Session 4: CAPI Data Entry

At the end of this lesson participants will be able to:

- Add question text using multiple colors and fonts
- Add fills in question text
- Use rosters with occurrence labels
- Understand the different capture types
- Create multiple response questions using checkboxes
- Create dynamic value sets to set the list of responses at runtime

Question Text

We can add text to each question in our survey by clicking on *CAPI Questions* in the toolbar.

Let's start with the first few questions in section B, Adults.

For B2 enter "Are you male or female?", and for B3 enter "What is your relationship to the head of household?". Run the application on Windows and then on Android to see how the question text is displayed.

In addition to the text we can add instructions to the interviewer. For example for question for B1 we can add:

Enter the first and last name of the next person in the household 5 years or older.

To make it clear to the interviewer that this is an instruction we can use a different font and color to distinguish it from the question. Let's make the questions blue and the interviewer instructions green italics. It doesn't matter what the color scheme is, as long as it easy to distinguish literal questions from interviewer instructions and it is consistent from question to question.

If you have your question text in Word or Excel you can copy and paste into CSPro and it will preserve the formatting.

Fills in Question Text

It is possible to have the question include the values of dictionary variables. For example for B2 instead of asking "Are you male or female?" we can include the respondents name by using %NAME% inside the question text. At runtime this will be replaced by the contents of the variable NAME. Change the text for B2 and B3 to "Is %NAME% male or female?" and "What is %NAME%'s relationship with the head of household?".

You can also use logic variables as fills in question text. This allows you use to calculated values. For example let's display the current line number in the roster as part of the question text for MORE_PEOPLE:

> *You have entered %currentLineNumber% members 5 years and over. Are there more?*

Now we need to declare the logic variable currentLineNumber. We will declare it in the PROC global section so that it is available everywhere. Up to now we have always declared our logic variables inside the PROC of a dictionary variable or group in which case it is only available inside that PROC. Anything in the PROC global is available in all the PROCs and in the question text.

To view the proc GLOBAL, in the logic view, click on the first item in the form tree. This shows all of the program logic at once: the proc GLOBAL plus all the other procs. Clicking on any other item in the form tree shows just the procs for that item.

We need to assign a value to currentLineNumber. Which proc do we that in? We do that in the preproc of MORE_PEOPLE since we need to use it when we are in that field. We can use the function curocc() (current occurrence) which gives the current row number of the roster:

```
PROC MORE_PEOPLE

preproc
currentLineNumber = curocc();
```

There is one problem with what we have done. If we go into the second row of the roster and then back into the first row, the line number in the question text doesn't change. Why not? The problem is that the preproc is not called when you go back into a field. It is only called when you are going forward. Instead of the preproc we can use another proc called onfocus which is triggered every time you enter the field whether you are going backwards or forwards.

```
PROC MORE_PEOPLE

onfocus
currentLineNumber = curocc();
```

Occurrence Labels

Let's add section F, household assets. Question F1 is a grid so it would make sense to use a repeating record. However we would like to display the grid with asset names in CSPro the same way they are in on the paper questionnaire. We can do this using occurrence labels.

First, create the new record. Let's call it ASSETS. Is it required? No, the household could have no assets. What is the MAX? Twelve since there are twelve rows. Add the asset code, quantity, value per unit, total value and owner fields. Add a "Don't know" code to for value per unit so that we will be able to implement the skip pattern.

In the dictionary tree, select the record and select "Occurrence labels" from the Edit menu. You can copy and paste the labels from the questionnaire into the dialog that comes up.

Now create a form and drag the assets record onto it. Make sure to check "Use occurrence labels" in the drag options. You should see the occurrence labels in the roster.

Note that now that there is a new form we need to change the *endlevel* in the household characteristics form to a skip to the assets form, otherwise we will end up skipping over the assets form.

```
PROC DISTANCE_TO_WATER_UNITS
// Skip distance in km to water if respondent chose to
// give distance in minutes.
if DISTANCE_TO_WATER_MINUTES_OR_KM = 1 then
    skip to ASSETS_FORM;
endif;
```

We can also add the occurrence labels as fills in the question text by including *%getocclabel%*. Add the question texts for the assets questions:

> How much/many %getocclabel%s does the household own that are useable/repairable?
>
> What is the value per unit for your %getocclabel%s? How much would pay if you had to buy it in its current state?
>
> What is the total value of your %getocclabel%s? How much would pay if you had to buy it in its current state?
>
> Who owns the %getocclabel%s?

While we are working on this section we should add in the skip patterns:

```
PROC QUANTITY
// Skip quantity, value and ownership if quantity is zero
if QUANTITY = 0 then
    skip to next;
endif;

PROC VALUE_PER_UNIT
// Skip total value if value per unit is known
if VALUE_PER_UNIT <> 999999 then
    skip to OWNER;
endif;
```

And we should make the asset code protected and fill it in automatically using *curocc()*.

```
PROC ASSET_CODE

preproc
// Fill in asset code automatically based on current line number
ASSET_CODE = curocc();
```

Capture Types

We have already seen a few of the capture types: text box (text/numeric input), radio buttons (choose one) and combo boxes (ranges). There are three others that we have not seen: date picker, checkbox and number pad. You can set the capture type by right clicking on a field and choosing field properties. If you have extended controls enabled when you drag an item onto the form the capture type will be inferred from the value set (if there is one). No value set will result in a text box, a value set with ranges will result in a combo box, a value set with no ranges will result in radio buttons and a value set on an alpha field where the number of values is equal to the length to the length of the field will result in checkboxes.

Checkboxes

Let's start implementing section G, crops. While this looks we would implement it using a roster with occurrence labels like we did for the assets we could instead use checkboxes to make a simpler interface for the interviewer. Create a new singly occurring required record called CROPS. Instead of using a roster we will use a checkbox field for G1. To use a checkbox we must create an alphanumeric item long enough to hold the maximum number of items that can be checked. For G1 there are 7 options that can all be checked at once so we will create a length 7 variable. The value set will contain one entry for each of the options. Add the following value set:

Maize	A
Rice	B
Sorghum	C
Cassava	D
Sweetpotato	E
Beans	F
Groundnuts	G

Create a new form for section G and drop the variable onto it. Right click on the field and verify that the capture type is checkbox. Run it and verify that the checkboxes are shown.

Interpreting checkboxes

Let's add question G4, which asks for the types of sweet potato produced (if any). We only want to ask this question if sweet potato was checked in question G1. The result of question G1 is an alphanumeric so how can we tell if sweet potato was checked? If sweet potato was checked then the letter "E" appears in the answer to G1. We can check for using the *pos()* function in logic. This function returns the position of a character in a string or zero if the character does not appear in the string.

```
PROC CROP_PROCEEDS_SPENT

// Skip type of sweet potato if sweet potato not checked in crops produced
if pos("E", CROPS_PRODUCED_CHECKBOX) = 0 then
    skip to CROP_PROCEEDS_SPENT;
endif;
```

Limiting the number of boxes checked

Now let's add question G5 which asks to choose the three most important ways money from crops was spent from a list of 9 options. We can also use checkboxes for this by limiting the maximum number of choices to 3. This is done by simply setting the length of the alpha field used for the checkboxes to 3.

Converting checkboxes to yes/no

The checkbox makes a nice interface but the resulting data is a string that is hard to interpret and deal with in other software. To make it easier to use for export we can convert from the checkbox to a repeating item with yes/no options. Create a new item in the crops record named CROPS_PRODUCED_REPEATING with length 1 and 7 occurrences. Add a value set with Yes – 1 and No – 0. Add occurrence labels. Drag this field onto the crops form as a roster and make it protected. We will display it on the form purely for debugging purposes.

For each item in the checkbox field that is checked we want to set the corresponding item in the repeating field to one and for each entry in the checkbox field that is not checked we set the corresponding item to zero. To find out if an item is checked we use the function *pos()* which finds the position of one string in another and returns zero if the string is not found:

```
PROC CROPS_PRODUCED_CHECKBOX

// Fill in crops repeating item based on crops checkboxes
if pos("A", CROPS_PRODUCED_CHECKBOX) > 0 then
    CROPS_PRODUCED_REPEATING(1) = 1;
else
    CROPS_PRODUCED_REPEATING(1) = 0;
endif;
if pos("B", CROPS_PRODUCED_CHECKBOX) > 0 then
    CROPS_PRODUCED_REPEATING(2) = 1;
else
    CROPS_PRODUCED_REPEATING(2) = 0;
endif;
if pos("C", CROPS_PRODUCED_CHECKBOX) > 0 then
    CROPS_PRODUCED_REPEATING(3) = 1;
else
    CROPS_PRODUCED_REPEATING(3) = 0;
endif;
```

This works but it is a lot of code when the number of options is long. We can simplify it using a loop and a clever trick.

```
PROC CROPS_PRODUCED_CHECKBOX
```

```
// Fill in crops repeating item based on crops checkboxes
numeric i;
string alphabet = "ABCDEFG"; // used to convert from index to letter

// Loop from 1 (Maize) to 7 (Groundnuts) and for
// each crop type find out it if it was checked by
// seeing if the ith letter of the alphabet is in the
// checkbox field.
do i = 1 while i <= 7
    if pos(alphabet[i:1], CROPS_PRODUCED_CHECKBOX) > 0 then
        CROPS_PRODUCED_REPEATING(i) = 1;
    else
        CROPS_PRODUCED_REPEATING(i) = 0;
    endif;
enddo;
```

Group Exercise:

Add question G2, crops sold, to the crops form. Use checkboxes to represent the crops. Add a second variable called CROPS_SOLD_REPEATED with multiple occurrences and a value set of Yes-1 No-0 and convert the results from the checkbox to the repeating variable.

Dynamic Value Sets

It is often useful to change the value set for a question from logic. This can be done using the command *setvalueset*. Let's start with a simple example. Currently our value set for relationship in section B has labels like "Son/daughter" and "Brother/sister" to allow for both genders. However, when we show the relationship value set we already know the gender of the household member so we could show "son" for males and "daughter" for females. To do this we create two new value sets for relationship in the dictionary: RELATIONSHIP_MALE and RELATIONSHIP_FEMALE. Then in the onfocus of relationship we choose between the two value sets:

```
PROC RELATIONSHIP
onfocus
// Show male or female version of value set depending on sex of the person.
if SEX = 1 then
    setvalueset(RELATIONSHIP, RELATIONSHIP_MALE);
else
    setvalueset(RELATIONSHIP, RELATIONSHIP_FEMALE);
endif;
```

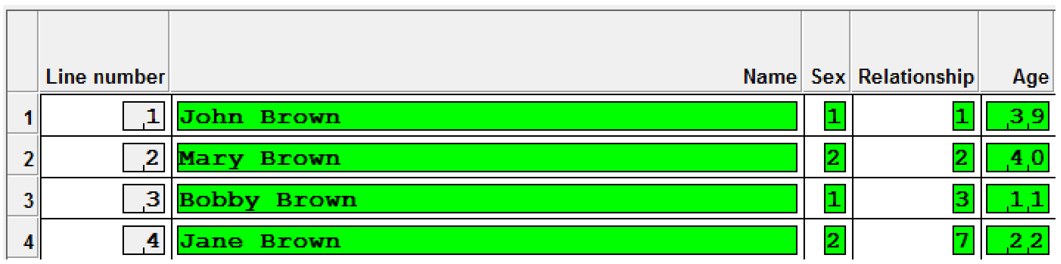
For question C8, id number of mother, we would like to create a value set from the names and line numbers of the eligible women in the section B roster. To do this we need the second form of setvalueset that takes an array of codes and an array of labels. This will allow us to create the list of names in logic instead of in the dictionary. First we need to declare the two arrays in the PROC global.

```
PROC GLOBAL
```

```
numeric currentLineNumber;
array string labels(30);
array numeric codes(30);
```

An array logic variable is similar to a dictionary item with occurrences. A numeric array of length seven stores seven numbers, each of which is accessed through subscripts.

We will fill in the two arrays of codes and labels with names and line numbers of the eligible women in the household. For example if we have the following household:



	Line number	Name	Sex	Relationship	Age
1	1	John Brown	1	1	39
2	2	Mary Brown	2	2	40
3	3	Bobby Brown	1	3	11
4	4	Jane Brown	2	7	22

We would fill in the two arrays as follows:

Subscript	Codes	Labels
1	2	Mary Brown
2	4	Jane Brown
3	87	Non-resident
4	88	Dead
5	notappl	

To do this in logic we need to loop through the section B roster and add an entry into our arrays for each eligible woman:

```
PROC CHILD_MOTHER
onfocus
// Create the value set for child mother from all eligible
// women in section B
numeric indexRoster;
numeric nextEntryValueSet = 1;
do indexRoster = 1 while indexRoster <= totocc(ADULTS000)

    if SEX(indexRoster) = 2 and AGE(indexRoster) >= 12 then
        labels(nextEntryValueSet) = NAME(indexRoster);
        codes(nextEntryValueSet) = indexRoster;
        nextEntryValueSet = nextEntryValueSet + 1;
    endif;
enddo;
```

Then we need to add the special codes for non-resident and dead to the arrays:

```
labels(nextEntryValueSet) = "non-resident";
codes(nextEntryValueSet) = 87;
nextEntryValueSet = nextEntryValueSet + 1;
labels(nextEntryValueSet) = "dead";
codes(nextEntryValueSet) = 88;
nextEntryValueSet = nextEntryValueSet + 1;
```

Finally we need to terminate the array of codes with a *notappl* to tell CSPro not to use the whole array and then pass the array of codes and the array labels to the *setvalueset* command.

```
codes(nextEntryValueSet) = notappl;
setvalueset(CHILD_MOTHER, codes, labels);
```

Group Exercise:

Add questions B7 and B8, languages spoken and main language. Use checkboxes for languages spoken and then create a dynamic value set for main language that lists only the languages chosen in the first question. For example if in question B7 the interviewer checks English, French and Chinese then the value set for main language should only have those three choices. If the interviewer only picks one language in B7 then automatically set that as the main language and go directly to the next question. How will you handle the case where no language is selected in B7?

Dynamic Checkboxes

Let's add question F3, "were assets purchased with a loan?", to the assets form. Rather than a series of yes/no question, we can implement this using a single variable with checkboxes. We could have one checkbox for each of the 12 items in the assets roster but it would be better if we only displayed the checkboxes for the assets that the household actually possesses. How do we know if the household possesses an item? The household possesses the item if its quantity is greater than zero. We need to loop through the lines of the roster and add a checkbox to the value set for each item with quantity greater than zero. The only tricky part is that since these are checkboxes we need to use alpha values. Since there are up to 12 types of assets we create a dictionary variable of type alpha length 12. Which record does it go on? It needs to go on a single record so lets it put it on the household characteristics record. Drag it onto the assets form.

In order to create a value set with alpha values we need a code array of type string. Where do we declare it? PROC GLOBAL of course.

```
PROC GLOBAL
numeric currentLineNumber;
array string labels(100);
array numeric codes(100);
array string codesString(100);
```

Now we build the value set in the onfocus of the checkboxes field. We use the alphabet trick again to get the alpha codes from the occurrence number. We also use the function *getocclabel()* to get the occurrence label from the assets roster to use in the value set.

```
PROC ASSETS_PURCHASED_WITH_LOAN

onfocus
// Create the dynamic checkbox value set from assets that have quantity > 0
numeric i;
numeric nextValue = 1;
string alphabet = "ABCDEFGHIJKL";
do i = 1 while i <= totocc(ASSETS000)
    if QUANTITY(i) > 0 then
        // Add to value set
        labels(nextValue) = getocclabel(ASSETS000(i));
        codesString(nextValue) = alphabet[i:1];
        nextValue = nextValue + 1;
    endif;
enddo;
codesString(nextValue) = "";
setvalueset(ASSETS_PURCHASED_WITH_LOAN, codesString, labels);
```

Group Exercise:

Return to question G2, crops sold. Instead of displaying checkboxes for all possible crops, limit the checkboxes to only those crops that selected in question G1, crops produced. Use setvalueset() to generate the value set for question G2 based on the selected options in question G1.

Exercises

1. Add question text to the rest of sections B and to sections C and D. Use a fill to include the name as we did in the examples. For section D you will need to use a logic variable for the name since section D does not have a name variable.
2. For question D05, calculate the total births based on the answers to the previous questions (be careful of notappl). Use a fill to show the total births in the question text and if they answer “no” to D05 go back to D02.
3. Add section H, livestock. Use a roster with occurrence labels. Add the question text and use the occurrence labels in the question text as we did in the example.
4. In section D, fertility, for the IDNO use a dynamic value set to list the names of all eligible women.
5. Implement a dynamic value set for the IDNO of the father in question C10 that lists the names of eligible fathers (male, 12 years and older).
6. Add question G6 to the crops form. Use checkboxes where the options are the school age children in the household (school age children are between 5 and 25 and will be in the section B roster). Create the value set for G6 using setvalueset.
7. Skip question G6 if “school fees” was NOT checked in question G5.

8. Add question G3, ranking, to the crops form. Use a variable with 7 occurrences where the first occurrence is the code of the top ranked crop, the second occurrence is the code of the second most important crop… Use dynamic value sets to list the crops for each occurrence so that it is not possible to pick the same crop twice. You can use getocclabel() to get the labels from the crops produced roster. Bonus: display the current rankings in the question text using a logic variable and a fill.