Session 3: Flow Control and Navigation

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

- Add custom controls using userbar
- Implement navigation
- Understand when to use skip, advance and move
- Understand when to use visualvalue
- Implement flow control using a navigation field
- Implement flow control using multiple applications

The Userbar

From logic we can add buttons to the CSEntry user interface that call user-defined functions. Here is how to add a userbar button that will create an errmsg dialog that says "hello". First, we define the function hello in the PROC GLOBAL:

```
function hello()
    errmsg("Hello");
end;
```

Then in the preproc of the application we add it to the userbar:

```
PROC HOUSEHOLD_FF
preproc
userbar(clear);
userbar(add button, "Hello", hello);
userbar(show);
```

When adding a button in the preproc of the application it is important to call `clear` first, otherwise we can end up with two or three copies of the same button if we start the application multiple times. We added the button in the preproc of the application but you can add or remove buttons in any proc so you can, for example, only a show button within a certain field or a certain roster.

Navigation

Let's try a more interesting example. Let's add a "Go To..." button that will let the user navigate directly to a particular section of the questionnaire.

```
// Userbar function for navigating directly to different parts of
// questionnaire.
function goto()
    numeric section = accept("Go to?",
        "Section B: Roster",
        "Section C: Health",
        "Section D: Food Security",
        "Section E: Consumption");

    if section = 1 then
        skip to SECTION_B_HOUSEHOLD_ROSTER_FORM;
    elseif section = 2 then
        skip to SECTION_C_HEALTH_FORM;
    elseif section = 3 then
        skip to SECTION_D_FOOD_SECURITY_FORM;
    elseif section = 4 then
        skip to SECTION_E_CONSUMPTION_FORM;
    endif;
end;

PROC HOUSEHOLD_FF
preproc
userbar(clear);
userbar(add button, "Go To...", goto);
userbar(show);
```

Off Path Variables

The above will let the interviewer jump from one section to another but using `skip` means that if we use our Go To... button to jump over a section, that section will end up skipped and won't be saved in the data file. This is because skipped values are considered “off path”. In system-controlled mode, CSEntry keeps track of which variables are on and off the "path". Variables that you have entered are considered "on path" but those that have been skipped, even if there was a value in them before they were skipped, are considered "off path". As we have seen before, variables that are "off path" are considered blank (notappl) in logic. It turns out that variables that are ahead of the current field are also considered "off path" until you pass through them. The idea is that these fields have not yet been validated by running their preproc and postproc with the current values of all preceding fields and therefore cannot be considered final. The effect of this is that the values of all fields ahead of the current field in the questionnaire are notappl in logic.

You can see which fields are "on path" by looking at the background color of the field:

- Green: on path
- Dark Grey: skipped
- White: not yet filled in
- Light grey: protected

Advance and Move

Instead of using `skip` we can use `advance` which moves forward in the questionnaire without marking fields as skipped. Using `advance` also runs all the preprocs and postprocs of the fields that are passed through to ensure that no consistency or out of range checks are missed.

```
function goto()
    numeric section = accept("Go to?",
        "Section B: Roster",
        "Section C: Health",
        "Section D: Food Security",
        "Section E: Consumption");

    if section = 1 then
        advance to SECTION_B_HOUSEHOLD_ROSTER_FORM;
    elseif section = 2 then
        advance to SECTION_C_HEALTH_FORM;
    elseif section = 3 then
        advance to SECTION_D_FOOD_SECURITY_FORM;
    elseif section = 4 then
        advance to SECTION_E_CONSUMPTION_FORM;
    endif;
end;
```

This stops the function from skipping over data when navigating, however it still has a limitation. We can only navigate forward in the questionnaire. To go backwards we need to use `reenter`, but in our goto() function we don't know if the user wants to move forward or backward. Fortunately, CSPro provides the command `move` which will use either `skip` or `reenter` as appropriate. By default, when going forward, `move` does a skip but you can add `advance` after the field name to make it do an advance instead of a skip.

```
function goto()
    numeric section = accept("Go to?",
        "Section B: Roster",
        "Section C: Health",
        "Section D: Food Security",
        "Section E: Consumption");

    if section = 1 then
        move to SECTION_B_HOUSEHOLD_ROSTER_FORM advance;
    elseif section = 2 then
        move to SECTION_C_HEALTH_FORM advance;
    elseif section = 3 then
        move to SECTION_D_FOOD_SECURITY_FORM advance;
    elseif section = 4 then
        move to SECTION_E_CONSUMPTION_FORM advance;
    endif;
end;
```

Flow Control

We can now easily jump from one section to another, but sometimes in long and complex surveys, you would like to be able to complete sections in the survey out of order. For example, if there is no one at home who can answer the health section (C) you could instead complete the food security section (D) and later come back to the health section. However, in system-controlled mode, CSPro forces you to complete the questionnaire in a linear fashion. You must complete all prior forms before starting on the next form, so you must complete section C before starting section D. We can see this if we try to navigate to section E from our userbar button if we have not yet filled in section D.

There are, however, a couple of ways to work around this behavior to permit answering the sections out of the usual order. The first technique is similar to what we have already built but goes back to using skips to move around the questionnaire and the second technique implements a single questionnaire as multiple data entry applications.

Flow Control with a Navigation Field

Currently we cannot ask sections out of order because we are using advance when moving forward rather than using skip. As we saw earlier, using skip allows us to go out of order but has the disadvantage that if we skip a section that we had previously filled in, we those values are not saved to the data file, and we are not guaranteed to have run all of our consistency checks. This can lead to errors in the data.

We can fix this by using skip for navigation while entering data and then using a final advance from the start of the questionnaire to the end once all the individual sections are completed. This final advance will visit every field in the questionnaire in order and run the consistency checks to ensure that there are no errors in fields that we skipped over.

To make sure we advance over the whole questionnaire, we need to add a field at the start of the questionnaire that we will initiate the final advance from. We will add this new field just after section A so that at least the identifiers are complete before we start navigating. Since we now have a field for navigation, we may as well use it to choose the section to go to instead of choosing it in our userbar function.

Add a new item in the dictionary named NAVIGATION. We can add it to the interview record. The value set will be:

Section B – 1
Section C – 2
Section D – 3
Section E – 4

We will create a new form named "navigation form" in between section A and section B and drop this variable onto it. In the postproc of the field we will use skip to move the section chosen.

```
PROC NAVIGATION

onfocus

// Clear previous choice
$ = notappl;

postproc

// Go to selected section
if $ = 1 then
    skip to SECTION_B_HOUSEHOLD_ROSTER_FORM;
elseif $ = 2 then
    skip to SECTION_C_HEALTH_FORM;
elseif $ = 3 then
    skip to SECTION_D_FOOD_SECURITY_FORM;
elseif $ = 4 then
    skip to SECTION_E_CONSUMPTION_FORM;
endif;
```

Now our userbar function can simply go to the navigation field.

First, we need a function in the proc global.

```
function goto()
    reenter NAVIGATION;
end;
```

Now we need to add the final advance to ensure the consistency checks are run and all fields that should be on path are saved. Let's add another option to the value set for NAVIGATION called "Validate and Complete". When the interviewer chooses this option, we use advance to go from the NAVIGATION field to the end of the questionnaire. Advance will run all of the consistency checks for all of the fields and it will stop if it encounters any errors. This will ensure that no sections are left blank, and no inconsistencies were added by entering data out of order.

```
PROC NAVIGATION

postproc

// Go to selected section
if $ = 1 then
    reenter SECTION_B_HOUSEHOLD_ROSTER_FORM;
elseif $ = 2 then
    skip to SECTION_C_HEALTH_FORM;
elseif $ = 3 then
    skip to SECTION_D_FOOD_SECURITY_FORM;
elseif $ = 4 then
    skip to SECTION_E_CONSUMPTION_FORM;
elseif $ = 9 then
    // Validate and complete
    advance;
endif;
```

This fixes the problem of skipped fields getting erased, but only if the interviewer makes sure to use the NAVIGATION field to select “validate and complete”. If instead they just complete the final section after having skipped earlier sections, the skipped sections will still be erased. To prevent this, we will force the interviewer to return to NAVIGATION before completing the questionnaire. We will add a new field named QUESTIONNAIRE_COMPLETE and put it on a new form at the end of the questionnaire. In the preproc for QUESTIONNAIRE_COMPLETE we will reenter the field NAVIGATION to force validation. In order to allow completion of the questionnaire from the NAVIGATION field, we will add a logic variable named inFinalValidation. We will set it to false in the preproc of NAVIGATION and set it to true before calling advance. In the preproc of QUESTIONNAIRE_COMPLETE we will check the value of inFinalValidation and only return to NAVIGATION if inFinalValidation is false. This way, the only way to get to the end of the questionnaire will be to do it by choosing "validate and complete" from NAVIGATION.

```
PROC QUESTIONNAIRE_COMPLETE
preproc

// Only allow completion of questionnaire if coming
// from NAVIGATION, otherwise return to NAVIGATION
if not inFinalValidation then
    reenter NAVIGATION;
endif;
$ = 1;


PROC NAVIGATION

preproc
// By default not in final validation
inFinalValidation = 0;

onfocus

// Clear previous choice
$ = notappl;

postproc

// Go to selected section
if $ = 1 then
    skip to SECTION_B_HOUSEHOLD_ROSTER_FORM;
elseif $ = 2 then
    skip to SECTION_C_HEALTH_FORM;
elseif $ = 3 then
    skip to SECTION_D_FOOD_SECURITY_FORM;
elseif $ = 4 then
    skip to SECTION_E_CONSUMPTION_FORM;
elseif $ = 9 then
    // Validate and complete
    inFinalValidation = 1;
    advance;
endif;
```

We can make the questionnaire complete field protected and set it to true when the questionnaire has been validated.

There are consistency checks and roster control fields in section C and D that use values from section B. If we skip over section B to section C or D, these consistency checks will fail because the values in section B have been skipped and are considered blank. We could try using `visualvalue` to access the values but not only does this complicate our code, but if the

interviewer tries to complete section C before completing section B, even with `visualvalue`, the values will still be blank. To avoid this, we can move the navigation form to be after section B. This forces the interviewer to complete section B before using the navigation menu. Now in the NAVIGATION postproc we need to use reenter instead of skip for section B.

```
postproc

// Go to selected section
if $ = 1 then
    reenter SECTION_B_HOUSEHOLD_ROSTER_FORM;
elseif $ = 2 then
    skip to SECTION_C_HEALTH_FORM;
elseif $ = 3 then
    skip to SECTION_D_FOOD_SECURITY_FORM;
elseif $ = 4 then
    skip to SECTION_E_CONSUMPTION_FORM;
elseif $ = 9 then
    // Validate and complete
    inFinalValidation = 1;
    advance;
endif;
```

We now have a working flow control scheme, but we can improve it by showing the status of each section when in the navigation field. To do that we add a variable in the dictionary for the completion status of each section with complete – 1 and incomplete – notappl. At the end of each section, we set the status variable for the section to complete. In the NAVIGATION field we can show the status in the question text. To ensure that sections do not stay marked as complete when they are modified, we need to also set the status to incomplete at the beginning of each section. Since each section is on its own form, we can do this in the preproc and postproc of the form for the section.

```
PROC SECTION_D_FOOD_SECURITY_FORM
preproc
// Start section, mark incomplete
SECTION_D_COMPLETE = notappl;

postproc
// End section, mark complete
SECTION_D_COMPLETE = 1;
```

To show the status in the question text we can use fills in the question text for each of the status variables:

> *You must the complete the following sections of the questionnaire:*
>
> *Section B: ~~getvaluelabel(SECTION_B_COMPLETE)~~*
> *Section C: ~~getvaluelabel(SECTION_C_COMPLETE)~~*
> *Section D: ~~getvaluelabel(SECTION_D_COMPLETE)~~*
> *Section E: ~~getvaluelabel(SECTION_E_COMPLETE)~~*
>
> *Choose a section to go to*

If we want to be fancy, we could use a table in the question text and add a check mark for completed sections like this:

Section B	✅
Section C	✅
Section D	
Section E	

Now every time we enter the navigation field, we see the status of each of the questionnaire sections.

Note that we did not put any of the questionnaire status variables onto a form. If we had put them onto the form for each section we would have to use `visualvalue` to read their values to build the question text. When a variable is not on a form, you never have to use `visualvalue` since the variable can't be skipped or be ahead of the current field.

Group Exercise

Whenever the interviewer runs the GoToNavigation() save the name of the current field for the section that they are currently on. Then when they restart that section, use advance to move them back to the field they were on when they navigated away from that section. For example, if the user is on the field D02 in section D and they tap "Navigate" on the userbar to work on another section and then use navigate again to go back to section D, the program should automatically put them directly into D02 again.

You will need variables in the dictionary to store the last field for each section: one variable for the last field in section C, one for the last field in section D, and one for the last field in section E. You can get the name of the current field as a string using the function **getsymbol()** and you can get the name of the record that the variable is on using **getrecord(getsymbol())**. To move to the saved field, use advance to the saved field name in the preproc of the form for the section.

Flow Control with Multiple Applications

An alternative approach to flow control is to implement each section as a separate application and have the main application launch the section applications using a pff variablethe same way we launch the household applications from the menu application. By making each section a separate application, each section has its own flow, so no fields ever need to be skipped.

Let's start by moving section C (health) into a separate application. Create a new application named "Section C". Put it in a subfolder of the Household folder also named "Section C". Copy the id-items from the household dictionary but change the names so they won't conflict when both dictionaries are used in the same application e.g. (C_PROVINCE, C_DISTRICT…). Copy the health-related fields from the person record into the household dictionary and paste it into a new section C record in the new dictionary with the same number of occurrences as the person record. Create forms in the section C application containing the id-items and the section C record.

Copy the logic and question text for section C from the household application into the new application. In order for the logic to work correctly, we need to access the variables in the household dictionary from the section C application. To do that, add the household dictionary as external dictionary in the section C application. In the postproc of C_HOUSEHOLD_NUMBER use loadcase to load the household roster:

```
PROC C_HOUSEHOLD_NUMBER

// Load the case from the main household file to get access to the
household roster
if not loadcase(HOUSEHOLD_DICT, C_PROVINCE, C_DISTRICT, C_CLUSTER,
                C_HOUSEHOLD_NUMBER) then
    errmsg("Invalid household identifiers");
    stop(1); // Back to menu application
endif;
```

You will need to fix errors and warnings in logic mainly due to the fact that since the health questions and the household roster items are no longer in the same record (or even the same dictionary), we need to use subscripts to correctly access the household roster variables.

To correctly size the roster in section C we need to add a new singly occurring record containing a variable for the number of household members, drag the variable onto the form before the roster and use it as the roster control field. Copy the value from the NUMBER_OF_HOUSEHOLD_MEMBERS variable in the household dictionary into this new variable.

The next step is to launch the section C application from the household application. We use OnExitPff to return to the main household application when the section C application is closed. Set the *key* parameter to the id-items for the current case.

```
function launchSectionC()

    pff sectionPff;
    sectionPff.load("SectionC/SectionC.pff");
    // Return to this application when the section is complete
    sectionPff.setProperty("OnExit", "../PopstanHouseholdSurvey.pff");
    // Set the id-items to id-items of the main household dictionary
    sectionPff.setProperty("key", key(HOUSEHOLD_DICT));
    sectionPff.exec();
end;
```

Modify the postproc of NAVIGATION to launch the section C application:

```
postproc

// Go to selected section
if $ = 1 then
    reenter SECTION_B_HOUSEHOLD_ROSTER_FORM;
elseif $ = 2 then
    savepartial();
    launchSectionC();
elseif $ = 3 then
```

Note that we must call `savepartial()` before launching the section C application so we do not lose any data.

Follow the same procedure to create separate data entry applications for sections D and E.
The postproc for NAVIGATION should now look like:

```
postproc

// Go to selected section
if $ = 1 then
    reenter SECTION_B_HOUSEHOLD_ROSTER_FORM;
elseif $ = 2 then
    savepartial();
    launchSectionC();
elseif $ = 3 then
    savepartial();
    launchSectionD();
elseif $ = 4 then
    savepartial();
    launchSectionE();
elseif $ = 9 then
    // do nothing which falls through to end of questionnaire
endif;
```

Note that we no longer need the `advance` to complete the questionnaire. Since NAVIGATION is now the last field in the household questionnaire application, we can simply allow the program to end by doing nothing. For all other selections, one of the section applications will be launched instead.

To update the completion status of the section we must add the dictionary for the section application as an external dictionary to the household application and use loadcase to check if the case has been completed in that dictionary. This can be done in the onfocus of NAVIGATION since that is where the completion status is displayed.

```
onfocus

// Clear previous choice
$ = notappl;

// Update completion statuses for sections
if loadcase(SECTIONC_DICT, PROVINCE, DISTRICT, CLUSTER, HOUSEHOLD_NUMB
ER)
    and not ispartial(SECTIONC_DICT) then
    SECTION_C_COMPLETE = 1;
else
    SECTION_C_COMPLETE = notappl;
endif;
```

By using multiple applications, we have a control flow that does not require skips that could erase data. The disadvantage of this approach is that our data is now spread out among multiple data files. We will need to merge these files together later for editing and analysis.

Group Exercise

Create a separate application for section D that is launched from the household data entry application. Make sure that the correct completion status is shown in the question text for the NAVIGATION field.