Session 8: Lookup Files and Coding

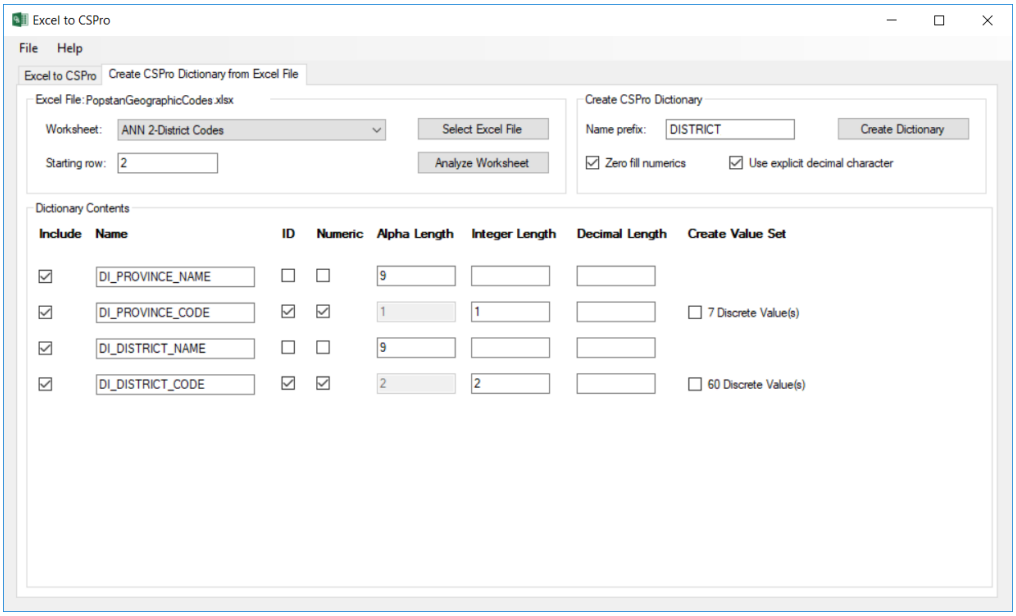
At the end of this session participants will be able to:

- Work with external dictionaries
- Import data from Excel to CSPro to create lookup files
- Create dynamic value sets from lookup files
- Capture complex hierarchical codes using multiple methods

Lookup Files

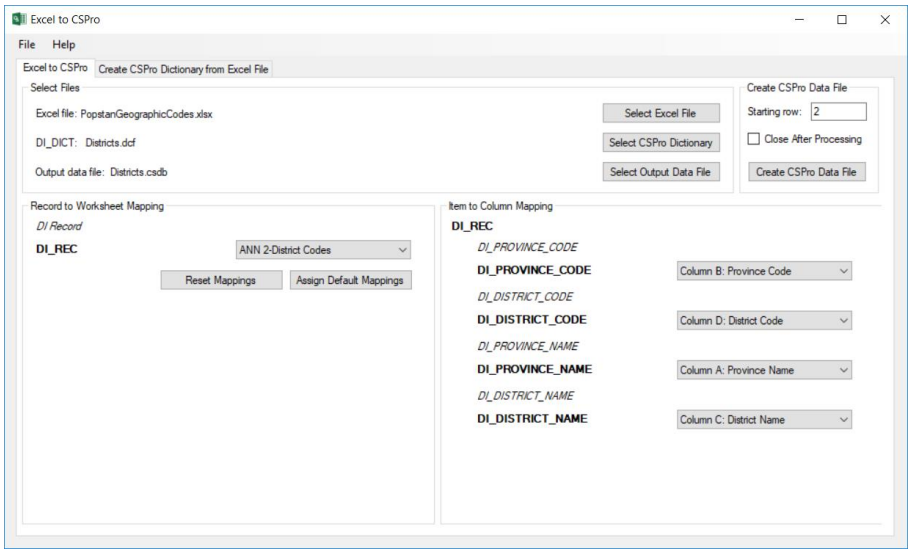
It is currently possible for the interviewer to enter a district code that is not valid for the province that was selected. In order to verify that the district code is valid for the province we need to use the list of province and district codes in the spreadsheet PopstanGeographicCodes.xlsx. We can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

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 province and district codes is a valid district. In the second tab of the spreadsheet we have a table with the province code, district code and district name. We can use this as a lookup file where the keys are the province and district code and the value is the district name. We can create a dictionary for this using the **Excel2CSPro** tool from the tools menu. Select the "Create CSPro dictionary from Excel File" tab, click "Select Excel File" and browse to the file PopstanGeographicCodes.xlsx. Choose "ANN2 – District Codes" and click "Analyze Worksheet". This detects the dictionary variables to create from the columns of the spreadsheet. The tool will show four variables, one for each column and will set the default names to column headers of the spreadsheet. Our lookup file dictionary will have the province code and district code as the id-items and province and district names as regular variables. Next to the province and district codes check the "ID" box to indicate that these should be ID items. In order to avoid name conflicts with the main dictionary we can use DI_PROVINCE, DI_DISTRICT and DI_DISTRICT_NAME as the variable names. Modify them in the Excel to CSPro tool.



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. Verify that the length of the province and district codes match those in the main dictionary and check the box "Zero fill numerics" to make sure that the id-items are zero filled. Once all settings are correct, click create dictionary to generate the CSPro dictionary file. Save it as "Districts.dcf" in the questionnaire folder.

Once we have the dictionary we need to convert the Excel spreadsheet into a CSPro data file. We can use the first tab of the **Excel2CSPro** tool to do this. Select the PopstanGeographicCodes.xlsx file again along with the Districts dictionary that we just created. Specify a new output data file to write the lookup file to. Under "Record to Worksheet Mapping" choose the second worksheet "ANN 2-District Codes". Under the "Item to Column Mapping" link the dictionary variables to the appropriate columns in the Excel spreadsheet: DI_PROVINCE_CODE to "Column B: Province Code", DI_DISTRICT_CODE to "Column D: District Code", etc… Finally, click "Create CSPro Data File" to generate the file. Verify the file in DataViewer to make it sure it was converted correctly.



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

```
loadcase(DISTRICTS_DICT, 3, 6)
```

If loadcase finds a record in the lookup file with province code 3 and district code 6 it will return 1 and set the variables in DISTRICTS_DICT to the values from the case it found. In this case that means setting the two id-items DI_PROVINCE, DI_DISTRICT and the variable DI_DISTRICT_NAME.

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

```
PROC DISTRICT

// Verify that the district code is valid for the province
// selected.
if loadcase(DISTRICTS_DICT, PROVINCE, DISTRICT)  = 0 then
    errmsg("District code %d is not valid for province %s",
           DISTRICT, getlabel(PROVINCE, PROVINCE));
    reenter;
else
    errmsg("You have selected district: %s", DI_DISTRICT_NAME);
endif;
```

Note that we are using the PROVINCE and DISTRICT from the main dictionary as arguments to *loadcase*, not the id-items from the districts 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 .csdb file).

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

```
PROC PROVINCE

// Assign province name to dictionary variable
// so we can display it on form.
PROVINCE_NAME = getlabel(PROVINCE, PROVINCE);
```

```
PROC DISTRICT

// Verify that the district code is valid for the province
// selected.
if loadcase(DISTRICTS_DICT, PROVINCE, DISTRICT)  = 0 then
    errmsg("District code %d is not valid for province %s",
           DISTRICT, getlabel(PROVINCE, PROVINCE));
    reenter;
else
    // Assign district name from lookup
    // to main dictionary variable so we can
    // display it on form.
    DISTRICT_NAME = DI_DISTRICT_NAME;
endif;
```

The Resource Folder

Instead of copying the lookup file to the Android file each time it is updated we can put it in the resource directory of the application. All files in the resource folder are built into the pen file and extracted on the device when the application is run. This makes it more convenient to distribute an application with lookup files. Let's create a resource folder, add it to our application and put our lookup file in it. To create the resource file just create a new folder in Windows explorer. We will create the folder "resources" in the household folder. Then in CSPro go to Add Files… from the File menu, choose "resource folder" and browse to the new folder. Now put the file districts.csdb in the resource file and rebuild the pen file. Note that very large lookup files will increase the size of the .pen file and cause updates to the application to use more data and take longer. In such cases, it may be better to use syncdata to update the lookup file separately from the .pen file, especially if the lookup file is updated less frequently than the .pen file.

Group Exercise

Add the staff file and login code check to the menu program. Create an Excel spreadsheet with the columns Name, Login Code, Role (supervisor/interviewer). Use Excel to CSPro to create a corresponding dictionary and data file named staff.dcf and staff.csdb respectively. To avoid name conflicts with other dictionary variables, add the prefix ST_ (for staff) the variable names. Add the dictionary as an external dictionary to the menu program. Change the menu program so that instead of just choosing interviewer/supervisor at the login screen, the user enters their login code. Use loadcase to lookup the login code in the lookup file. If the login code is not found, don't move past the login screen. If the login code is found, use the role from the lookup file to determine which screen to go to next.

Dynamic Value Set from a Lookup File

Another alternative is to combine the lookup file with setvalueset to create a dynamic value set for the district that includes only districts in the selected province. To do this we need to extract all the districts in the selected province from the lookup file. We can do this using the forcase loop which iterates over all cases in an external data file. By itself, forcase will loop through each case in the data file. You can also add an optional "where" clause to only go through the districts in the selected province.

```
PROC DISTRICT
onfocus
// Create dynamic value set of districts for selected
// province using lookup file
numeric nextEntry = 1;
forcase DI_DICT where DI_PROVINCE_CODE = PROVINCE do
    codes(nextEntry) = DI_DISTRICT_CODE;
    labels(nextEntry) = DI_DISTRICT_NAME;
    nextEntry = nextEntry + 1;
enddo;
codes(nextEntry) = notappl;
setvalueset(DISTRICT, codes, labels);
```

Coding Occupation and Industry

The occupation and industry question eventually need to be coded using international standards. These codes are large and complex. Depending on the context it may be better to simply enter the name of the occupation or industry and perform the coding in the main office. In contexts where it is possible to do the coding in the field there are multiple strategies to implement it. We will explore the following options:

- Value set search
- Drill down
- Hybrid Search/Drill down
- Index file

Value Set Search

The simplest solution is to create a value set containing all the codes and use the built-in value set search function in CSPro. First, we need to filter out all but the level four codes in Excel and rearrange the columns so that the labels are in one column and the codes are in the next column. Then we can select these two columns and paste them into the value sets for OMJISCO and IMJISIC. When we run this, a HUGE value set is presented to the interviewer but by tapping on the magnifying glass icon they can easily filter the value set to only show the relevant codes.

Drill down

A second option is to have the interviewer select the code hierarchically. First, they pick from the ten top level codes, then they are shown the list of second level codes for the first level code they chose, then the list of third level codes for the chosen second level code, and finally the fourth level codes for the third level code chosen. This can be implemented using dynamic value sets. In order to do this, we need to add subitems to ISCO/ISIC code for each level code and put those subitems on the form instead of the item itself. Call these subitems OMJISCO1, OMJISCO2…

There are multiple ways to implement the dynamic value sets in this approach. The simplest is to use a lookup file. Create an Excel spreadsheet with the ISCO codes broken up into 4 columns plus the description in the 5th column.

Level1	Level2	Level3	Level4	Description
1				Managers
1	1			Chief executives, senior officials and legislators
1	1	1		Legislators and senior officials
1	1	1	1	Legislators
1	1	1	2	Senior government officials
1	1	1	3	Traditional chiefs and heads of village
1	1	1	4	Senior officials of special-interest organizations
1	1	2		Managing directors and chief executives
1	1	2	0	Managing directors and chief executives

Use CSPro to Excel to create a dictionary and data file with the ISCO codes as the id-items and the description as the data item.

To create the value set for the first sub-item we want to loop through the lookup file and create a value set from all the cases where the level 2 id is blank. This should give us all of the level 1 codes.

```
PROC OMJISCO1
onfocus

// Set value set for top level codes from ISCO lookup file
numeric nextValueSetEntry = 1;
forcase ISIC_DICT where LEVEL2 = notappl do
    codes(nextValueSetEntry) = LEVEL1;
```

```
    labels(nextValueSetEntry) = DESCRIPTION;
    nextValueSetEntry = nextValueSetEntry + 1;
enddo;
codes(nextValueSetEntry) = notappl;
setvalueset(OMJISCO1, codes, labels);
```

For the second subitem the code is similar. This time we include all cases where the level 1 code matches the one just selected, the level 2 code is not blank and the level 3 code is blank. This should give us all the level 2 categories in the selected level 1 category.

```
PROC OMJISCO2
onfocus

// Set value set for 2nd level codes from ISCO lookup file
numeric nextValueSetEntry = 1;
forcase ISIC_DICT where LEVEL1 = OMJISCO1 and LEVEL2 <> notappl and LEVEL3 =
notappl do
    codes(nextValueSetEntry) = LEVEL2;
    labels(nextValueSetEntry) = DESCRIPTION;
    nextValueSetEntry = nextValueSetEntry + 1;
enddo;
codes(nextValueSetEntry) = notappl;
setvalueset(OMJISCO2, codes, labels);
```

The value sets for the other two subitems follow the same pattern.

Now when we run the application we are prompted to pick the ISCO code one category at a time resulting in a 4-digit code.

Hybrid Search/Drill Down

It is possible to combine these two options. When we create the value set for the first category we can add an 11th item “Show All”. This option will show the entire list of 4th level 4-digit codes as in the first example allowing the interviewer to use search. This gives the interviewer the option of either using categories or search.

To implement this, we can no longer have a single variable with subitems. We need to have separate 1-digit variables for the drill down and a 4-digit item on the form since it is not possible to have both the item and subitems on the form at the same time. In this scenario, if the interviewer chooses the drill down approach we copy the four one-digit codes chosen into the four-digit field. If they choose “Show All” we skip over the one-digit fields and have them fill in the four-digit field.

Index File

A third option is to have the interviewer type in the name of the occupation/industry and to use an index file that maps commonly used occupation/industry names to the appropriate codes. The index file could come from an Excel spreadsheet like the following:

ISCO	Description
2460	Abbess
2460	Abbot
1229	Academic, university: head of department or faculty
2310	Academic, university: lecturer
3439	Accessioner, library
2453	Accompanist
2411	Accountant
2411	Accountant, certified
2411	Accountant, chartered
2411	Accountant, management
2411	Accountant, tax

We can use Excel to CSPro to convert this file to a CSPro lookup file with Description as the id-item and the code as a regular variable. In order to make the lookup case-insensitive we will convert the descriptions to all upper case before generating the CSPro lookup file.

Add a new field to use as the text to lookup in the index called OMJISCO_LOOKUP to the dictionary and add it to the form before the field OMJISCO. To do the lookup itself we can use the CSPro function *selcase* which does a search in a lookup file and shows the results in a pop-up window so the interviewer can choose one of the search results.

```
PROC OMJISCO_LOOKUP
preproc
ask if MULTJB in 1:2 and DMWCHK <> 1;

postproc
// Lookup text description in index file and load corresponding case
if selcase(ISCO_INDEX_DICT, toupper(strip(OMJISCO_LOOKUP))) include(ISCO_88)
then
    OMJISCO = ISCO_88;
else
    reenter;
endif;
```

Now when the interviewer enters the text in this field we display all matching cases in the index file. If the user chooses one of the matches, that case is loaded in the external dictionary, as if it had been added via loadcase, so we can use the variables for that case to assign the code to OMJISCO.

The current solution is limited since it only searches from the start of the description. With a small change, we can use the pos function to have it match anywhere within the description.

```
// Lookup text description in index file and load corresponding case
if selcase(ISCO_INDEX_DICT, "") include(ISCO_88) where
pos(toupper(strip(OMJISCO_LOOKUP)),ENGLISH_TITLE) > 0 then
    OMJISCO = ISCO_88;
else
    reenter;
endif;
```

Exercises

Implement the solution of your choice (search, drill down, hybrid, index file) for the main job industry code **IMJMAC**.