Session 9: Menu Programs

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

- Use the command *execpff()* to launch one CSPro application from another.
- Use *filewrite()* to generate a pff file from logic.
- Create a simple menu program to launch and sync the main data entry program.

Launching one CSPro application from another

We have seen how to use *execsystem()* to launch external programs from inside a data entry application. We can also launch other CSPro programs from within our data entry program using *execpff()*. You give *execpff* the path to a pff for a CSPro application and it will start it. It takes the same options as *execsystem (wait, stop…)*;

Let's create a simple launcher application that will start our survey application. Create a new folder named Menu in the same parent folder as the survey and create a new app named Menu in it. Using the default dictionary and forms, add the following to the postproc of MENU_ID.

```
execpff("../Survey/Survey.pff", wait);
reenter;
```

This will start our survey data entry application and wait for us to close it before continuing. The ".." in the path to Survey.pff tells CSEntry to go one directory up, i.e. to go to the parent of the Menu directory and from there to Survey/Survey.pff.

We can modify the pff file so that it automatically starts in add mode. Right click on Survey.pff and choose "Edit" to bring up the pff file editor. From there we can choose the start mode, name of the data file, external files and a host of other parameters.

Making the Survey Return to the Menu

Normally if we start a CSEntry application in add mode and complete a questionnaire, CSEntry will automatically begin entry of the next questionnaire. When starting from the menu program we do not want this behavior. Instead we want to return to the menu program as soon as we finish the questionnaire. We can do this by adding a call to *stop()*. Which proc can we put this in so that it will be called when we are done with a questionnaire? The postproc of the level is that last proc to be called for each questionnaire so we can put it there.

```
PROC SURVEY_QUEST

// If launched by menu we should stop and return to menu
// here after questionnaire is completed. If not, in add mode
```

```
// we will start to add another household instead of going
// back to the menu.
stop(1);
```

On Windows, once we exit via the call to *stop()* we will return to the already running menu program. However on Android, only one CSEntry instance can run at a time so if we have multiple .pff files in our CSEntry device on Android we will return to the main screen that lists all the CSEntry applications. However if there is only one pff file on the device, the corresponding application will be launched automatically.

Writing the pff file from logic

Launching pff files from logic really only gets interesting when we can control the parameters in the pff file from our menu application. In order to do this we need to write out the pff file from logic and then call *execpff*() on the file that we wrote out. To write out a pff file we use *setfile*(), *filewrite*() and *close*() just like we did when writing out reports. To simplify this we can use the "Expert Mode" of the pff editor which will give us the code we need to write out the pff. Here is the code that it gives us for our Survey.pff:

```
file pffFile;

function CreateAndRunPFF(string pffFilename)

    setfile(pffFile,pffFilename,create);

    filewrite(pffFile,"[Run Information]");
    filewrite(pffFile,"Version=CSPro 6.1");
    filewrite(pffFile,"AppType=Entry");

    filewrite(pffFile,"[DataEntryInit]");
    filewrite(pffFile,"StartMode=Add");

    filewrite(pffFile,"[Files]");
    filewrite(pffFile,"Application=%s","./Survey.ent");
    filewrite(pffFile,"InputData=%s","./Survey.dat");

    filewrite(pffFile,"[ExternalFiles]");
    filewrite(pffFile,"VILLAGESPERDISTRICT_DICT=%s","./VillagesPerDistrict.da
t");
    filewrite(pffFile,"VILLAGES_DICT=%s","./villages.dat");
    filewrite(pffFile,"HOUSEHOLDGPS_DICT=%s","./HouseholdGPS.dat");
    filewrite(pffFile,"ASSETLIMITS_DICT=%s","./AssetLimits.dat");
    filewrite(pffFile,"HOUSEHOLDSINVILLAGE_DICT=%s","./HouseholdsInVillage.da
t");

    filewrite(pffFile,"[UserFiles]");
    filewrite(pffFile,"TEMPFILE=%s","");

    close(pffFile);

    execpff(filename(pffFile));
```

```
end;
```

Unfortunately this code only works if you write the .pff into the same directory as the existing Survey.pff i.e. in the Survey folder. This is because the paths to the application and data file are relative. In order to be able to run this from anywhere we need to make them absolute. To do this we get the path of the menu application (using the *pathname* function) and append "../Survey" to get the absolute path of the Survey application.

```
function CreateAndRunPFF()

    // Generate a pff file for the data entry program
    // and execute it. See "Run Production Data Entry"
    // in the online help for more information on the format
    // of the pff file.

    // Get paths to survey application relative
    // the current application path.
    string surveyApplicationPath =
concat(pathname(Application),"../Survey/");

    // Put the generated pff file in the temp directory
    // so it doesn't show up on the list of applications on
    // Android.
    string pffFilename = concat(pathname(Temp), "Survey.pff");

    setfile(pffFile,pffFilename,create);

    filewrite(pffFile,"[Run Information]");
    filewrite(pffFile,"Version=CSPro 6.1");
    filewrite(pffFile,"AppType=Entry");

    filewrite(pffFile,"[DataEntryInit]");
    filewrite(pffFile,"StartMode=Add");

    filewrite(pffFile,"[Files]");
    filewrite(pffFile,"Application=%sSurvey.ent", surveyApplicationPath);
    filewrite(pffFile,"InputData=%sSurvey.dat", surveyApplicationPath);

    filewrite(pffFile,"[ExternalFiles]");
    filewrite(pffFile,"VILLAGESPERDISTRICT_DICT=%sVillagesPerDistrict.dat",
surveyApplicationPath);
    filewrite(pffFile,"VILLAGES_DICT=%svillages.dat", surveyApplicationPath);
    filewrite(pffFile,"HOUSEHOLDGPS_DICT=%sHouseholdGPS.dat",
surveyApplicationPath);
    filewrite(pffFile,"ASSETLIMITS_DICT=%sAssetLimits.dat",
surveyApplicationPath);
    filewrite(pffFile,"HOUSEHOLDSINVILLAGE_DICT=%sHouseholdsInVillage.dat",
surveyApplicationPath);

    filewrite(pffFile,"[UserFiles]");
    filewrite(pffFile,"TEMPFILE=%s","");
```

```
    close(pffFile);

    execpff(filename(pffFile), wait);

end;
```

Now that we are writing out the pff file we can customize the name of the data file based on the id-items. This will simplify data management and synchronization.

To do this let's add fields for village and district to the dictionary and to the form for our menu program. Then modify the function to generate the filename of the data file from the district and village codes:

```
    filewrite(pffFile,"InputData=%sSurvey%d%02d.dat",
              surveyApplicationPath, DISTRICT, VILLAGE);
```

Then we move the call to the function that launches the pff to the postproc of the village field so that it doesn't get called until after both district and village are entered.

```
PROC VILLAGE

CreateAndRunPFF();
reenter DISTRICT;
```

Now that we are filling in the id items in the menu program it would be nice if we could fill them in automatically in the survey application itself. To do that we need to pass them as parameters in the pff file. We can do this by adding a parameters section at the end of the pff file:

```
    filewrite(pffFile,"[Parameters]");

    // This passes the district ids as parameters to the survey
    // application which can access them via the sysparm() function.
    filewrite(pffFile,"District=%d", DISTRICT);
    filewrite(pffFile,"Village=%d", VILLAGE);
```

Then in the survey application itself we need to read these parameters using the function *sysparm()*.

```
PROC DISTRICT

preproc
// Fill in district if value was passed as
// parameter in pff (from menu program)
if sysparm("DISTRICT") <> "" then
    DISTRICT = tonumber(sysparm("DISTRICT"));
    noinput;
endif;
```

```
PROC VILLAGE

preproc
// Fill in village if value was passed as
// parameter in pff (from menu program)
if sysparm("VILLAGE") <> "" then
    VILLAGE = tonumber(sysparm("VILLAGE"));
    noinput;
endif;
```

Adding a menu

We can add a simple menu to our application by adding a new field named MENU to our form with a value set containing the menu options. For starters let's have just two choices: 1 – Enter new household, 9 – Exit. Add this field to the form, before the DISTRICT and VILLAGE fields and then add the following code to its postproc.

```
PROC MENU

// Handle menu choice
if MENU = 1 then
    // Add new household.
    // Do nothing, fall through to DISTRICT and VILLAGE
    // which will launch the application and
    // eventually return here via reenter.
elseif MENU = 9 then
    // Quit
    stop(1);
endif;
```

Next we need to reenter the menu once the execpff returns.

```
PROC VILLAGE

CreateAndRunPFF();
reenter MENU;
```

Deploying multiple applications on Android

When you copy your application to Android it is important that you preserve the same folder structure as you have on the PC. In our case we must create a separate menu folder to contain the .pen and .pff files for the menu and this menu folder must be in the same parent directory as the Survey folder so that when we append "../Survey/ to the menu application path it points to the folder containing the survey application we are launching.

One way to simplify deployment is to use a Windows batch file to generate the .pen files for both the survey and the menu and copy them into a deployment folder that can be copied onto the device. This can greatly speed up testing of your application on Android.

```
setlocal


REM Find CSEntry.exe (path differs on 32 and 64 bit Windows)

SET CSEntry="%ProgramFiles(x86)%\CSPro 6.1\CSEntry.exe"
if exist %CSEntry% goto :gotcspro
SET CSEntry="%ProgramFiles%\CSPro 6.1\CSEntry.exe"
if exist %CSEntry% goto :gotcspro
echo "Can't find CSEntry version 6.1. Is it installed?"
goto :eof
:gotcspro


REM Create deployment directory
rmdir /q /s Deployment
mkdir Deployment
cd Deployment
mkdir Survey
mkdir Menu
cd..

REM Create .pen files
%CSEntry% /pen .\Menu\Menu.ent
%CSEntry% /pen .\Survey\Survey.ent

REM Copy applications to deployment
move /y .\Menu\Menu.pen .\Deployment\Menu
move /y .\Survey\Survey.pen .\Deployment/Survey

REM Copy .pff for Menu
copy /y .\Menu\Menu.pff .\Deployment\Menu\Menu.pff

REM Copy Lookup files
copy /y .\Survey\AssetLimits.dat .\Deployment\Survey\
copy /y .\Survey\HouseholdGPS.dat .\Deployment\Survey\
copy /y .\Survey\HouseholdsInVillage.dat .\Deployment\Survey\
copy /y .\Survey\villages.dat .\Deployment\Survey\
copy /y .\Survey\VillagesPerDistrict.dat .\Deployment\Survey\
```

Exercises

1. Add a new field in the menu program for the household number. Use the household number entered in the menu program as part of the filename of the data file used for the survey. If the user enters district 1 village 2 household 3 the name of the data file should be *Survey1-02-003.dat*.
2. Pass the household id captured in the menu program as a parameter in the pff file and in the survey application use that value, if it exists, to fill in the household id field automatically.

3. Add a third menu option to sync the data files to a folder in your Dropbox (you choose the folder). Note that now that the data files name is based on the id-items you can no longer say “Put=Survey.dat”. Maybe using wildcards (*) will help. Make sure to return to the menu field after the sync is complete.