Session 5: Advanced Data Entry

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

- Create cascading questions
- Use the date picker control and do consistency checks on dates
- Use field notes
- Implement questions with other (specify)
- Use the prompt function
- Create CSPro applications that support multiple languages
- Use lookup files in data entry

Cascading Questions

Let's revisit question B5, place of birth. Rather than present a giant list of all countries in the world, it would be nice to allow the interviewer to narrow down the list first by continent. This is simple given the way that codes for place of birth are structured. The first digit is the continent and the next two digits are the country. We can create subitems for the continent and country and drag those onto the form instead of the parent items. Then once we know the continent we can loop through the value set containing all the countries to find those that are in the selected continent.

```
PROC COUNTRY_OF_BIRTH

onfocus
// Create value set based on continent selected in previous question
numeric i;
do i = 1 while i < 100
    string label = getlabel(PLACE_OF_BIRTH, CONTINENT_OF_BIRTH * 100 + i);
    // A blank label means that there are no more countries in the chosen
    // continent so we can break out of the loop early.
    if label = "" then
        break;
    endif;
    codes(i) = i;
    labels(i) = label;
enddo;
codes(i) = notappl;
setvalueset(COUNTRY_OF_BIRTH, codes, labels);
```

Make sure that we use zero fill for the country subitem so that we don't end up with blanks in the middle of your place of birth.

Also when removing the place of birth item we will lose the skip in the place of birth proc and we will need to add it into the proc for the subitem.

Date Pickers

Let's use a date control for the child's date of birth in section C. First we need to remove the subitems for day, month and year and then drag the 8 digit date item onto the form. Then set the capture type to date picker and run the application.

Manipulating Dates

Let's implement a consistency check between the date of birth of the child and the age in months. Which proc does the check go in? It will go in age in months since age in months comes after date of birth.

We can use the function *datediff()* which will calculate the difference between two.

```
PROC CHILD_AGE_IN_MONTHS
// Ensure that date of birth and age are consistent

numeric ageInMonthsFromDOB = datediff(CHILD_DATE_OF_BIRTH,
                                      INTERVIEW_DATE, "m");

if ageInMonthsFromDOB <> CHILD_AGE_IN_MONTHS then
    errmsg("Date of birth (%d/%d/%d) and age (%d) are inconsistent.",
           CHILD_DOB_DAY, CHILD_DOB_MONTH,
           CHILD_DOB_YEAR, CHILD_AGE_IN_MONTHS)
           select("Correct date of birth", CHILD_DATE_OF_BIRTH,
                  "Correct age", CHILD_AGE_IN_MONTHS);
endif;
```

Note that we use the date of interview rather than the current date in case the data is revisited after the interview.

If this check fails it may be that the subitems used for either the interview date or the date of birth do not have zero fill on. If this is the case then ageInMonthsFromDOB can result in default causing the error message to trigger.

Handling Other (specify)

In the household characteristics section question E05 has an option for other (specify). How can we handle questions like this? There are multiple techniques.

Other (specify) with skip

The simplest way to implement other (specify) is to add an alpha variable in the dictionary that comes directly after E05 and skip it if the answer is anything but other (specify).

First add the new variable ROOFING_MATERIAL_OTHER to the dictionary and put in on the form after E05. Then add the following logic to implement the skip pattern.

```
PROC ROOFING_MATERIAL

// Skip other (specify) field if other not selected
if ROOFING_MATERIAL <> 4 then
```

```
    skip to WALL_MATERIAL;
endif;
```

Other (specify) with field notes

Some people don't like adding additional fields to their forms. The alpha fields for other (specify) often take up a lot of space, especially in a roster. An alternative is to use field notes. Every field on a form can have a note. Notes are stored in a separate file and are saved with the case identifiers and the name of the variable. The interviewer can add notes at any time by choosing "Edit Field Note…" from the edit menu on Windows or by clicking on the field note icon on Android. You can also open the note editor using the command *editnote()*, read the note for a field with *getnote()*, and set it with *putnote()*.

Instead of adding a field to the form you can call editnote() if the interviewer chooses other (specify).

```
PROC ROOFING_MATERIAL

// Use field note for other (specify)
if ROOFING_MATERIAL = 4 then
    editnote();
endif;
```

You can then extract the other (specify) responses from the notes file or if you want to add them to your main data file then you copy from the note into a dictionary variable.

```
PROC ROOFING_MATERIAL

// Use field note for other (specify)
if ROOFING MATERIAL = 4 then
    string oldNote = getnote();
    putnote(strip(ROOFING_MATERIAL_OTHER)); // put current value of other in
note field
    ROOFING_MATERIAL_OTHER = editnote();
    putnote(oldNote); // restore old note
endif;
```

You do not have to copy the notes into a dictionary variable. It is possible to leave the notes in the notes file and later create a CSPro dictionary to match your notes file so that you can analyze or export it. Look up "notes file" in the help to for an explanation.

Other (specify) with prompt

A third alternative is to use the new prompt() function available in the current beta version of CSPro 6.2. It is like editnote() except that instead of storing the value in the notes file it simply returns the value.

```
PROC ROOFING_MATERIAL
// Use prompt for other (specify)
if ROOFING_MATERIAL = 4 then
```

```
    ROOFING_MATERIAL_OTHER = prompt("Specify other type of roofing material",
strip(ROOFING_MATERIAL_OTHER) );
endif;
```

Multiple Language Question Text

For many surveys it is useful to support multiple languages. In CSPro you can enter translations of the question text in the question text editor. To add a new language go to the “CAPI Options” menu and choose “Define CAPI Languages”. Add a new language by entering the name and label in the dialog that comes up. Let’s add support for French to our application. Define a new language with name ”FRA” and label ”French”. Now when you go to the question text editor you will see an additional panel for editing the question text for French. Add the following translations for questions B2-B4:

> Quel âge a %NAME%?
>
> Est-ce que %NAME% est masculin ou feminin?
>
> Quel est le lien de parenté entre %NAME% est le chef de ménage?

Now run your application and choose “Change Language” from the options menu to see your translated text.

Multiple Language Dictionaries

In addition to translating the question text you may also need to translate the labels and the responses. To add a new language to the dictionary, right click on the dictionary in the tree control on the left side of the screen and choose “Edit Languages”. Now when editing the dictionary you will see a dropdown menu at the top of the screen that you can use to switch between languages. Initially when we switch to French everything is still in English but we can edit all the labels (item, record, value set…) the way we normally do in the dictionary and we will be changing only the French versions. We can switch back to English to see that the original English labels are still there. Let’s modify the variable SEX in section B by changing the label to “Sexe” and changing the values to “Masculin” and “Feminin”. Now when we run the application we change the language to French we see the French version of our value set and on Android we see the French version of our label too.

Lookup Files

It is currently possible for the interviewer to enter a village code that is not valid for the district that was selected. In order to verify that the village code is valid for the district we need to use the list of district and village codes in the annex of the questionnaire. We can do this by creating a lookup file from the Excel spreadsheet.

First we need to create a dictionary for our lookup file. For this task we want to be able to query if the combination of the district and village code is a valid village. In annex 3 of the questionnaire we have a table with the district code, village code and village name. We can use this as a lookup file where the

keys are the district and village code and the village name is the value. We create a dictionary by going to “Add Files” in the “File” menu add entering the name of a new external dictionary. Let’s call it “villages.dcf”. This dictionary will have the district code and village code as the id-items and a single record containing a single item: the village name. In order to avoid name conflicts with the main dictionary we can use VL_DISTRICT_CODE, VL_VILLAGE_CODE and VL_VILLAGE_NAME as the variable names. We must make sure that the lengths and zero-fill settings of the id-items exactly match those in the main dictionary otherwise we won’t be able to use the variables from the main dictionary as keys for the lookup.

Once we have the dictionary we need to convert the Excel spreadsheet into a CSPro data file. We can use the Excel2CSPro tool to do this. We point it at the dictionary and spreadsheet, set the worksheet number and start row, enter the name of the resulting data file (let’s call it “villages.dat”) and hit “Convert”. We can view the resulting file in TextViewer to make sure that it matches the Excel spreadsheet.

Finally in the village proc we use the *loadcase*() command to lookup the district and village codes in the file. Loadcase takes the name of the dictionary (VILLAGES_DICT) and the values to use as keys (id-items) for the lookup. For example to lookup district 3, village 6 we would do:

```
loadcase(VILLAGES_DICT, 3, 6)
```

If loadcase finds a record in the lookup file with district code 3 and village code 6 it will return 1 and set the variables in VILLAGES_DICT to the values from the case it found. In this case that means setting the two id-items VL_DISTRICT_CODE, VL_VILLAGE_CODE and the variable VL_VILLAGE_NAME.

We can use this to test if the district and village codes are valid in the VILLAGE proc:

```
PROC VILLAGE

// Verify that the village code is valid for the district
// selected.
if loadcase(VILLAGES_DICT, DISTRICT, VILLAGE)  = 0 then
    errmsg("Village code %d is not valid for district %s",
           VILLAGE, getlabel(DISTRICT, DISTRICT));
    reenter;
else
    errmsg("You have selected village: %s", VL_VILLAGE_NAME);
endif;
```

Note that we are using the DISTRICT and VILLAGE from the main dictionary as arguments to *loadcase*, not the id-items from the villages dictionary. Before calling *loadcase* the id-items for the external dictionary are all blank. They are only set if *loadcase* is successful.

Note that when you run this application, in addition to copying the .pen and .pff files to the Android device, you must now also copy the lookup file (the .dat file).

We can now add alpha variables to the main dictionary, assign the district and village names to them and display them on the form as protected fields:

```
PROC DISTRICT

// Assign district name to dictionary variable
// so we can display it on form.
DISTRICT_NAME = getlabel(DISTRICT, DISTRICT);
```

```
PROC VILLAGE

// Verify that the village code is valid for the district
// selected.
if loadcase(VILLAGES_DICT, DISTRICT, VILLAGE)  = 0 then
    errmsg("Village code %d is not valid for district %s",
           VILLAGE, getlabel(DISTRICT, DISTRICT));
    reenter;
else
    // Assign village name from lookup
    // to main dictionary variable so we can
    // display it on form.
    VILLAGE_NAME = VL_VILLAGE_NAME;
endif;
```

Group Exercise: GPS Lookup File

Automatically fill in the GPS coordinates using a lookup file based on the district, village and household number. Use the Excel file HouseholdGPS.xls that contains columns for the district, village, household number, latitude and longitude. First create a dictionary for the lookup with these columns as variables. Which will be the id-items? Use Excel2CSPro to generate the data file. Create the variables for latitude and longitude in the main dictionary. Make sure to use at least length 14 with 9 decimal places (including decimal character) to have enough precision. Drag these variables onto the section A form. Add the following code the LATITUDE preproc to do the lookup. If the lookup succeeds, fill in the latitude and longitude fields on the form with the values from the lookup file and then go to the next form.

As a final example we will combine the lookup file with dynamic value sets to create the dynamic value set for the village. Up to now our lookup files have had a one to one correspondence between keys and values. For this case we need to get a list of multiple villages per district. To do this we make a new dictionary containing the district as the only id-item and add a multiply occurring record to contain the village codes. Let's call this dictionary "villagesperdistrict.dcf". Create the lookup file for it from the village codes in annex 2 of the questionnaire. This data file will actually be identical to the one we created for checking the village and district. Unfortunately CSPro will not let us use the same data file for two different external dictionaries (since both could potentially modify the file simultaneously) so we will need to create a copy.

Once we have the dictionary and lookup file we use loadcase as before, however now the case that is loaded contains more than one record occurrence. There will be one occurrence for each village in the district. We loop through all the villages and them to the codes and labels array to make the dynamic value set.

```
PROC VILLAGE

onfocus
// Create dynamic value set of villages in selected district
// using lookup file
if loadcase(VILLAGESPERDISTRICT_DICT, DISTRICT) = 1 then
    // In the lookup file, VILLAGESPERDISTRICT_REC is a multiply occurring
    // record since there is more than one village per district. We
    // need to iterate through all the villages in the district
    // and them to the value set.
    numeric i;
    do i = 1 while i <=
count(VILLAGESPERDISTRICT_DICT.VILLAGESPERDISTRICT_REC)
        labels(i) = VPD_VILLAGE_NAME(i);
        codes(i) = VPD_VILLAGE_CODE(i);
    enddo;
    codes(i) = notappl;
    setvalueset(VILLAGE, codes, labels);
endif;
```

Exercises

1. For B10, occupation, allow the user to choose the occupation based on the hierarchy. Let them first choose from the 1 digit categories (Managers, Professionals, Technicians and associate professionals…) then from the 2 digit categories that fall under the top-level category chosen, then the three digit categories and finally the 4 digit categories. If you had previously just copied the 4 digit codes from the questionnaire to your value set you will need to update the value set to include the 1, 2 and 3 digit codes as well.
2. Record the other (specify) for the owner in the assets roster in section F. You may use a skip, a field note or prompt.
3. Add another language to the application (your choice) and add translated question text for the questions in section B.
4. Add translated versions (in the language of your choice) of the labels and value sets for sex and relationship in section B.
5. Implement a check on the minimum and maximum per unit asset values in section F. Use the spreadsheet in annex 5 to create a lookup file containing the asset code, minimum value and maximum value. Use the *loadcase* command with this lookup file to find the minimum and maximum values for the selected asset and show an error message if the per unit asset value entered in the roster is below the minimum value or above the maximum value. This should be a soft check.
6. Use the HouseholdGPS spreadsheet to add a dynamic value set for the household number field in section A. Hint: you cannot use the same dictionary that we used for pre-filling the GPS

coordinates. It will be similar but the id-items and number of record occurrences will be different.