior to and following the addition/deletion. Normal survey flow of control required a variable to dictate the next question in the survey sequence. In addition, since this was to be a computer-assisted interview, allowing a respondent to go back and change an answer would be a good thing, so we created a variable to represent the question previously asked. Both of these variables assume there is no skip logic, or that the skip pattern was not triggered. Performing this skip pattern required two variables as well: one to define the response that triggers the skip pattern, and another to define the next question in the survey sequence when the skip pattern had been invoked. Together, these four variables defined the entire flow of control within the survey from the QDS.

After the survey control problem was solved, we needed a way to tell the application where to store the response. We created a variable to hold the name of the variable in the data capture table (DCT). MR questions were stored as yes/no variables, one per possible response. This allowed us to use the variable prefix in the QDS, and avoid having to define each variable that would contain a possible answer, creating multiple records for a single question. There were two more variables added to the QDS related to application control. One variable indicated the single response in a MR answer that would cancel any positive responses, e.g., “none of the above.” The second indicated whether the prefix phrase was to be displayed with the question text. Finally, we added a variable for text capture; it indicated whether the value entered in the text frame is to be stored as a character or numeric variable in the DCT. This was the list of variables in the QDS resulting from our systems analysis of the survey.

TABLE OF VARIABLES IN THE QDS

order	Num	4	Order that questions should appear (used for sorting ONLY)
q_no	Char	32	Question number (used by program)
q_text	Char	300	Question text as it is to be displayed
Rcat	Char	32	Response category (R=single response, M=multiple response, T=text field)
skip_to	Char	32	Question number (q_no) to skip to
skip_trigger	Num	4	Answer to trigger skip pattern
prev	Char	32	Previous question in sequence (q_no)
next	Char	32	Next question in sequence (q_no)
vname	Char	32	Dataset variable name associated with question
Exc	Num	3	Exclusive answer in multiple-response category (ONLY used with M)
type	Char	1	Char or Num variable (for text (T) answers)
use_common_text	Num	3	Turn pre-text box on or off?

One difficulty that occurred in programming this method was determining exactly which question was the last asked. It could not be assumed that the value defined in the QDS for the previous question was the last question asked, because the previous question in the standard survey sequence may have been skipped due to the survey’s skip pattern. It would be confusing to many survey respondents if they had to press backwards more than once to find the last question they had answered.

It turned out that the procedure to move backwards was not very complicated. It only required two additional steps, and was still driven by using the question metadata. Here are the steps you would take:

1. Determine if the current question is a possible skip destination by checking to see if the current question is a value in the skip destination variable. If not, use the value in the "previous question" variable.
2. At this point, you know which record points to the current question as a skip destination. Get the question number for that record, as well as the values for the skip trigger and the DCT variable name from it.
3. Test the value for that variable in the DCT. If it is equal to the skip trigger, then go to the question number obtained in step two above. If not, use the value in the "previous question" variable.

This sends the user back to the previous question answered, not the previous question in the survey sequence, and is accomplished by using the variables in the QDS and the DCT.

METADATA TABLE TWO: THE RESPONSE DATASET (RDS)

The second piece of metadata we needed was the RDS. Where the QDS contains one record per question, the RDS has one record per possible response, and contains all possible responses for the entire survey. Records can be placed in any order in the RDS; there are two key variables that are used in WHERE clauses. The key variables are response class and then, within each response class, response index. Response class groups responses together so the application can load all the possible responses for a given question. Response index provides not only the display order for the items in each response class, but also the application's link to the data values themselves. This application only used two more variables from the RDS. One contained the full text of each response as it should be displayed on the screen. The other was the actual data value to be written to the DCT when a given response was selected. The remaining two variables were used to sort the RDS for hardcopy documentation purposes. Response class designations begin with the letter "R" for a SR category, or the letter "M" to indicate a MR category. This tells the application which module is responsible for displaying the question and capturing the data. The list of variables in the RDS is below.

TABLE OF VARIABLES IN THE RDS

rcat	Char	32	16	Response class
indx	Num	4	8	Response index
val	Num	4	12	Value to be written to data capture table
resp_text	Char	200	48	Response text to be displayed
rtyp	Char	1	248	Response type (R or M) (for documentation purposes)
rnum	Num	8	0	Response class number (for documentation purposes)

Together the two metadata tables provided enough information to drive an application that conducts a survey. Now that we had the parts and fuel for the engine, all we had to do was design and build a frame for it.

CREATING SAS/AF FRAMES FOR THE METADATA ENGINE

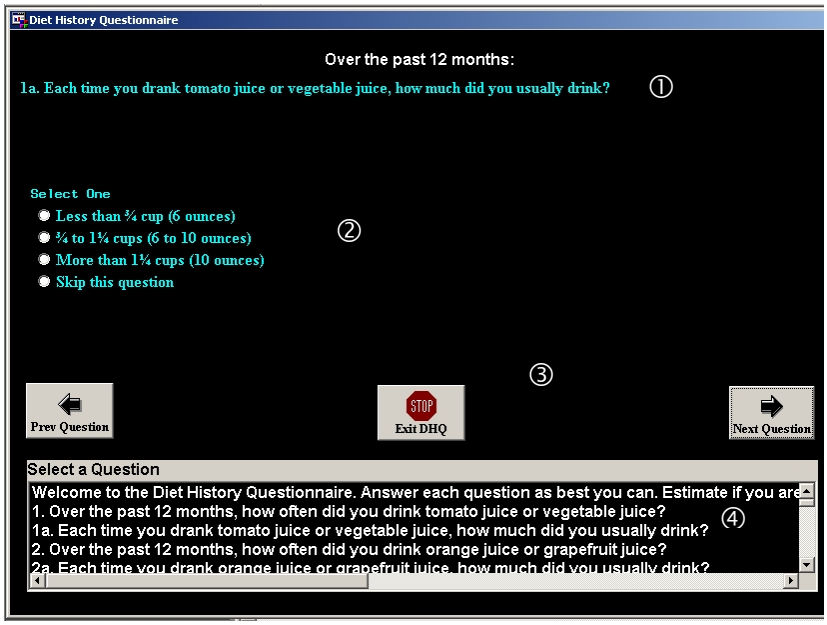
The first step was to visualize what we wanted to display. We needed a frame to display questions and their possible responses. Since we had two distinct styles of responses, it was easier to build two frames rather than try using one frame with alternately disappearing objects. The next question asked was, "What, in addition to the question text and possible responses, do we need to show a person who is taking a survey?" Even though the application itself was going to handle the default flow of control for the survey, the respondent should be able to move forward and backward during the survey. Backward motion was critical so that a respondent could change a previous answer if necessary. The respondent should also have a way to leave the application gracefully before the survey is finished; longer surveys (the DHQ takes at least an hour) might require more than one session. The ability to go directly to a specific question would be useful in an interviewer-administered setting or for data entry from paper.

We worked on the frames one at a time. The SR frame was incorporated into the main program module. After careful consideration, the MR frame became two MR frames. One frame displays two columns of up to 12 response choices for a total of 24, while the other frame displays a single column of up to 12 choices. This allows display of longer responses using full screen width. A utility text-response frame was also built for data that is to be typed, although it is not used in the DHQ application as distributed.

SINGLE-RESPONSE FRAME

The main control frame incorporated the administration of SR questions because the majority of the questions on the DHQ were single responses. At some point, the application will be re-engineered to separate the administration of SR questions from the application control. It currently works fine in production, so the change is a low priority.

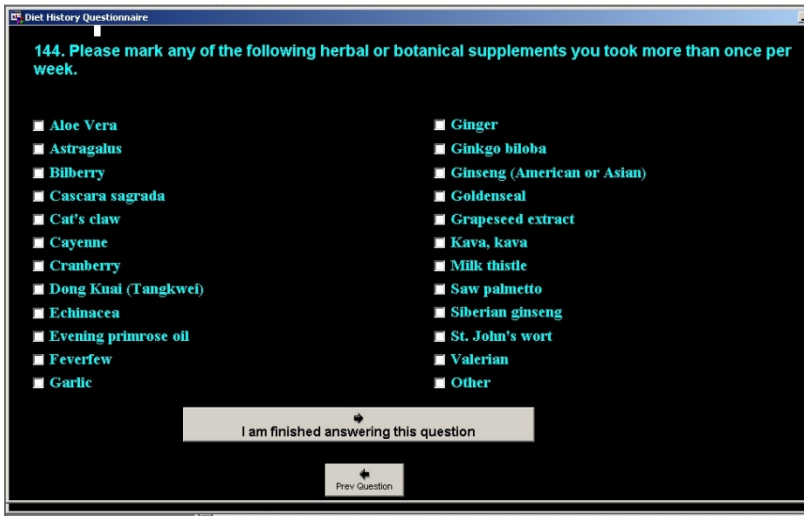
Single-Response Frame



The SR frame looks like this. The question text is at the top (❶). The question prefix is centered, and in white. It uses a different font than the rest of the frame to stand out visually. A variable in the QDS controls whether it is visible or not. The blue text in ❶ is the question text as specified in the QDS. The question area can run the entire width of the frame, but this particular question does not use the entire space. Item ❷ is the response box area. This area also encompasses the width of the frame, since there are several questions with more than one column of possible single-choice responses. The list that fills the radio box is generated dynamically from the RDS. The value of response category in each QDS record defines which records from the RDS are used to create the SCL list to be used. Area ❸ is the navigation area. It contains push buttons that cause the application to go to the next or previous question, and a stop button. The forward and backward buttons are only displayed if the survey context allows; that is, the last question shows no “next” button, while the first question shows no “previous” button. The combo box at the bottom (❹) is intended for administrative use only, and allows direct access to any question in the survey. It can be made invisible based on a toggle. In production for the general population, the toggle is a check box in the opening frame; in our study, the toggle was based on the user's authorization level.

The elements required for MR questions are the same as for SR questions with one exception. Instead of getting the response from a radio box of mutually exclusive options, multiple answers are obtained from a list of check boxes that can be independently checked. The respondent also must explicitly close the question. It is implicit that selecting one answer in a single-choice will cause the survey to advance; the question has been completely answered. In a MR situation, there is no way of knowing how many check boxes the respondent wants to select. Therefore, the respondent must take some action to signal all desired choices have been checked.

Multiple-Response Frame

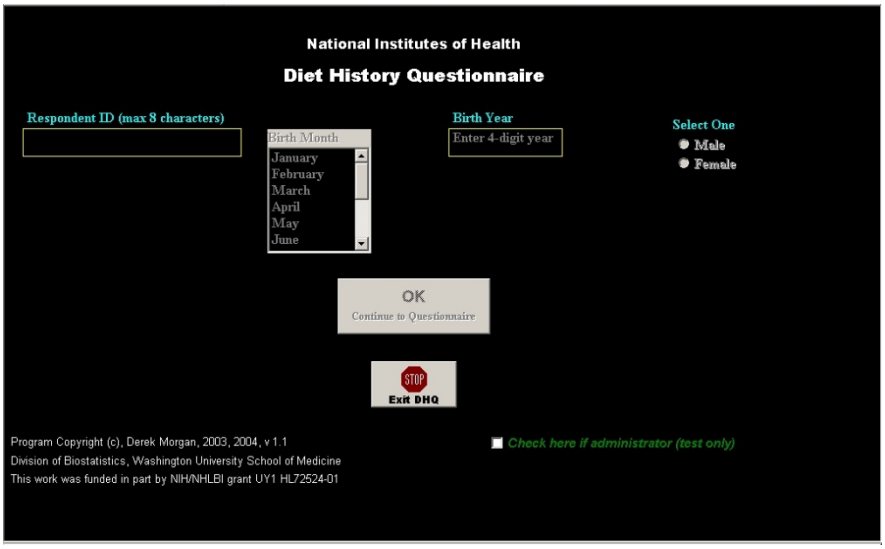


This screen shot is the two-column, 24-item check box version of the MR frame. There are twenty-four choices shown, the

maximum number of choices for MR questions in the DHQ. The two 12-item columns are populated with a list from the RDS. The left column fills first, from top to bottom, and then the right. The question text from the QDS is at the top. Note that the preferred push button option at the bottom, “I am finished answering this question,” is indicated by its size. The 12-choice, single-column version is identical in appearance except there are only 12 check boxes that fill the width of the screen. These three frames handle the bulk of the application, directed by the QDS metadata table using response information from the RDS.

Five other frames in the general application serve specific functions. The START_DHQ frame (shown below) is where the application starts. It obtains the necessary demographic information and is responsible for initializing the record for the respondent in the DCT.

START_DHQ Frame



The TEXTRESP frame (mentioned earlier) captures anything a respondent types. It can handle both numeric and character data up to the character limit in the SAS System. Both unformatted and formatted inputs are accepted, with the caveat that any formats and informats must be defined in the DCT with the corresponding variables. The frame uses two text entry objects: one for numeric data and one for character data. They overlay each other in the frame, and the inactive one is invisible and disabled. Determination of which one to display is controlled by a variable in the QDS. TEXTRESP has one limitation, however; there can be no skip logic associated with the text response.

There is also a custom frame in the prototype application. It asks the questions and obtains the responses for one specific set of DHQ survey questions whose flow of control cannot be handled by any of the main frames. It still utilizes both the QDS and RDS to display questions and capture data. The application easily accommodates custom frames as long as the custom frame adheres to a naming convention. Another utility frame, IDIOTBOX, forces the respondent to confirm any change in an existing answer to a SR question. Finally, there is the frame that generates the scoring file.

THE ENGINE

The overall procedure is surprisingly simple. The start-up frame creates the subject record in the DCT, if necessary, and writes the demographic information the user has entered. If the subject's record already exists, it reads the information and then calls the main control frame.

At initialization, the main frame uses the QDS to create the list populating the direct question access list box, and decides whether it is to be displayed. The last initialization task is to determine where to start the questionnaire. The DCT has a variable that contains the name of the current question. If it is blank or the respondent has completed the questionnaire, the questionnaire starts at the beginning. Otherwise, it starts at the question stored in the DCT.

In MAIN, the previous and next buttons are turned on and off according to the question (first question, no previous button; last question, no next button.) If it is the last question, the frame terminates.

Now that the set-up is complete, the survey can start in earnest. First, the QDS is opened using a WHERE clause based on the current question number, and the desired record is FETCHed (using CALL SET to make sure all variables in the QDS are available to the SCL program.) The progam then closes the QDS.

The next action is dependent upon the response category type. If it is MR, text-response or a custom-response, control passes to the appropriate frame. This can be one of four: an MR frame, the text response frame, or a special frame to handle specific cases not addressed by the existing response frames. Otherwise, control remains in the main control frame.

SINGLE-SR AND NO-RESPONSE QUESTIONS AND THE CONTROL FRAME

The control frame handles SR and no-response questions. This is not necessarily good programming practice, but a time-saving shortcut developed as part of a much larger production application with a very tight deadline. First, the common text is turned on or off, triggered by the USE_COMMON_TEXT variable in the QDS, and then the question text is displayed. If the response category variable in the current QDS record has no value, then the response-handling portion of the program is skipped. This allows "display only" questions, where the respondent is supposed to read text without answering before proceeding. The response radio box is then made invisible, control is returned to the application and the respondent will have to press the “Next Question” button to proceed.

If a response is warranted, the response radio box has to be displayed and populated. The RDS is opened with a WHERE clause that subsets records based on the response category variable from the QDS. The RDS records create an SCL list. One more response option, “Skip this Question,” is added to all SR questions by appending the option to . This is done by appending it to the SCL list in the frame code rather than by adding an extra record to each response class in the RDS. This SCL list then becomes the **list** property of the radio box.

Now that the response radio box is populated, we check for an existing answer from the DCT. If there is one, applying a WHERE clause to the already-subset (by response class) RDS allows us to get the value of the index variable based on the value stored in the DCT. The value of INDX becomes the **selectedIndex** property for the radio box, which is otherwise left blank. At this point, the RDS closes and control returns to the frame to await respondent action. Upon respondent selection, the application checks to see if it modifies an existing response. If so, the IDIOTBOX frame is called and obtains the user’s response to the question of whether or not this response should be modified. If not, control returns to the frame for a different respondent action.

If it is not a modified response, or the user has confirmed the modification, the response is processed. The RDS opens with a WHERE clause that subsets the RDS on the response class and index value returned by the radio box. GETVARN() is used to obtain the actual value to be written to the DCT. If the response was “Skip…,” the value to be written is a special missing value defined in the code.

The next step is to write the value to the correct variable in the DCT. Only three variables are available to the DCT when open:

- The ID, necessary to select the correct DCT record
- The LASTFLD variable, telling the application where to re-start the survey if it is stopped
- The variable where the data should be stored, obtained from the QDS by getting the value of VNAME

The response radio box is then disabled to prevent any accidents. Finally, the next question to be displayed is determined based on the SKIP_TRIGGER, SKIP_TO and NEXT variables from the QDS. If the response is not the same as the value that triggers the skip logic, or there is no skip logic (SKIP_TO=.), the next question is represented by the value of NEXT. Otherwise, it is the value of SKIP_TO. If skip logic is triggered, missing values are written to the DCT for all variables corresponding to questions that have been skipped.

The loop iterates until the final question. The metadata dictates the entire flow of control for the survey. There are no specific instructions inside the code about any specific field or survey question, except for the exit point. The value of Q_NO for the last question must be "fini".

MULTIPLE-RESPONSE QUESTIONS

MR questions are handled in a similar, although greatly simplified, manner. The main control frame determines which of the multiple-response frames to use by finding the number of responses for the response class. If there are twelve or fewer choices, it will use the single-column frame. Instead of using a single-answer radio box, either 12 or 24 check box components create a single object in the frame. Because dynamic allocation is not available, the object is defined as having the maximum number of individual components for each of the MR frames. Each check box component is disabled and invisible at frame initialization.

Both the QDS and RDS are opened within the MR frames. The QDS provides the question text, response category, DCT variable name (which serves as the prefix of the DCT variable names in these frames) and the index for any exclusive answer. An exclusive answer is one that locks out all other choices. For example, “None of the above” is an exclusive answer. Survey flow in the MR frames is restricted to forward and backward, and handled in the main control frame via a RETURN value.

The MR frame starts by receiving two parameters, question number and subject ID when it is invoked with DISPLAY(). Question number is needed to use QDS metadata, and subject ID is used for writing the data to the DCT. First, all of the check box components are assembled into a single object, making array-style referencing of individual check box objects possible. Next we get the information needed from the QDS by opening it with a WHERE clause and fetching the record. The QDS is closed, and the **label** property of the question-text text box object is set to display the survey question.

The next step is to get the information from the RDS. As with the single-response frame, it is opened with a WHERE clause to subset the RDS and restrict it to the records for the current response class. A DO-WHILE loop reads each RDS record. Upon reading a record, the program fills a corresponding check box (using the response index) with the response text as the **label** property of the check box object, enables the check box, and makes it visible. This loop also assembles a character string representing the variable list used for the questions, appending the loop counter to the variable name prefix obtained from the QDS. This character string is used when the data are written to or read from the DCT. This does require the variable names in

the DCT for a MR question follow a strict naming convention (*variable-name**1**...variable-name**N***), but it is a relatively small constraint for the convenience it provides. The DO-WHILE loop executes until no more records are found in the subset RDS, leaving any unused check boxes empty, inactive and invisible. The final value of the loop counter is stored as the maximum number of responses and the RDS is closed, as its job is complete.

Now the DCT gets into the act. It opens with access to the subject's record (based on ID received as a parameter), the variables defined by the response string created during the RDS loop and the current question variable. If the values for the variables corresponding to this question are not missing (i.e., the question has already been answered), the frame has to fill the respective check boxes before accepting any respondent input. The data control table remains open after this task is finished, because we will need it to write the responses before leaving the frame.

The MAIN section has to test for the exclusive answer. If there is an exclusive answer to the current question, each time a box is checked may affect the contents of the other boxes. In other words, if "none of the above" is selected, no other boxes should be checked and vice versa.

Writing the data to the DCT is done by using CALL PUTVARN() in a DO loop that runs from one to the number of maximum responses. This is currently DHQ-specific, writing zero or one, but could easily be made more general by putting the values from the RDS into an array as the RDS is processed at the beginning of the frame. The code that writes the data to the DCT is called from both control push buttons. The forward button returns one to the calling frame, while the backward button returns zero. Both terminate the MR frame, and the return value allows the calling frame to know which direction to go to the next question. The multiple-choice frames do not allow for skip logic.

TEXT-RESPONSE QUESTIONS

Text responses are questions that require the user to type in text and thus have no records in the RDS, but the QDS still dictates the frame's behavior. The TEXTRESP frame receives question number and subject ID as parameters from the main frame.

The program opens the QDS and reads the record for the current question. The variable TYPE in the QDS defines whether the data item to be entered is character or numeric, and the maximum number of characters is obtained from the response category variable. Text responses are indicated in the QDS by the letter "T", followed by a number that represents the maximum number of characters that can be entered for this response. The question text is displayed, and the control buttons are made visible if warranted (no "Next" button for the final question, no "Previous" button for the first question.)

There are two identical, overlaid text box objects in the frame. The active object is determined by whether the variable to be saved from this frame is character or numeric. The DCT opens, and any format or informat for the variable is obtained from there along with the value (if there is one). The contents of the active text box are written to the DCT with CALL PUTVARN() when the user clicks on either the forward or backward button. As in the MR frame, this frame closes when the user clicks on one of the control buttons. It also returns a direction to the calling frame, and does not allow for skip logic.

GETTING OUT

This is the only function that is hard-coded throughout the entire application. The last question number in the survey must be "fini". This ends the application, closes any open datasets, deletes any SCL lists and invokes the scoring program. These actions also occur whenever the user presses the "stop" button.

HOW LONG DID THIS TAKE TO DEVELOP?

Development time was, at most, equivalent to the amount of time required to do it in a traditional SAS/FSP manner, even recognizing the method and prototype were built from scratch. This is true despite the wealth of existing SAS/FSP prototypes potentially available to build the survey instrument. However, these prototypes were built for specific forms. They could only be recycled, not reused, unless the entire form itself was being reused. Considering the length of the DHQ, it was likely the traditional SAS/FSP method would have taken much longer. At a minimum, it would have mandated an enormous increase in the amount of typing. The metadata tables were created by using base SAS to read ASCII files created by cut-and-pasting question and response text from the original DHQ Word document.

DEPLOYMENT OF THE DHQ APPLICATION

By the time of deployment, the application's design had already changed from its original state, which employed a separate, custom frame for each MR question. The single-column MR frame was created by modifying the largest MR frame in less than two minutes! The changes necessary in the control frame took less than ten.

The version of this application as used at the Washington University School of Medicine's Division of Biostatistics held promise for reuse in other survey-type data entry situations. It was accessed from remote field centers with computers that had X-Windows software installed, and connected to a LINUX system located at the Division of Biostatistics through a virtual private network (VPN). The LINUX system had the SAS System installed, and the DHQ was administered on-line, without paper forms, in real time as part of a larger SAS-based data management application. Since the application writes the data as each question is answered, the only data ever lost due to connectivity issues was limited to data from the question just answered. When the user restarted the application for that person, it would pick up where it left off, so there was no need for the end user

to scroll through the questionnaire searching for the last question answered. There were no response time or user issues, so we were confident enough to use the same concept of remote data management with X-Windows at the remote end, and a VPN connection to a local server for our next large-scale project.

A generalized version of the DHQ application is available free of charge by sending an e-mail to the author. This version requires Base SAS for Windows, and is designed for the express purpose of administering the DHQ as either a computer-assisted interview or an interviewer-administered computerized questionnaire. This is currently the only freely available computerized method for capturing data from the DHQ. Instructions on how to modify the question and response datasets to customize the DHQ for the user's specific needs are provided. SAS/AF is only necessary if the user wants to create any custom frames.

TWO QUICK VARIATIONS ON A DATA-DRIVEN THEME

The method was modified to fit two other uses without affecting its data-driven nature. The first added the capacity to randomize the ordering of answer choices for any given question to reduce any ordering effect on respondent answers, and was effected by adding a variable to the QDS indicating whether the response order should be randomized. The following table shows the difference in the process of getting the responses between the DHQ application and its survey variant when the randomization is in effect for a given question.

DHQ	Survey
1. Use SQL with WHERE and ORDER BY clauses to get the correct subset of response choices from RDS based on the index value from the RDS.	1. Use SQL to get correct subset of response choices from RDS using WHERE clause and create a random number for each response choice. Use ORDER BY clause to order according to the random number.
2. Display response choices in current sorted order.	2. Based on sorted order, create a temporary index value for each record (from 1 to n, incremented by 1) while retaining original RDS index value.
	3. Use temporary index value as internal key for question and RDS index value as key for skip logic.
	4. Display response choices in current sorted order.

The second derivation of the method administers a multiple-choice exam with immediate feedback on a question-by-question basis. The exam application also provides for an optional explanation of the correct answer. As in the survey applications, the order of the response choices is randomized for each student to lessen the furtive copying of key presses. While the correct answer to a specific question may be the first choice listed for student A, it is not guaranteed that the same question will have the correct answer associated with the first response choice for student B.

CHANGES AND IMPROVEMENTS TO RELEASE 1.0

The metadata system was originally designed for a specific survey with only three possible types of data entry. In order to expand the concept beyond this narrow scope, we had to consider what would be necessary for the use of this method as a generalized data entry tool. After all, what good does it do to have a standard method where you still have to custom code 50 percent of the pieces? We created a few more standard FRAMES and added them to the package to administer the following types of data entry:

1. Date (using a calendar pop-up)
2. Date as month, day, and year separately
3. Date as day, month, and year separately
4. Data where the units can be specified as either/or (e.g., pounds or kilograms)

Two small improvements were also made at this point. The metadata method was enhanced to capture additional text if "Other (please specify)" was listed as a possible response choice. The second improvement changed the way common text was handled. In version 1.0, the text was hard coded, and the object that displayed the text was simply toggled on or off by the USE_COMMON_TEXT variable. Instead of a simple toggle, version 2.0 of the metadata system utilizes another dataset, one that contains header text. This allows the use of many different headers throughout the system.

The biggest enhancements, however, involved the skip logic. In release 1.0, when a question or questions were skipped, the user entered no responses because the questions were not displayed. The issue arises when someone goes back to a prior question after having followed the standard flow of control and then invokes a skip pattern. Any responses already given to the questions now skipped will remain. That problem was solved as a critical part of the second release. Now, if the user modifies an existing response that invokes a skip pattern, the program will follow the original flow of control from the change point to the

most recently answered question. It uses the answers stored (including any skip patterns embedded between the two questions), and retroactively sets the response for each of those questions to missing. After the program corrects the questions now skipped, it proceeds with the alternate flow of control as defined by the skip logic invoked from the changed response.

The second enhancement to the skip logic handling removed the skip logic control from the QDS and placed it into a metadata data set of its own. This allowed questions to have more than two possible destinations based on the response given. In version1.0, there were only two possible paths for a given survey question--the "normal" flow of control, or a single, alternate flow of control. This caused problems when the questionnaire had a structure such as:

If the answer is:	
Yes	Go to question 9b
No	Go to question 10
Maybe	Go to question 9d

Now the metadata system can handle an infinite number of destinations from SR and numeric text response questions. From the example above, you could specify the default destination (next question) as question 10, and instruct the system to go to question 9b if the answer was "yes," or to question 9d if the answer was "maybe."

All of these enhancements lead us to one question: Now that the metadata system was more robust than ever, how could we get people to use it?

And that brought us to phase two.

THE METADATA TOOLS

To allow for general usage of the metadata system, we had to make it more accessible to non-technical users. Although the SAS data sets used are very simple in structure and can be created via an INPUT statement, the IMPORT procedure from an Excel spreadsheet, a VIEWTABLE window or even Enterprise Guide, that still implied opening SAS and working within it to yield the data sets in the format that the metadata system requires. To remove this problem, we created a series of tools to allow a non-technical user to start with an electronic version of a questionnaire and create a metadata system survey via a point-and-click interface. The tools are:

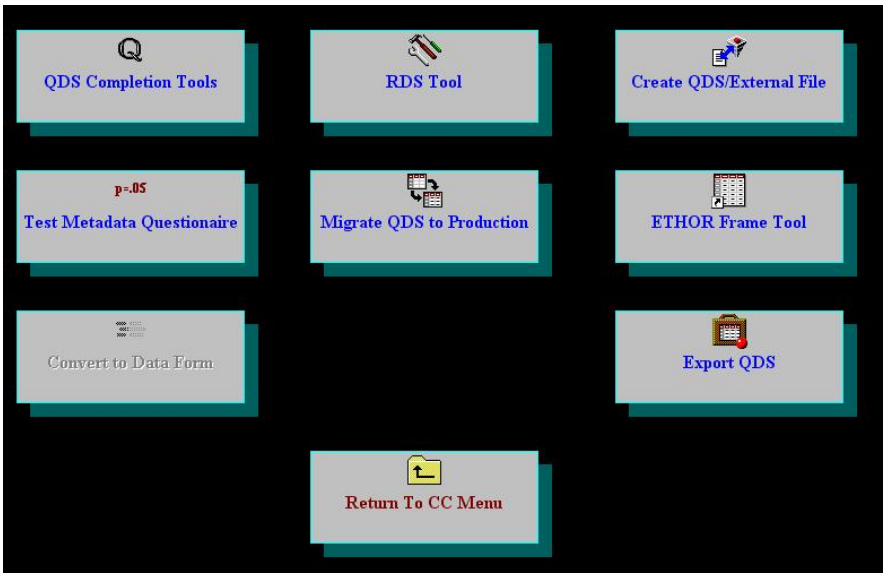
QDS2TXT: Creates the basic QDS from a raw text file, or will write out a finished QDS to a text file

RDS_TOOL: Allows manipulation of the RDS; create/delete/modify response groups from a text file or by direct data entry

QDS_TOOL: "Finishing" tool for QDS; once the QDS is created from the text file, this allows interactive manipulation of the QDS; defines response categories, skip logic, variable names, and create header information; allows insertion/deletion of questions

A second set of tools is used to create menus to access several surveys like those used in medical studies, add categories to the either/or frame and to transfer the QDS to another operating system using Cross-Environment Data Access.

Metadata Tool Menu



The best way to show the user tools and to demonstrate their usage is to follow the process of creating a metadata questionnaire from start to finish. We will start with a sample paper form, and turn it into a metadata survey.

SAMPLE QUESTIONNAIRE

Here is a simple, two-page questionnaire for the Men in Black immigration desk. All aliens and returning earth space travelers must be registered with this questionnaire.

MIB Sol-3 Visitor Questionnaire

Interviewer Script: Welcome to Sol-3 (Earth) Intergalactic Customs. Before you may enter the Sol-3 zone, I need to ask you a few questions. You will be issued a zone visitor's visa at the completion of this brief interview.

A. **Arrival Date:** ☐☐-☐☐☐-☐☐☐☐
day month year

B. **MIB Staff Person Identifier:** ☐☐☐☐☐☐☐☐

1. Are you animal, mineral or vegetable?

- ☐ Animal
- ☐ Vegetable
- ☐ Mineral
- ☐ Other (please specify) _____

2. Are you a resident of the planet Earth?

- ☐ Yes **Go to Question 4.**
- ☐ No

3. What is your home galaxy? _____

4. What is the primary purpose of your visit?

- ☐ Business
- ☐ Pleasure
- ☐ Domination ***Please go to the Dept. of Megalomania in bldg. 5, rm 30300903A.***
- ☐ Complete Destruction ***PRESS PRE-EMPTIVE ANNIHILATION BUTTON IMMEDIATELY!***
- ☐ Re-settlement ***Please go to the department of residency transfer on level 6.***
- ☐ Returning Native

5. How long are you planning to stay? ☐☐☐ **days** ***(if 0, If you are only using the planet as a jump point, you do not have to register.)***

6. What is your date of birth or age (Sol-3 equivalent)?

DOB: ☐☐-☐☐☐-☐☐☐☐ **OR** **Age:** ☐☐☐☐☐
day month year

7. What vital organs are you missing (check all that apply)?

- ☐ Brain
- ☐ Eyes
- ☐ Ears
- ☐ Organs for Verbal Communication
- ☐ Organs for Telepathic Communication
- ☐ Organs for Visual Communication
- ☐ Heart
- ☐ Lungs/Gills
- ☐ Organs to process non-oxygenated environments
- ☐ Reproductive Organs
- ☐ Other (please specify) _____

8. How many limbs do you have? ☐☐☐☐

9. What is your familiar designation? _____

10. In case of emergency, how should we contact you?

- ☐ Telephone
- ☐ Subspace
- ☐ Non-visible light transmission
- ☐ Telepathic network transmission
- ☐ Other (please specify) _____

11. Please provide your contact frequency. _____

12. Please confirm the purpose of your visit.

- ☐ Business
- ☐ Pleasure
- ☐ Domination ***Please go to the Dept. of Megalomania in bldg. 5, rm 30300903A.***
- ☐ Complete Destruction ***PRESS PRE-EMPTIVE ANNIHILATION BUTTON IMMEDIATELY!***
- ☐ Re-settlement ***Please go to the department of residency transfer on level 6.***
- ☐ Returning Native

Thank you. Please enjoy your stay on the third planet of the Sol system.

BUILDING THE METADATA

The first step in building the metadata is to create the template data set for the data being collected, or the DCT. This should contain all the variables in the survey, as well an ID variable(s) to identify the record, and three internal record-keeping variables for the metadata system. Two of these variables track the entry person and the date and time of record modification. The third variable is used by the metadata system to keep track of the last question answered in the survey, allowing a user to return to the same survey at the point where he or she left it.

Code to Create the Template for the SAMPLE data set

```
1   DATA clinic.sample (ENCRYPT=YES READ=&rdpswd WRITE=&wrpswd ALTER=&altpswd)
2   /* For information on how the macro variables are used in line 1, see the second
    item in the "References" section */
3   LENGTH id $ 5 date 4 staffno $ 8 amv 3 amv_othr $ 80 resident 3 home $ 80 purpose
    3 duration_of_stay 8 lifespandate 4 lifespanage 4 vitalorg1-vitalorg11 3
    vitalorg_othr $ 256 limbs 4 name $ 80 contact_via 3 contact_via_othr $ 80 contact
    8 purpose2 3 entryid $ 8 lastfld $ 32 touchdat 6;
4   FORMAT amv amv. resident vitalorg1-vitalorg11 yn. purpose purpose2 bpd.
    contact_via cmeth. version date lifespandate date9. touchdat dateime18.;
5   INFORMAT amv amv. resident vitalorg1-vitalorg11 yn. purpose purpose2 bpd.
    contact_via cmeth. version date lifespandate date. touchdat datetime.;
6   LABEL
7   date = 'Arrival Date'
8   staffno = 'MIB Staff Person'
9   amv = 'Are you animal, mineral or vegetable?'
10  resident = 'Are you a resident of the planet earth?'
11  home = 'What is your home galaxy?'
12  purpose = 'What is the purpose of your visit?'
13  duration_of_stay = 'How long are you planning to stay?'
14  dob = 'What is your date of birth?'
15  vitalorg1 = 'Missing vital organs Brain'
16  vitalorg2 = 'Missing vital organs Eyes'
17  vitalorg3 = 'Missing vital organs Ears'
18  vitalorg4 = 'Missing vital organs Verbal Communication'
19  vitalorg5 = 'Missing vital organs Telepathic Communication'
20  vitalorg6 = 'Missing vital organs Visual Communication'
21  vitalorg7 = 'Missing vital organs Heart'
22  vitalorg8 = 'Missing vital organs Lungs/Gills'
23  vitalorg9 = 'Missing vital organs non-oxygenated environments'
24  vitalorg10 = 'Missing vital organs Reproductive'
25  vitalorg11 = 'Missing vital organs Other'
26  vitalorg_othr = 'Missing vital organ description'
27  limbs = 'How many limbs do you have?'
28  name = 'What is your familiar designation?'
29  contact_via = 'In case of emergency, how should we contact you?'
30  contact = 'Contact frequency'
31  purpose2 = 'Confirm the purpose of your visit.'
32  ;;;;
33  STOP;
34  RUN;
```

Once the template dataset is in place, it is time to create the QDS and RDS for the survey. We start by cutting and pasting the question text and responses into plain text files that will be used by the metadata tools.

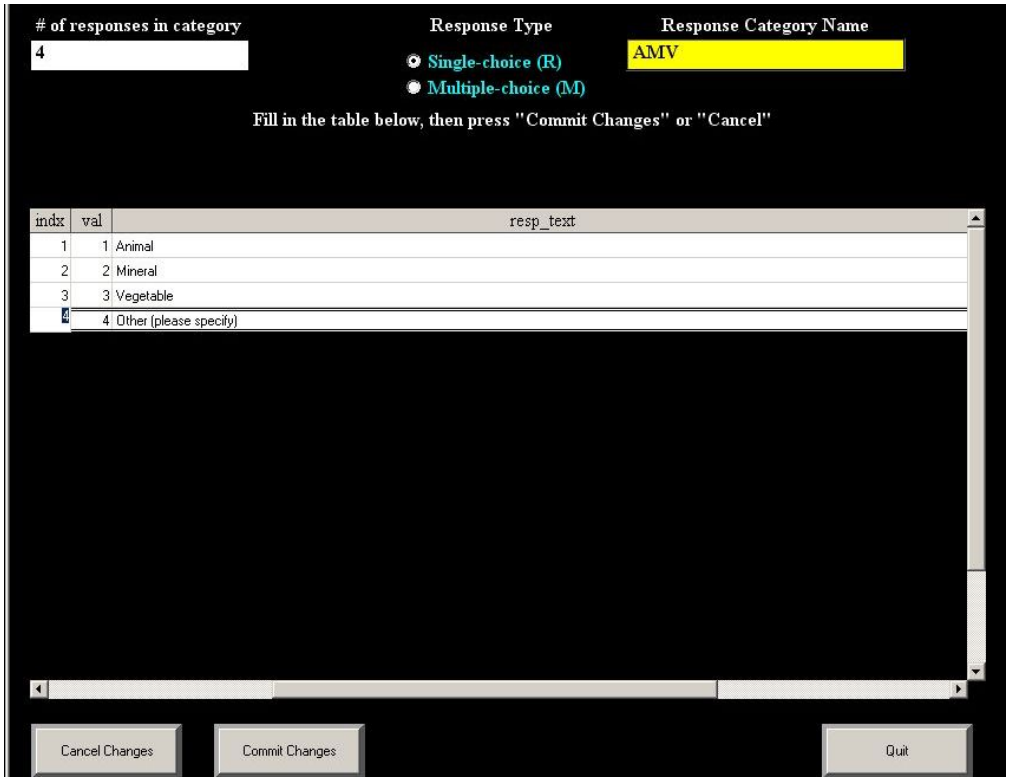
CREATING THE RESPONSE DATA SET

One of the principles of the metadata system is re-use of existing components, and this is especially true for the RDS. In practice, the RDS grows over time, because many surveys will use the same response groups (e.g., Yes/No). Any given set of responses only needs entry into the RDS once, and it will then be available for use by any metadata survey. After the user clicks on the RDS tool icon, the pull-down menu determines which task to perform.

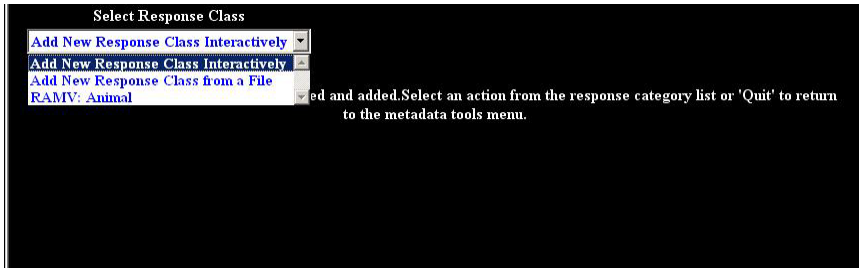
Since this is going to be the first use of the metadata system and, therefore, a brand new RDS, we will need to add the responses for all the questions from the survey. We will do the responses for question one ("Animal"; "Mineral"; "Vegetable"; and "Other (please specify)") interactively. This is a SR question with four possible responses. We have also given the response category a name, "AMV." If we leave that blank, the response category assigned would be a sequential number and

still available for re-use. Simply type in the value that is to be stored in the database with that selection, and the text for each answer where indicated.

Entering a Response Category Interactively



Once we have committed the changes, the tool displays the category selection screen again. Notice that the AMV category we just added is now in the drop down list. Clicking on an existing category will allow you to edit it interactively. Once you have selected the file name from a dialog box and pressed the “Create” button, you will see the response categories you are about to create. You may edit them before you actually add them to the response metadata.

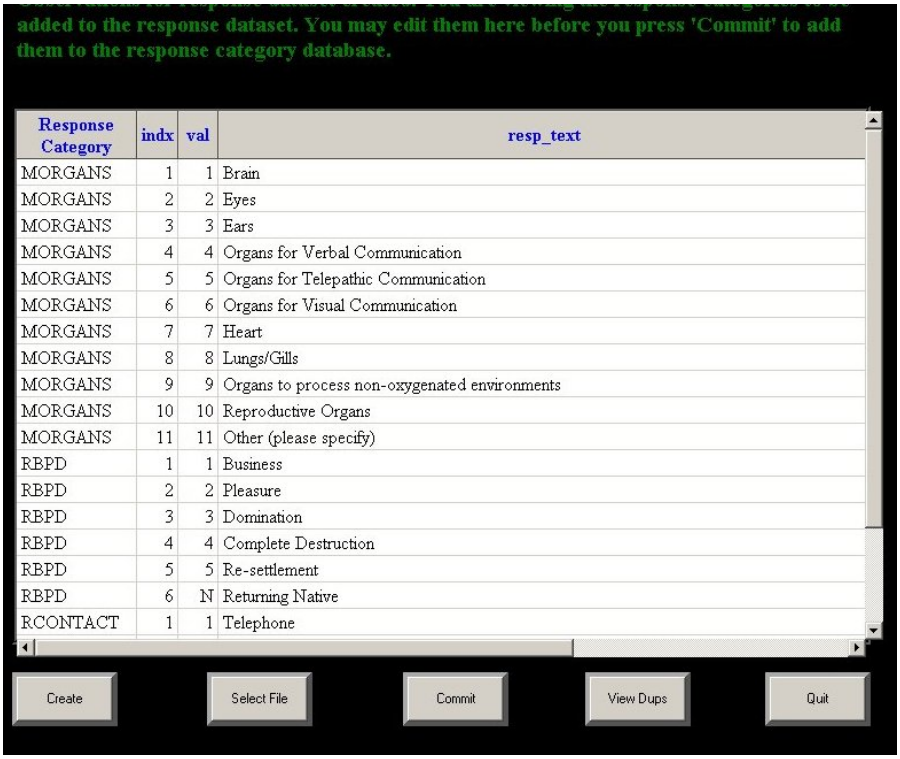


This method is useful for adding one or two response categories, or for making changes to existing ones. However, when you are starting from scratch, creating response categories for an entire survey can be time consuming. Normally, if you have many response categories, you would use the external file method to add them. For this example, we will use this for the remaining response categories. The file that the metadata system uses for this purpose must be formatted in a very specific way.

Metadata Response Input File

```
^RBPD¶
1 → Business¶
2 → Pleasure¶
3 → Domination¶
4 → Complete·Destruction¶
5 → Re-settlement¶
.n → Returning·Native¶
^MORGANS¶
1 → Brain¶
2 → Eyes¶
3 → Ears¶
4 → Organs·for·Verbal·Communication¶
5 → Organs·for·Telepathic·Communication¶
6 → Organs·for·Visual·Communication¶
7 → Heart¶
8 → Lungs/Gills¶
9 → Organs·to·process·non-oxygenated·environments¶
10 → Reproductive·Organs¶
11 → Other·(please·specify)¶
^RCONTACT¶
1 → Telephone¶
2 → Subspace¶
3 → Non-visible·light·transmission¶
4 → Telepathic·network·transmission¶
5 → Other·(please·specify)¶
```

The first line of each response group should begin with a caret character (^), followed immediately by the type of response group (R=single response, M=multiple response), and an optional name. If the name is omitted, a sequential number will be used to name the response group. The response levels must be listed in display order. Each line of the raw file contains the value to be stored in the data set for the given response, and is separated from the text of the response level by a tab character. The tool will process the file, and give you this display:



Once you have verified the response category name, the order (indx), the value to be stored (val) and the text to be displayed for each answer, you add it to the response metadata by pressing “Commit.” Before adding a category, the existing metadata

are checked to determine whether the category is a duplicate. A duplicate category has exactly the same order, the same values, and identical text for each response level. If this is the case, any duplicate categories will not be added. The system will tell you the existing category name for each duplicate so you can use that in the metadata questionnaire. This prevents redundancies in the response metadata.

CONSTRUCTING THE QUESTION DATA SET

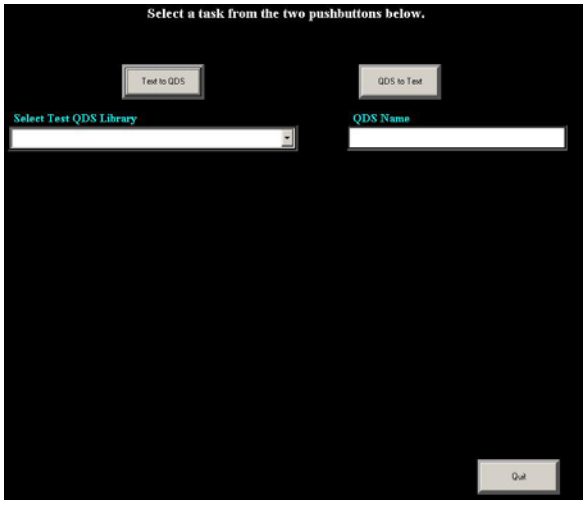
Now that we have the standardized responses ready, it is time to work on the questionnaire itself. This is a two-step process. First, we need to put the questions into a text file; cutting and pasting from your survey document file works well. As with the response data, the system expects the questionnaire text file in a specific format.

The Questionnaire Text File

```
strt→ Welcome·to·Sol-3·(Earth)·Intergalactic·Customs.·Before·you·may·enter·the·
Sol-3·zone,·I·need·to·ask·you·a·few·questions.·You·will·be·issued·a·zone·pass·at·
the·completion·of·this·brief·interview.¶
A  →  Arrival·Date¶
B  →  MIB·Staff·Person¶
1  →  Are·you·animal,·mineral·or·vegetable?¶
2  →  Are·you·a·resident·of·the·planet·earth?¶
3  →  What·is·your·home·galaxy?¶
4  →  What·is·the·primary·purpose·of·your·visit?¶
X1 →  Please·go·to·the·department·of·residency·transfer·on·level·6.¶
5  →  How·long·are·you·planning·to·stay?·(Enter·.N·if·returnee)¶
6  →  What·is·your·date·of·birth·or·age·(Sol-3·equivalent)?¶
7  →  What·vital·organs·are·you·missing·(check·all·that·apply)?¶
8  →  How·many·limbs·do·you·have?¶
9  →  What·is·your·familiar·designation?¶
10 →  In·case·of·emergency,·how·should·we·contact·you?¶
11 →  Please·provide·your·contact·frequency.¶
12 →  Please·confirm·the·purpose·of·your·visit.¶
x2 →  Please·go·to·the·Department·of·Megalomania·in·building·5,·room·30300903A.¶
x3 →  PRESS·PRE-EMPTIVE·ANNIHILATION·BUTTON·IMMEDIATELY!¶
OK →  Thank·you.·Please·enjoy·your·stay·on·the·third·planet·of·the·Sol·system.¶
OKTRANS  →  Thank·you,·and·have·a·nice·day.¶
fini→ Press·"Exit"·to·end·survey.¶
```

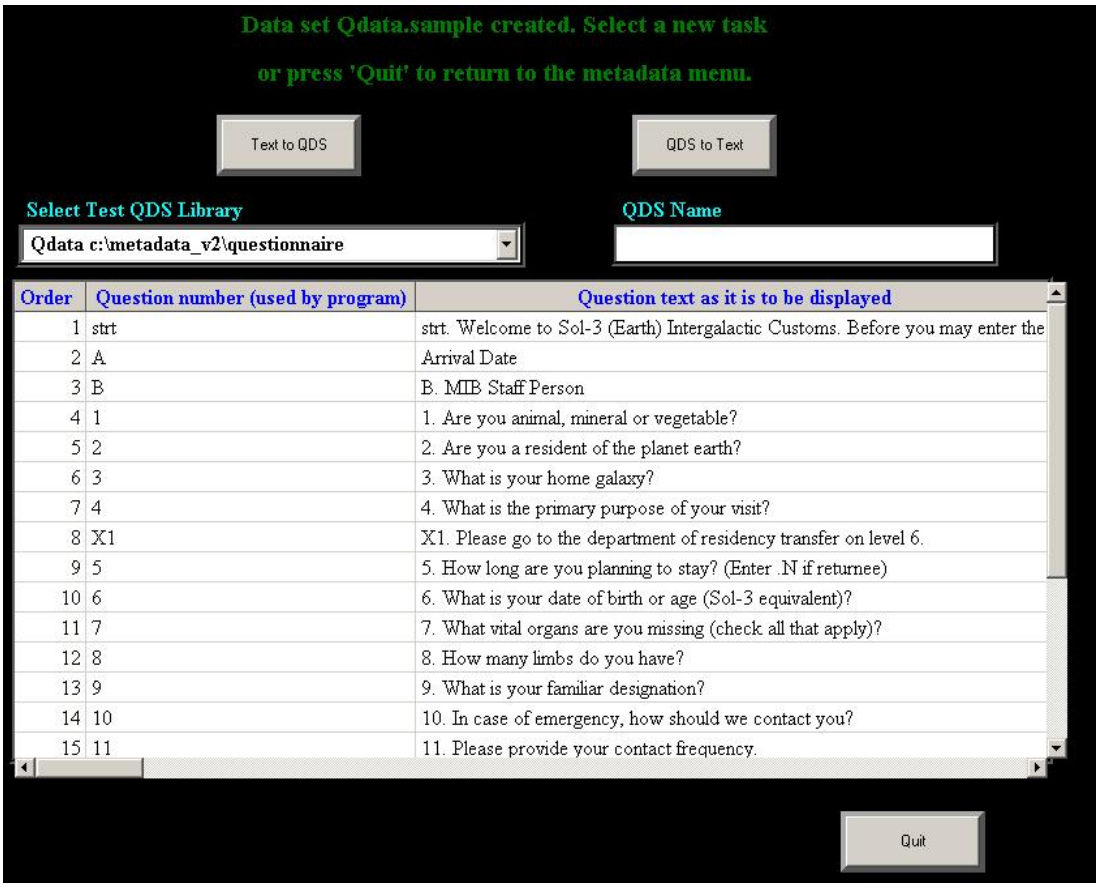
You can see that the file is again tab-delimited. The first field in each record is the internal question number identifier mentioned earlier. The only fixed rules for naming the question records are that the internal numbers are case sensitive, and the final question in the questionnaire must have the name ‘fini’. Any prompts that do not require responses are also included in this file as well (items X1, x2, x3, OK, OKTRANS, and fini). Once the file is ready, the next tool we use will be the “Create QDS/External File” tool. This is a two-way tool: it can create a QDS from an external text file, and it can create an external file from a QDS (at any stage).

Create Basic QDS from Text File



You begin by selecting which task you want: creating a QDS from an external file, or creating an external file from an existing QDS. In this case, we want the first option, and we are prompted for the name of the external file. We select the library name from the pull-down list on the left, and enter a name for the QDS on the right. When these are filled in and the external file has been selected, we press a button and the basic QDS is built and then displayed.

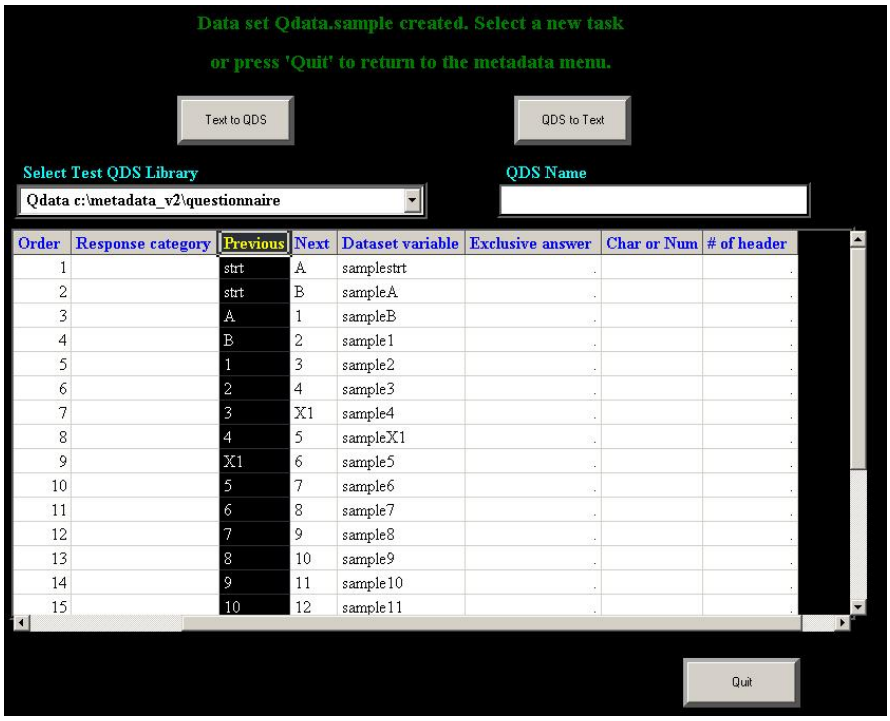
Base QDS from the External File



Order	Question number (used by program)	Question text as it is to be displayed
1	strt	strt. Welcome to Sol-3 (Earth) Intergalactic Customs. Before you may enter the
2	A	Arrival Date
3	B	B. MIB Staff Person
4	1	1. Are you animal, mineral or vegetable?
5	2	2. Are you a resident of the planet earth?
6	3	3. What is your home galaxy?
7	4	4. What is the primary purpose of your visit?
8	X1	X1. Please go to the department of residency transfer on level 6.
9	5	5. How long are you planning to stay? (Enter .N if returnee)
10	6	6. What is your date of birth or age (Sol-3 equivalent)?
11	7	7. What vital organs are you missing (check all that apply)?
12	8	8. How many limbs do you have?
13	9	9. What is your familiar designation?
14	10	10. In case of emergency, how should we contact you?
15	11	11. Please provide your contact frequency.

This is called the “base QDS” because only the question number and question text are truly ready to use. Default variable names are created, and default values of the QDS variables **prev** and **next** are filled in, providing the most basic flow of control for the metadata survey.

Default Settings for the QDS

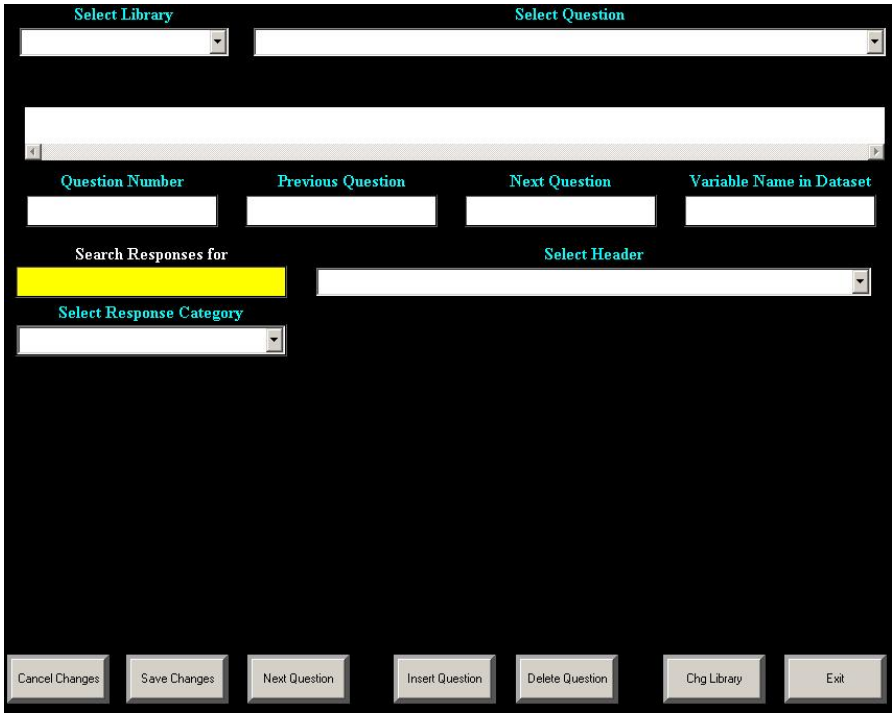


This version of the QDS cannot be used right away. Response categories have not yet been defined for any of the questions, the variable names probably are not correct, and the flow of control does not account for any skip patterns. The second step of the process of creating question metadata is using the QDS Completion Tool to correct all of these issues.

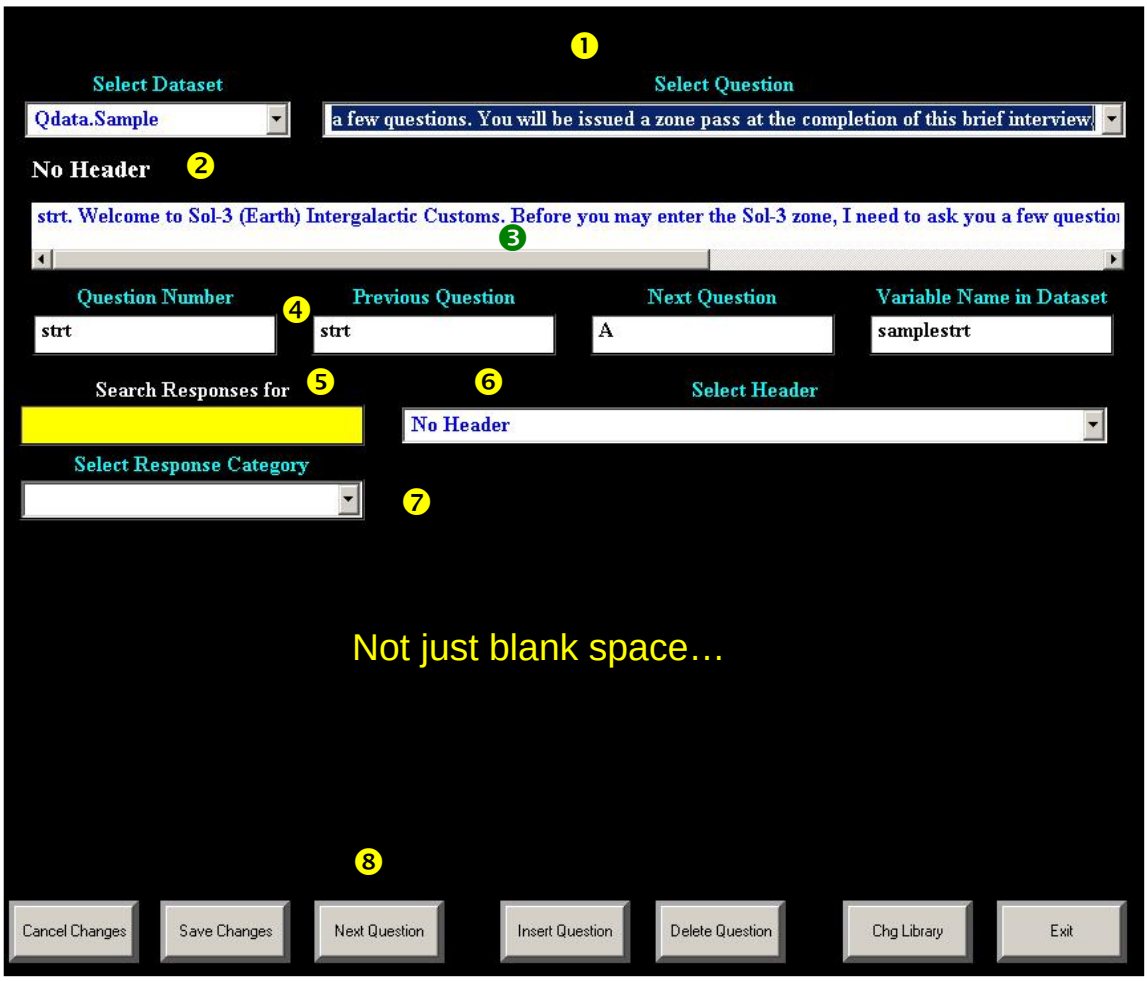
THE QDS COMPLETION TOOL (AKATHE QDS EDITOR)

The tool allows editing of the question metadata interactively, and enables the metadata system to be utilized by non-technical users. The “Select Library” pull-down allows you to select the name of the library where the QDS file is located. After you choose a library, it changes to display a list of all the SAS data sets in that library so you can select the QDS file you want to edit.

Starting the QDS Editor



Editing the QDS



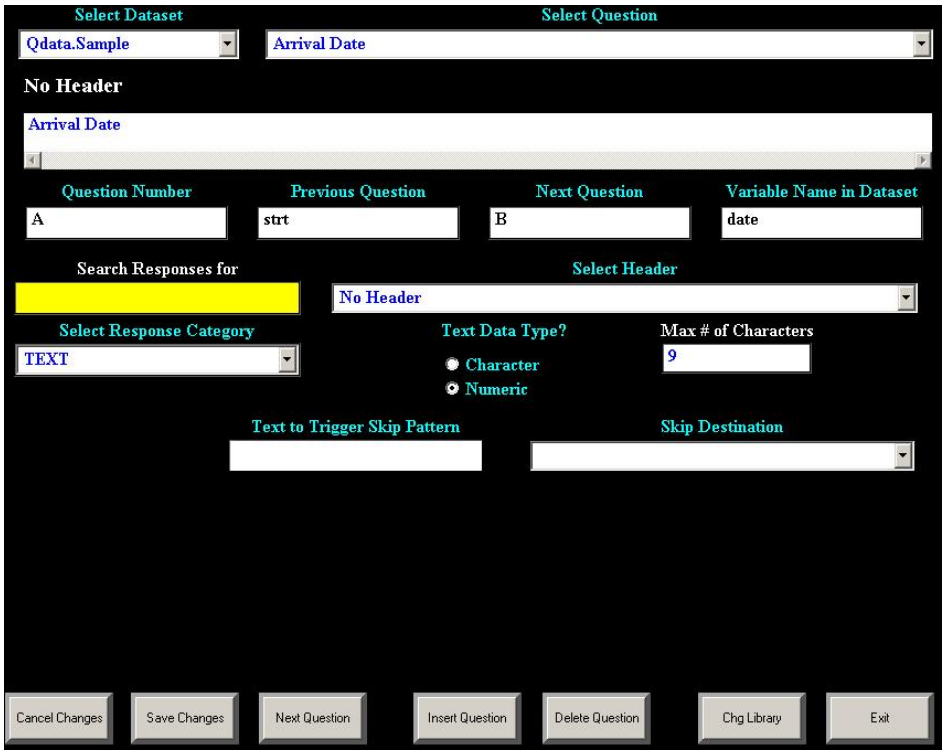
The QDS Editor provides access to all the fields and features of the metadata system from a point-and-click environment. The questions can be selected from the pull-down menu at the top right (❶). All question records for this data set are in this pull-down menu, and selecting one will display all the information in that record. You can also navigate through the questionnaire from the beginning to the end via the "Next Question" button (❽) at the bottom of the tool. Any header associated with the question is displayed in (❷), and is associated with the question by selecting a header from (❻). If you need to create a new header, one of the choices in the header pull-down list (❻) will open a window to allow you to type in the text of this new header, and it will automatically be added to the header metadata data set.

Item ❸ is where you can edit the text of the question itself. This usually is required to add/remove the question numbers added to the question text in the previous step of creating the base QDS from an external file. However, if the text of a question changes, you can make the change here without having to change the external file, re-create the QDS and edit the entire metadata survey from scratch. Basic flow-of-control fields are accessed in ❹. The internal question number, previous and next questions and the variable name in the DCT associated with the question are all here. A default variable name was created in the previous step by concatenating the name of the QDS file (in this case, "sample") with the internal question number. You can change that, or you can indicate the text is for display only and does not require a response by removing any value in this field. For example, the first record in the QDS is a standardized script to be read. If we remove "samplestrt" from the "Variable Name in Dataset" field, the metadata system will display the text without asking for any response.

A pull-down list (❼) gives you all the response categories available from the response metadata, as well as those special response categories built into the metadata system. To help you find the category you are looking for, a search tool is available (❺). This will search the response metadata for the text in the search box, and will display all categories with that text. Where does it do this? In the area marked, "Not just blank space…"

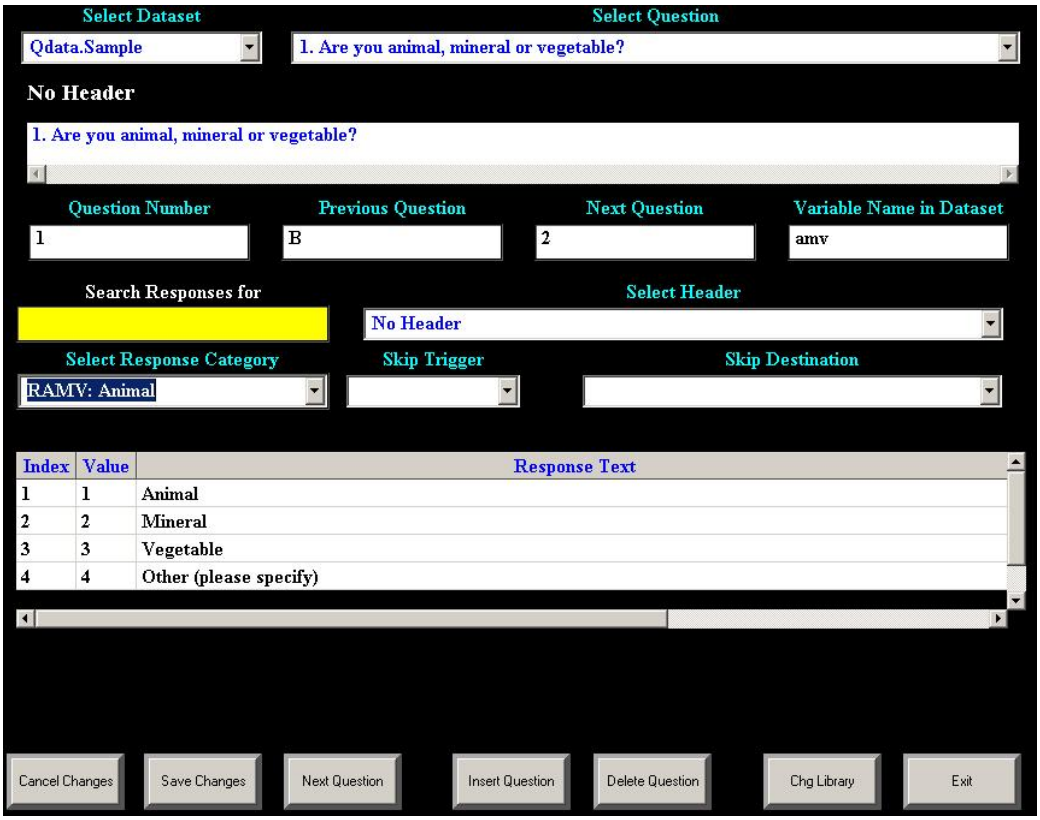
Now we will show how the QDS Editor handles some of the various types of questions. Starting at the top of the survey and working our way down, we will do the "Arrival Date" question first, which is a text response.

Text Response Question



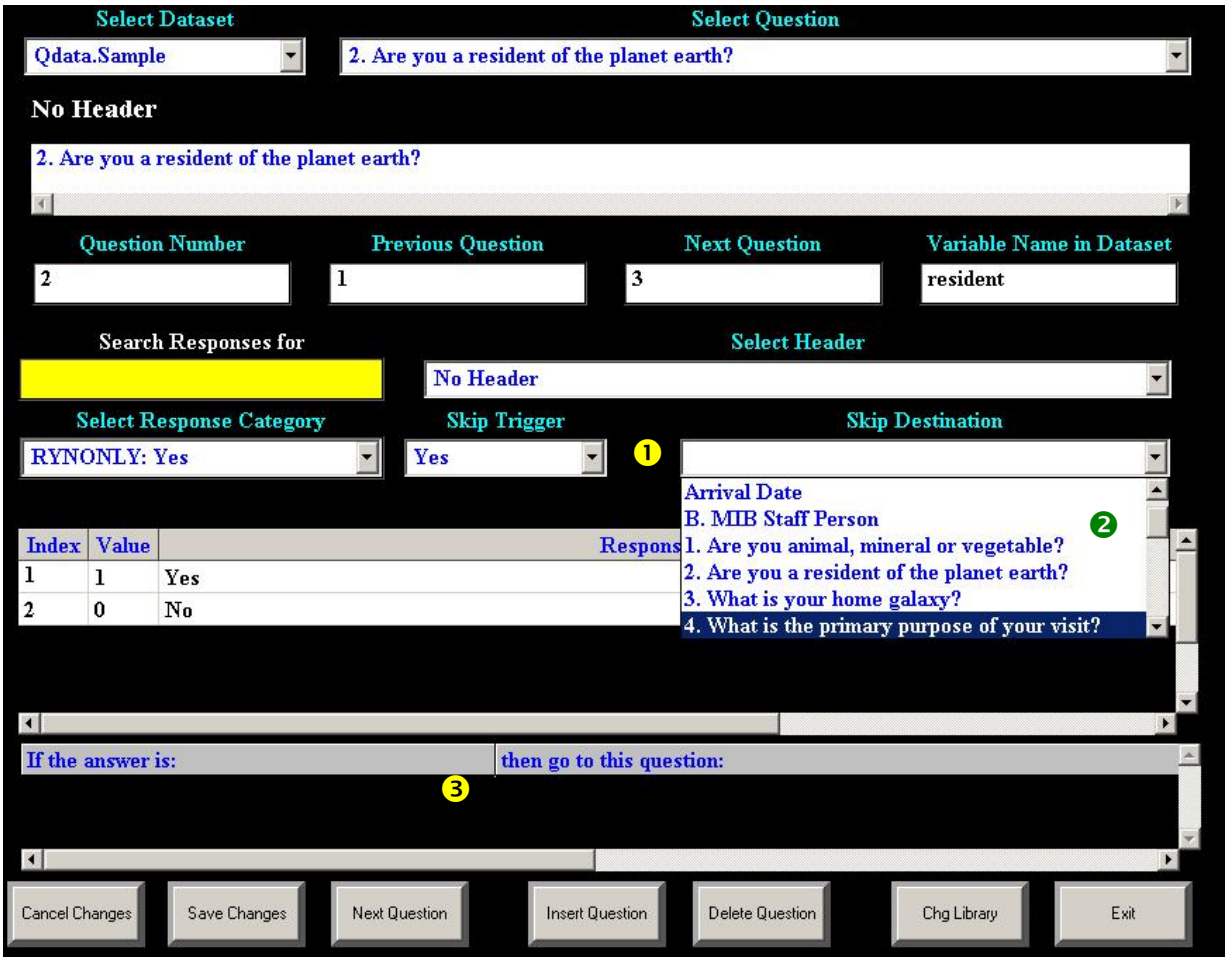
Now that the QDS Editor knows what type of response is expected, it will ask the necessary questions to complete the metadata for this record. When you select “TEXT”, the metadata system needs to know whether the text is going to be stored as character or numeric, and the maximum number of characters. Any formats or informats are obtained from the DCT at run-time. Because it is numeric text, it also asks for text that will trigger a skip pattern; leave these blank if a skip pattern is unneeded. You can click on “Save Changes” or “Next Question” to commit the changes to the QDS. Next, we will define a SR question using one of the response categories we created.

Single-Response Question without Skip Pattern



As you can see, we have changed the variable name and selected the RAMV category from the pull-down menu. The QDS Editor responds by displaying all levels for the category and opening boxes for skip logic. Since this question has no skip logic, we leave those blank and move to the next question.

Defining Skip Logic

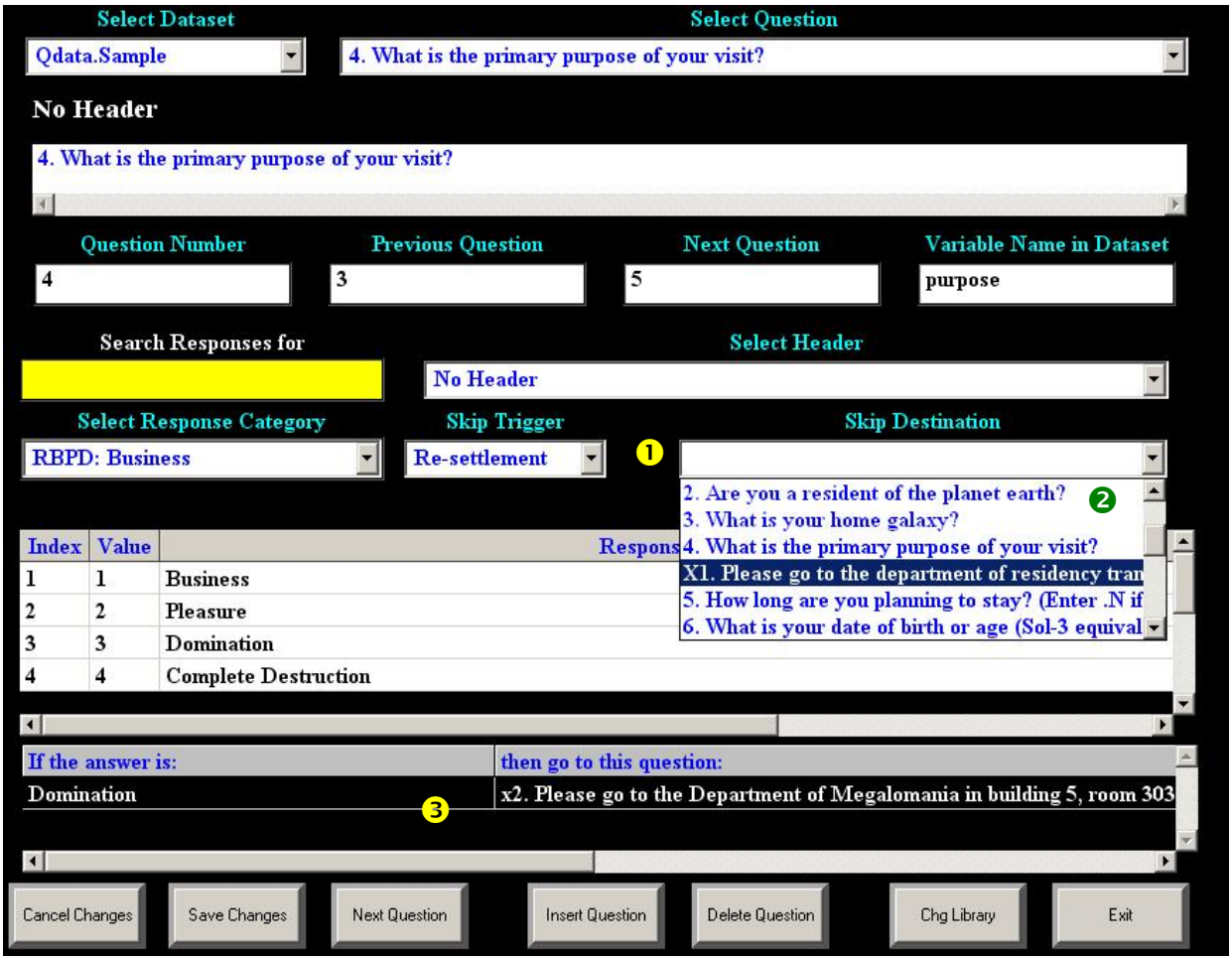


This question involves skip logic. The QDS Editor allows you to select the trigger based on the response text (❶), and to define the skip destination based on the question text (❷). The result(s) of your selection(s) are then displayed in section ❸, which is also interactive. Double clicking on a record in ❸ will erase it from the skip logic metadata, allowing you to redefine or correct the skip logic for this question. There is no limit to the number of skip triggers or destinations for a given question. You may even define the same destination for multiple triggers if the survey calls for that. We will now look at question 4 from the questionnaire, which has three destinations based on skip logic, as well as a default flow of control.

4. What is the primary purpose of your visit?

- ☐ Business
- ☐ Pleasure
- ☐ Domination ***Please go to the Dept. of Megalomania in bldg. 5, rm 30300903A.***
- ☐ Complete Destruction ***PRESS PRE-EMPTIVE ANNIHILATION BUTTON IMMEDIATELY!***
- ☐ Re-settlement ***Please go to the department of residency transfer on level 6.***
- ☐ Returning Native

Defining Multiple Skip Destinations Based on Different Answers to the Same Question



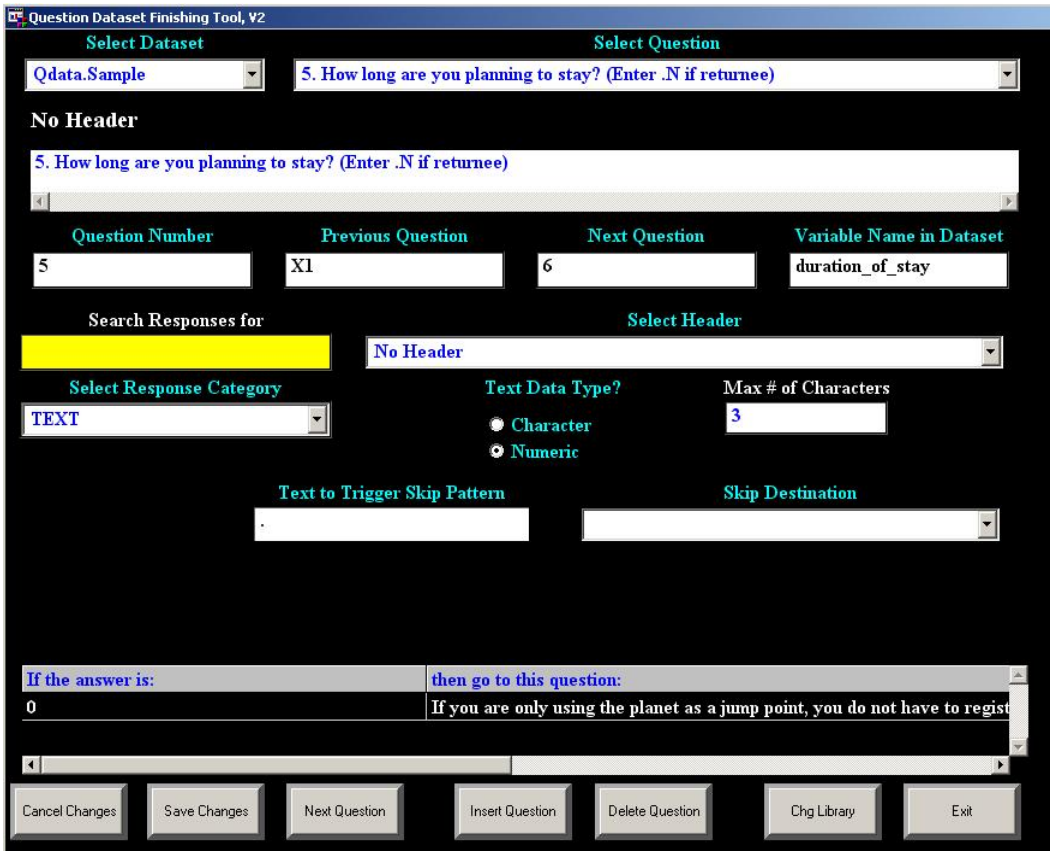
Everything looks similar to the previous example, except that ❸ now has something in it. If the answer is "Domination," then the metadata system should go to the question with the text "Please go to the Department…" The tool communicates with users via the text of questions and answers, so it does not matter what the internal question number stored in the metadata is. If you change the internal question number, your skip logic will be incorrect at the least or, at the worst, it will cause the metadata system to fail. In the above example, we have already added the skip pattern for the "Domination" answer, and are about to add one for the "Re-settlement" answer, which will show up in ❸ when we select the skip destination. If you make a mistake, it is easy to remove existing skip logic by double clicking on the record you want to remove in ❸.

Another feature is the ability to add or remove questions from the metadata. Even though you were careful when setting up the question file, it appears you forgot something. Question six has a skip pattern and the text of that skip pattern is not in the question file, unlike all the other skip patterns. You can just add the new question record from the tool.

Adding a Question

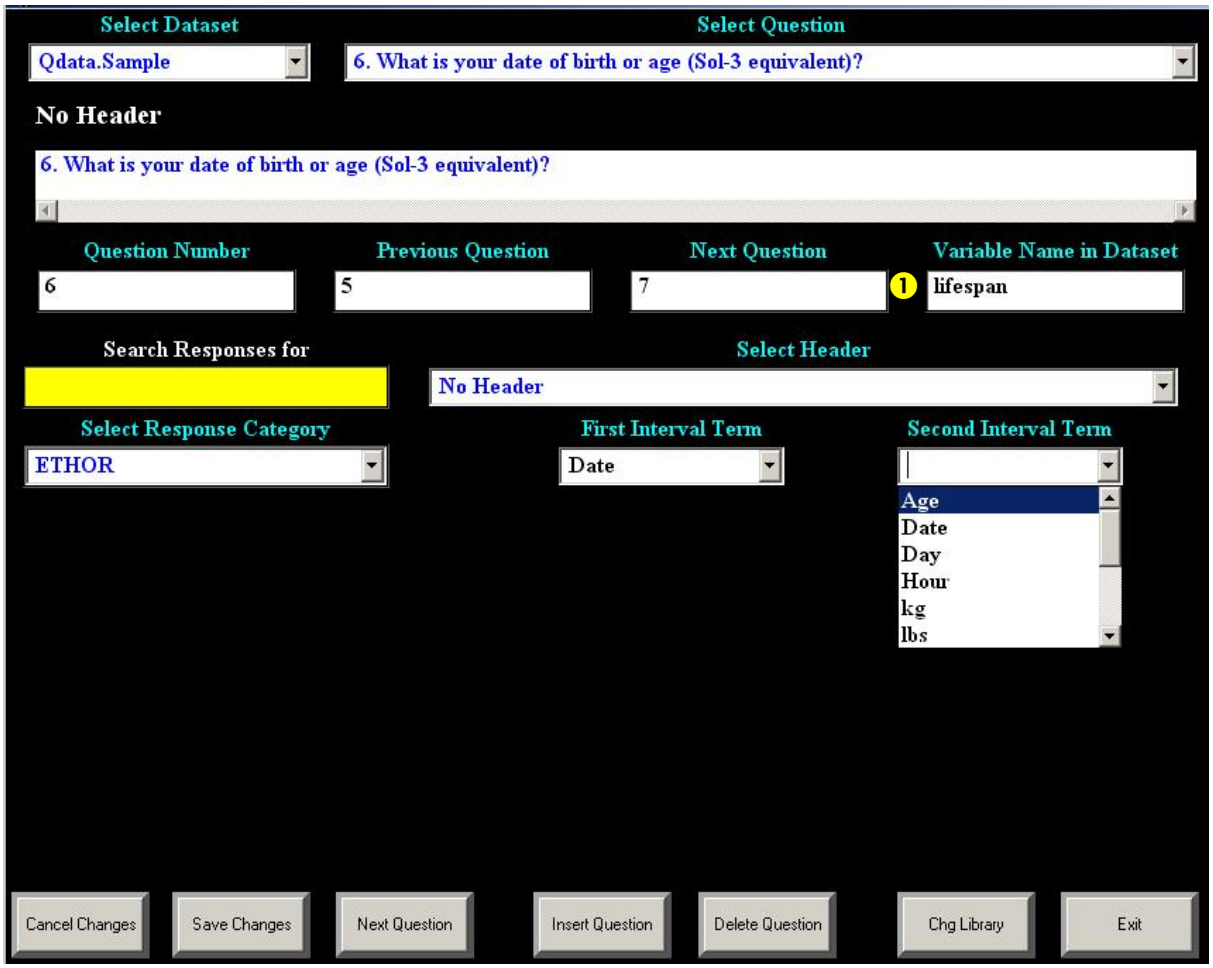
You can add a question anywhere in the file. The metadata system does not care where you put the actual record because the order of the records in the QDS does not determine flow of control. However, in practice most users prefer inserting questions where they belong with respect to the paper form. Now we can go back to question five and insert the skip logic.

Text Skip Pattern



The next question in the survey gives two possible units for the response. Here is how the metadata handles it.

Either / Or Questions

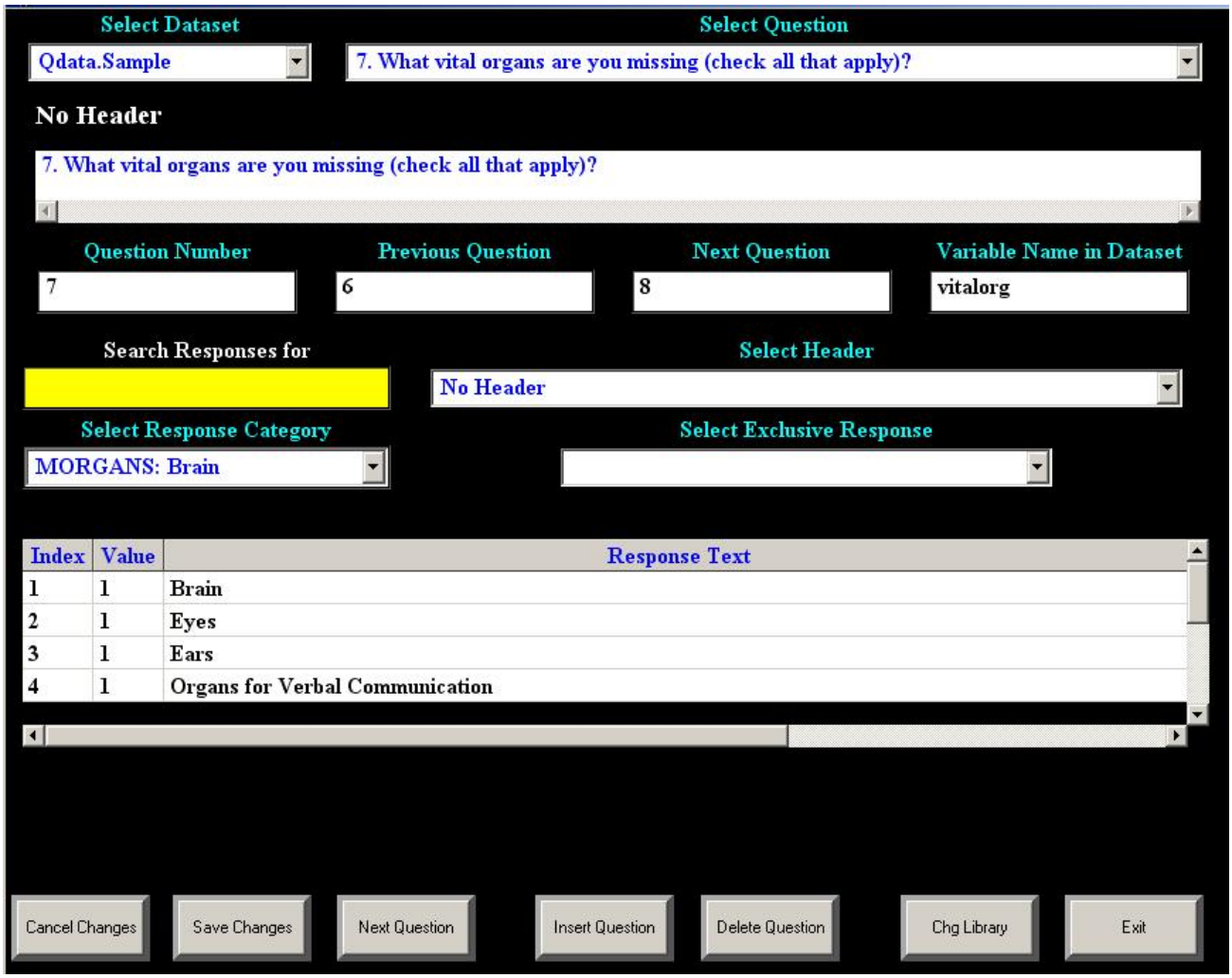


There is a special pre-defined response category called “ETHOR”. The variable name you provide the metadata system is used as a root for the actual dataset variables (“lifespan” above (❶). You select the terms from the two pull-down boxes. They are filled with metadata from a list of terms, and there is a separate tool to add terms not already in the term metadata table. Since we are dealing with date and age in question six, the variable names in the dataset are a concatenation of the root variable name you specified, and the term abbreviation. Therefore, the variable names are “lifespandate” and “lifespanage,” only one of which will not be missing for any given record. The term abbreviations are fixed and can be obtained from the tool that modifies the term metadata.

```
DATA clinic.sample (ENCRYPT=yes READ=&rdpswd WRITE=&wrpswd ALTER=&altpswd)
LENGTH id $ 5 version 4 fc $ 2 visit $ 2 date 4 staffno $ 8 amv 3 amv_othr $ 80
       resident 3 home $ 80 purpose 3 duration_of_stay 8 lifespandate 4 lifespanage 4
```

The last type of question we will cover here is the multiple-choice question, where you can select any one of a number of answers.

Multiple-Choice Response Question



Here the variable naming works the same way as with the either/or questions; the variable name you supply in the QDS Editor is the root. Multiple-choice questions are generally scored as yes or no, one variable per possible response, and they are numbered sequentially. Therefore, the response for “Brain” would be stored in the variable vitalorg1; the response for “Eyes” would be stored in vitalorg2; and so on. You also have the ability to define an exclusive response, which is the response that will reset all other choices to be unchecked. If question seven had a response of “None”, that would be an exclusive response since it is contradictory to the question.

THE METADATA METHOD: PAST, PRESENT, AND FUTURE

The National Cancer Institute’s Diet History Questionnaire is a long-survey instrument that does not translate well into an application using SAS/FSP. The flexible nature of the instrument as well as its length make coding it as an FSP application impractical, expensive to develop in terms of time spent, and difficult to maintain. A SAS/AF method and prototype application have been developed that uses metadata in the form of QDS and RDS. These have significantly reduced development time while providing the capacity for true application re-use in the future. The prototype application can be used as a computer-assisted interview, or by trained interviewers in either a data entry or telephone/live interview environment, with a direct navigation feature that can be turned on or off according to the situation. By changing the metadata, the same application can be used to administer different questionnaires.

The metadata system has evolved from a method to perform a single, specific data entry task into a generalized tool for rapid development of data entry forms from paper templates. This logical step in its life cycle required the construction of tools so that non-SAS programmers could use this to create computerized data entry forms, freeing programming staff for other projects. The metadata system and its tools are not perfect, and do not cover every possible data entry situation. While the system is not developed to the point where naïve users can simply go in and do this, it does make it easy for individuals with limited SAS experience to create fairly sophisticated data entry forms.

The metadata system was still in its relative infancy when asked to deploy a multiple-form, multiple-visit data entry protocol rapidly. It did so admirably, allowing six people to split up the development of over seventy forms and have them production

ready in less than three weeks while also accommodating existing staff workloads. The remote data entry worked well from field centers from as far away as London, England! While end-user acceptance was good, administrative acceptance failed because this method only asks one question at a time, and does not follow the traditional page-oriented mode of data entry. Future plans included the creation of a survey question data warehouse, and the development of additional tools to manage and utilize it so the manual step of creating a QDS from a text file would no longer be necessary. Instead it would be replaced by a process of dragging and dropping question records from the warehouse into the QDS being created, adding the occasional new question to the warehouse. The QDS Editor would still be necessary because skip patterns might change or answers might use slightly different categories, but the new data warehouse would have allowed for the standardization of some questionnaire information. The presence of relevant defaults for variable name and response category would have further reduced the time to deployment of questionnaires because of the increased ease of QDS development.

Although I am no longer actively working on the development of this method, I believe the next major step in the development of the metadata system and concept would be a migration from the thick-client technology of SAS/AF to a thin-client presence on the web. By using Java to connect to the SAS data structures already in place, and given the concepts already developed, tested and successfully deployed to a production environment in a thick-client model, the metadata method continues to hold promise as a viable survey development and data capture tool.

REFERENCES:

Morgan D, Province M. “Building A Better Data Entry Application Using PROC FSEDIT,” *Proceedings of the Twenty-Fourth Annual SAS® Users Group International Conference*, SAS Institute, 1999, 365-374

Miller JP, Schauer, J, Baty JD, Littlewood S, Trinkaus K, Lill RM, Richards R. “Managing Clinical Trials with a SAS-Based Web Portal,” *Proceedings of the Twenty-Eighth Annual SAS® Users Group International Conference.* April 2003 http://www2.sas.com/proceedings/sugi28/181-28.pdf

ACKNOWLEDGEMENTS

This work was funded in part by NIH grant UY1 HL72524-01.

CONTACT INFORMATION:

Derek Morgan
Covidien
675 McDonnell Blvd.
Biostatistics, 30-1
Hazelwood, MO 63042

E-mail: derek.morgan@covidien.com

SAS, and The SAS System are registered trademarks of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration. Any other brand and product names are registered trademarks or trademarks of their respective companies.

SIMPLIFIED FLOWCHART FOR MAIN IN DHQ.FRAME