Session 5: Menu Programs and Synchronization

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

- Understand the how to design the screens of a menu program.
- Use the command *execpff()* to launch one CSPro application from another.
- Create a simple menu program to launch the main data entry program.
- Use the synchronization options dialog to add synchronization with a CSPro web server to an application.
- Create advanced synchronizations using CSPro logic to synchronize data files and update applications in the field.
- Create peer to peer synchronizations between devices for situations where there is no internet access.

What is a Menu Program?

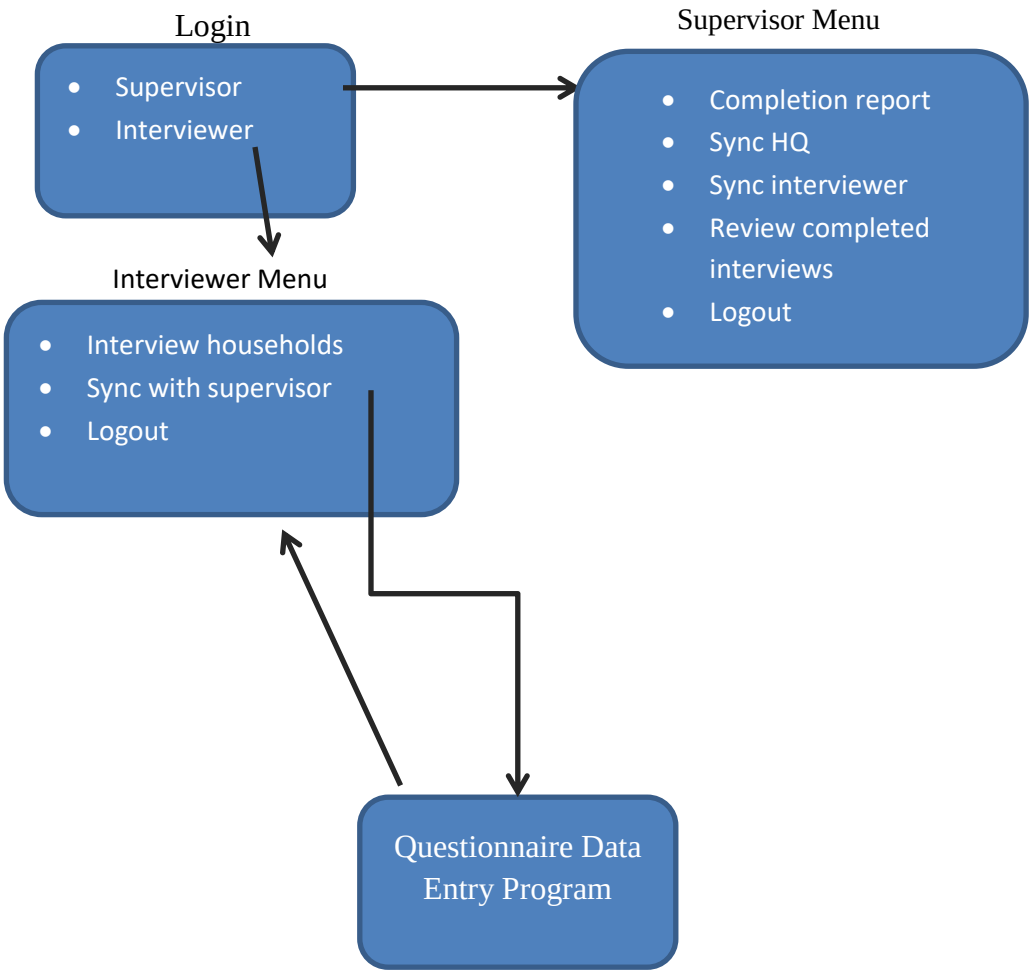
A menu program is a CSPro data entry application that is used to manage the data entry workflow. A menu program is not used for capturing any interview data itself. Instead, it launches other data entry programs for interviews. Menu programs generally have some or all of following functions:

- Launches other CSPro applications to do data collection (often pre-filling id items)
- Show reports on progress, summary statistics
- Manage user access through userid/passwords
- Manage interview assignments
- Launch synchronization

Designing the Screens and Flow

Once you decide on which functions you want your menu program to perform, the next step is to design the screens and how they link together. This is most easily done as a diagram like the one below for a fairly simple menu program.

S



Creating the Dictionary and Forms

Once you have the screens, the next step is to create a new data entry application for your menu program. Let's name ours *Menu* and put it in the folder *PopstanLFS/Menu*.
Each different menu screen will be a variable in the dictionary. The value set for the variable will show the available menu choices to the enumerator on that screen. In the menu example above we will have three variables:

- LOGIN (value set: 1 – Interviewer, 2 – Supervisor)
- INTERVIEWER_MAIN_MENU (value set: Interview Households - 1, Sync with Supervisor – 2, Logout -9)
- SUPERVISOR_MAIN_MENU (value set: Completion report – 1, Sync HQ – 2, Sync Interviewer – 3, Review Completed questionnaires – 4, Logout – 9).

We will keep the default id-item that CSPro creates for us. The menu program dictionary doesn't really need an id-item since we do not need to save our menu choices to the data file. However, CSPro requires that we have at least one id-item so we will keep it.

Create the items in the dictionary and then create a form and drop the items onto the form. Do not drop the id-item onto the form. Unlike typical data entry applications, menu programs tend not to have a linear flow. As a result, the order of the variables on the form is less important. We will use skips and reenters to move from one menu to another.

Menu Program Logic

The logic for processing the menu choice for each screen goes in the postproc of the variable for the menu. For example, to process the login menu field in our example we would have the following logic:

```
PROC LOGIN

// Go to the appropriate menu for the role chosen
if $ = 1 then
    skip to INTERVIEWER_MAIN_MENU;
else
    skip to SUPERVISOR_MAIN_MENU;
endif;
```

If the user selects to login as an interviewer we skip to the interviewer main field to show the interviewer menu, otherwise we skip to the supervisor main menu field to show the supervisor menu.

Handling the interviewer menu is similar. For now, we will leave the menu actions blank and fill them in later.

```
PROC INTERVIEWER_MAIN_MENU
postproc

// Handle the menu choice
if $ = 1 then
    // Main questionnaire
elseif $ = 2 then
    // Sync with supervisor
elseif $ = 9 then
    // Logout
    stop(1);
endif;

// Show interviewer menu again
reenter;
```

It is important to make sure that after the postproc of the menu field we do not let CSEntry continue to the next field, otherwise after the interviewer launches the household listing they would end up in the next field, which, in this case is the supervisor menu. To prevent this, we put a reenter at the end of the postproc so that we go back into the same menu field again.

It looks kind of strange that when we go back into a menu, the previous choice is still selected. We can prevent this by clearing the choice in the postproc of the field before the reenter.

```
// Clear previous entry
$ = notappl;

// Show interviewer menu again
reenter;
```

The supervisor menu is similar to the interviewer menu:

```
PROC SUPERVISOR_MAIN_MENU

// Handle the menu choice
if $ = 1 then
    // Report
elseif $ = 2 then
    // Sync HQ
elseif $ = 3 then
    // Sync interviewer
elseif $ = 4 then
    // Main questionnaire
elseif $ = 9 then
    // Logout
    stop(1);
endif;

// Clear previous entry
$ = notappl;

// Show supervisor menu again
reenter;
```

Launching one CSPro Application from Another

Let's fill in the function to launch the household data entry program. We can launch other CSPro programs from within our data entry program using ***execpff()***. You give ***execpff*** the path to a pff file for a CSPro application and it will start the data entry using the parameters in the pff file. In addition to the filename, you can also pass ***execpff*** either the keyword ***wait*** or the keyword ***stop***. Using wait will pause the menu application until the application that was launched exits, while using stop will exit the menu application immediately after launching the other application. On Android, we will always use the stop parameter since it is not possible to run two CSEntry applications at the same time on Android.

The following logic will launch a data entry application using the pff file PopstanLFS.pff in the sibling directory to the menu program directory.

```
if $ = 1 then
    // Main questionnaire
    execpff("../Questionnaire/PopstanLFS.pff", stop);
…
```

This will start the data entry application and immediately exit the menu. The ".." in the path to PopstanLFS.pff tells CSEntry to go one directory up, i.e. to go to the parent of the Menu directory and from there to Questionnaire/PopstanLFS.pff.

Making the Entry Program Return to the Menu Program

When we exit the main questionnaire, we want to return to the menu but the menu program stopped when we launched the main questionnaire. To get it to restart we can add a parameter to the pff file to tell it start the menu program when it exits. Right click on the PopstanLFS.pff and choose "Edit". This will bring up the pff file editor. You can then use the options menu to Add a new On Exit Pff entry. Use the browse button in that entry to locate the Menu.pff and fill in the path to it.

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 Questionnaire folder so that when we use "../Questionnaire/" from the menu application it points to the folder containing the entry application we are launching.

One way to simplify deployment is to use a Windows batch file to generate the pen files for both the entry application and the menu application. Copy both applications 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 7.1\CSEntry.exe"
if exist %CSEntry% goto :gotcspro
SET CSEntry="%ProgramFiles%\CSPro 7.1\CSEntry.exe"
if exist %CSEntry% goto :gotcspro
echo "Can't find CSEntry version 7.1. Is it installed?"
goto :eof
:gotcspro


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

REM Create .pen files
cd Menu
%CSEntry% /pen %CD%\Menu.ent
cd ..\Questionnaire
%CSEntry% /pen %CD%\PopstanLFS.ent
cd ..

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

REM Copy .pff files
copy /y .\Menu\Menu.pff .\Deployment\PopstanLFS\Menu
copy /y .\Questionnaire\PopstanLFS.pff
.\Deployment\PopstanLFS\Questionnaire
```

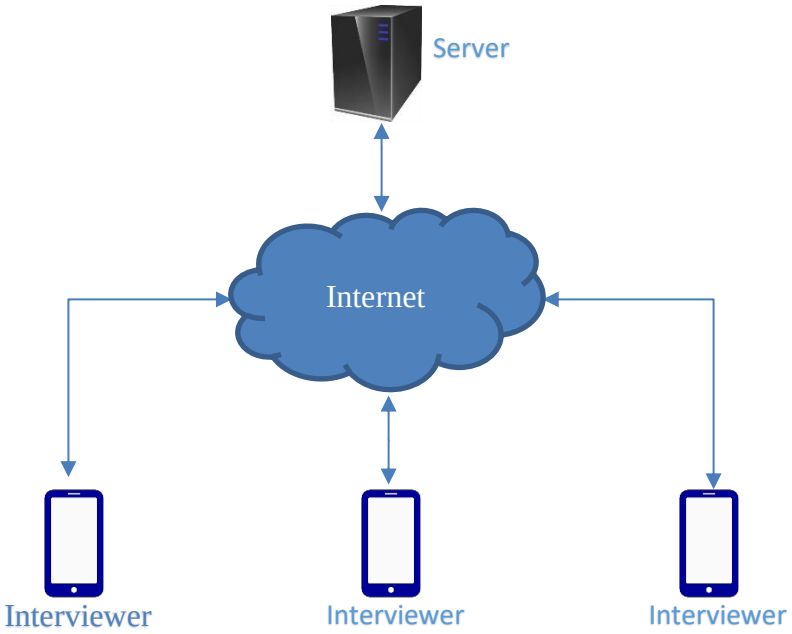
Tidying up the Menu Program

Currently the menu program shows up in the applications list on Android as “Menu” instead of something like “Popstan Labor Force Survey 2017”. In addition, when we tap on Menu we have to then tap “Start New Case” which doesn’t make sense for a menu. We can fix both of these

problems by modifying the pff file for the menu. Right click on the Menu.pff and choose "Edit with pff editor". Change "Start mode" to "Add" so that we won't have to tap "Start New Case". Enter "Popstan Labor Force Survey 2017" for the description to change what shows in the case listing. Finally, open the Menu program in the CSPro designer and in data entry options turn off the display of the case tree since the case tree is not useful for menu programs.

Synchronization in CSPro

After collecting data in the field, you need to get the data from the interviewer's devices back to headquarters to create a combined data file. While this can be done manually by connecting each device to a laptop, copying the individual data files and combining them using the Concatenate Data tool, this is rather inconvenient for large surveys. Instead, it is possible to send the data from each tablet over the internet to a central server that will combine the data into a single data file to be used for further processing. In CSPro this is called *synchronization*.



Direct synchronization between interviewers and central server over the internet

The Sync Server

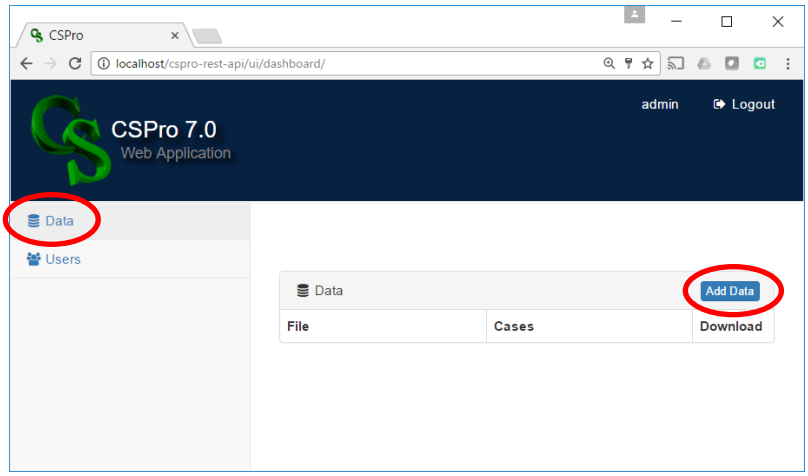
In order to use internet synchronization in CSPro you need a server accessible over a network. CSPro supports three types of Synchronization servers:

- CSWeb: A free web application that can run in a local data center or hosted on a server in the cloud. Best for large surveys. Requires skills in setting up and administering a website and SQL database.

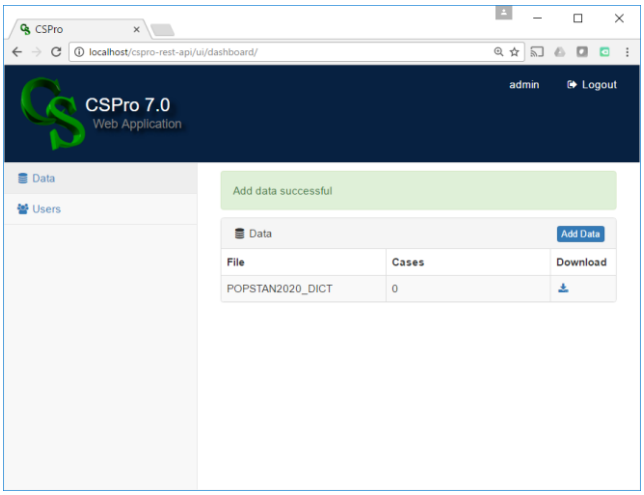
- Dropbox: A free service for synchronization with servers in the cloud using an account created at www.Dropbox.com. Data is stored on the servers managed by Dropbox in the United States. Best for small surveys when the skills and infrastructure for setting up CSWeb are not available.
- FTP: CSPro can synchronize with any FTP (File Transfer Protocol) server accessible over the network. Best for small surveys when the skills and infrastructure for setting up CSWeb are not available and you don't want your data on the Dropbox servers.

Using the CSPro Web Application interface

Before using the CSWeb server for synchronization we need to upload the data dictionary used by the application so the CSWeb can create a database table to store the data in. This step is not needed if you are using FTP or Dropbox as the server. With those servers, the dictionary is uploaded on the first sync. To upload the diction, first log into the CSWeb server in a web browser.



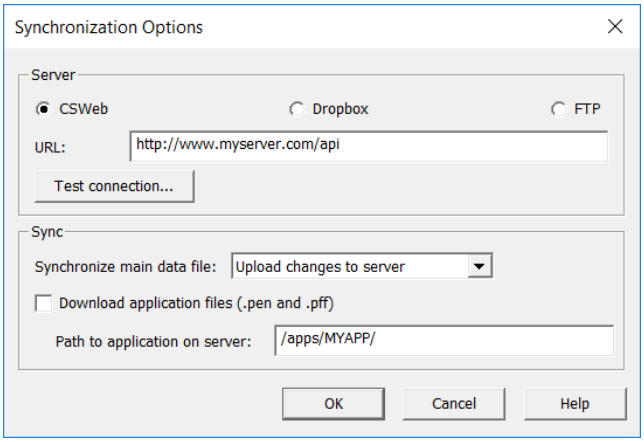
Next, choose "Data" from the left menu, click the "Add Data" button and browse to the dictionary to upload.



You should now see the dictionary listed in the table under Data. The server is now ready for synchronization.

Adding Synchronization to the Data Entry Application

In order to use synchronization, you must first add the server information to the application to enable synchronization. In CSPro choose "Synchronization…" from the options menu. Choose the server type (CSWeb, Dropbox or FTP). For CSWeb or FTP, enter the server URL (for Dropbox no URL is required). You can use the "Test Connection…" button to verify that the URL is correct.



In the dropdown, you have three choices for how to synchronize the main data file:

- Upload changes to server: all new cases and any cases modified locally will be uploaded to the server.
- Download changes from server: all new cases and cases modified on server will be downloaded from server.
- Sync local and remote changes: combines the first two options by sending updates to the server and downloading updates from the server.

In most cases you will want to choose either the first or third option. Use the third option if you want to share cases between multiple devices in the field as this will download every case on the server onto the local device. For a survey operation with a large number of interviewers you may want to avoid this as it will cause you to download a lot of data. For our test, we will use the first option.

In addition to synchronizing the data file you can also download the pen and pff files from the server. This provides a way to do remote updates to the application while interviewers are in the field. In order for this to work you must place a copy of the pen and pff files on the server. For CSWeb these must be placed in the csweb/files/ on the server. For Dropbox , they may be placed anywhere in the Dropbox folder. For FTP they must be placed in the home directory of the user that will be used to log into the server. Enter the path on the server where the files were placed under "Path to application on server". We will place these files in the directory /PopstanLFS on the server.

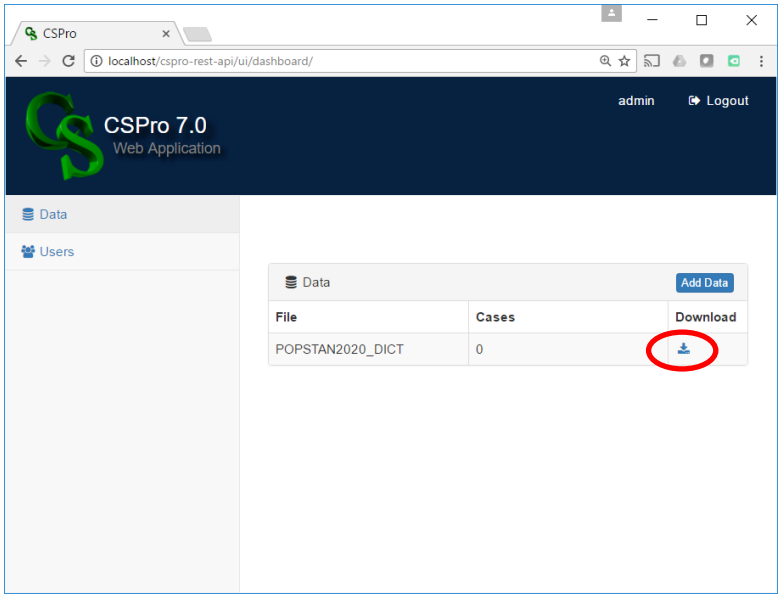
Once the sync settings are complete, regenerate the pen file, since the sync settings are stored in the pen file. Copy the pen and pff file into the PopstanLFS directory under the files directory on the server so that it will be downloaded to the local device during sync.

Running Sync

To synchronize the application with the server, run the application in CSPro and from the case listing screen choose “Synchronize…” from the file menu on Windows or tap the synchronize icon on Android.

Downloading the Combined Data File

Once you have used synchronize to upload data to the server you can download the combined data file from the web interface by clicking on the download icon on the Data page.



When you open the downloaded file, it will automatically open DataViewer and begin downloading the combined data. Once you have downloaded the data file the first time, you can use the “synchronize” function in DataViewer to only download the newly updated data rather downloading the entire file a second time.

Group exercise – Sync Race

Split into groups of 3-4 people and create a small CSPro application for the following questionnaire:

Synchronization Exercise Questionnaire

A) Team name: _____

B) Names of team members:

1	
2	
3	
4	

Upload your data dictionary to the following server:

URL: http://csweb.teleyah.com/ui
Admin username: admin
Admin password: adminadmin

Add synchronization to your application:

URL: http://csweb.teleyah.com/api
Username: test
Password: password
Choose to upload data to the server.
Do not sync the application files (pen and pff).

Run your application, complete a case and synchronize to send your data to the server.
The first team to get their data uploaded to the server is the winner.

Synchronization from Logic

Sometimes we need more control over the synchronization than the sync options dialog provides. For example, if there are external dictionaries in an application, if you want to sync with a universe or if there are multiple applications to sync at once. CSPro provides logic functions to run synchronizations from within an application that allow more complex synchronizations. Let's add synchronization to our menu application that will upload the main data file and download the latest version of the pen and pff files for both the menu application and for the main questionnaire. First fill in the logic for the menu item "Synchronize with Headquarters". The first step in running sync from logic is to call the command *syncconnect*() and pass it the server information. If this function succeeds then we are connected to the server and we can make calls to the commands *syncdata*() and *syncfile*() to synchronize data files and non-data files respectively. Finally we call *syncdisconnect*() to end the session.

The commands *syncdata*() and *syncfile*() operate differently. *Syncdata*() may only be used with data files in CSPro DB format. With *syncdata*(), CSPro keeps track of which cases in the file have already been synced with the server so that it avoids transferring cases that have already been synced. This reduces the amount of data transferred and thus keeps data costs and transfer times to a minimum. It also avoids one interviewer overriding changes made by another interviewer when they are working on different cases in the same file. *Syncfile*() may be used with any type of file including text files, images and application files (pen and pff). It does not look at the file contents so it simply uploads or downloads the entire file, overwriting any existing version. When syncing data files, you should always use *syncdata*().

Syncdata() takes two arguments: the direction and the dictionary name. The direction can be PUT (upload data to server), GET (download data from server) or BOTH (upload and download). These options correspond to the drop down in the sync options dialog. The second argument is the name of a dictionary in the application that corresponds to the data file to synchronize. Note that this dictionary must be added to the application as either the main dictionary or as an external dictionary.

In our application, we want to synchronize the main questionnaire dictionary so we need to first add it to the menu program as an external dictionary. Now we can use it in the call to *syncdata*().

We will also add calls to *syncfile*() to download the latest versions of the application programs from the server. This function takes the direction (PUT or GET), the "from" directory and the "to" directory. In the case of GET the "from" directory is the local directory on the device and the "to" directory is the path on the server.

```
    // Connect to server
    if syncconnect(web, "http://csweb.teleyah.com/api", "syncuser",
"password") = 1 then

        // Sync main data file.
        // Note that POPSTANLFS_DICT must be added as
        // an external dictionary to the menu program.
        // Use PUT to do a one way sync. Only cases added/modified
        // on the tablet are sent to the server. This way cases
        // added by other interviewers are not downloaded to the tablet.
        syncdata(PUT, POPSTANLFS_DICT);

        // Download latest application files from the server.
        // The files are in a directory named "PopstanLFS"
        // on the server.
        syncfile(GET, "/PopstanLFS/Menu/Menu.pen",
                 "Menu.pen");
        syncfile(GET, "/PopstanLFS/Menu/Menu.pff",
                 "Menu.pff");

        // Since the current application is in the Menu folder we need
        // to use "../Questionnaire" to go up one level and back down into
        // Questionnaire folder for the household application files.
        syncfile(GET, "/PopstanLFS/Questionnaire/PopstanLFS.pen",
                 "../Questionnaire/PopstanLFS.pen");

        syncdisconnect();

    endif;
```

Now that we have added the synchronization logic to the menu we rebuild the pen files and copy them to the files directory on the server.

Now we can run the menu program and test our synchronization.

Security Considerations

It is important to ensure the security of your server and data. Computer and network security is a challenging problem and well beyond the scope of this training. Here a few security considerations to think about. For any large survey or census operation you should consult with a security expert to ensure that your data is safe.

Use HTTPS on the web server for data transfer

This encrypts the data transmission that goes over the internet. It requires purchasing and installing an SSL certificate on the server. Without SSL, server passwords are sent unencrypted and are vulnerable to hackers.

Use device encryption on Android devices

By setting a PIN code on your Android tablet or phone you enable encryption of files on the device using strong hardware encryption. This makes it very difficult for anyone without the PIN code to retrieve data on the device.

Consider how to manage passwords

It is more convenient to have a single password that is shared by all devices and is hardcoded in the pen file however it is more secure to use a different username/password for each interviewer and have them enter it for every sync. For a large number of enumerators managing passwords and resetting passwords for interviewers who have forgotten their passwords could be a significant management burden. You will need to find the right balance of security and convenience.

Synchronization with a Universe

The *syncdata*() command has an optional third parameter, the universe. This is a string that CSPro tries to match to the concatenated case-id items to limit the data transferred. If a universe is provided, then only cases who case-ids match the universe are synchronized. For example, to limit synchronization to only province 1, district 2 we would call:

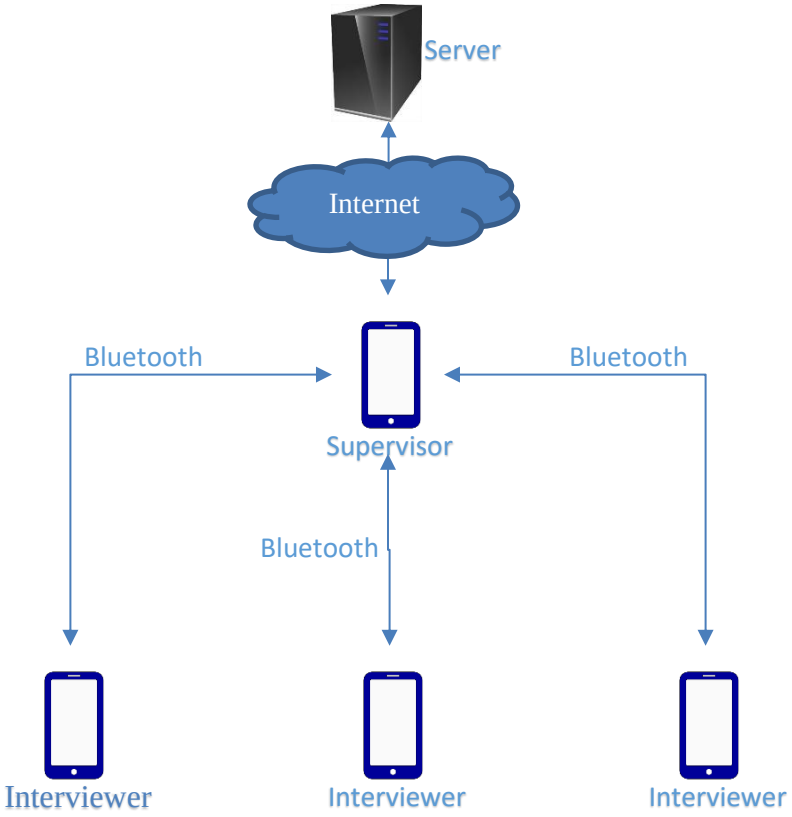
```
syncdata(BOTH, POPSTANLFS_DICT, "102");
```

This will only synchronize cases whose case-ids start with "102". Since province and district are "1" and "02" this limits the sync to cases in province 1 and district 2. Similarly, if we used "1" as the universe all cases in province 1 would be synced.

This is useful when there are a large number of interviewers and you don't want every interviewer to download the data from every other interviewer. You can specify the geographic area assigned to the interviewer as the universe so that only cases in the interviewer's area are synced.

Peer to Peer Synchronization Using Bluetooth

In some places, interviewers do not have reliable internet access and cannot synchronize directly with the server. In this case, you can use peer to peer to synchronization. Interviewers synchronize with a supervisor's tablet or laptop over Bluetooth. Later, the supervisor goes to a location where they can connect to the internet and they synchronize with the central server.



Synchronization with supervisor over Bluetooth when interviewers do not have

In order to implement Bluetooth synchronization, we need to add logic on both the supervisor device and on the interviewer device. One device acts like the web server, processing synchronization requests from the client device, and the other device, the client, sends syncfile and syncdata commands the same way that it would to a web server. For our example let's make the supervisor device be the server and the interviewer device be the client. This is arbitrary, we could easily reverse the roles.

To extend the menu program to support peer to peer synchronization we will implement the menu options: *sync with supervisor* and *sync with interviewer*.

The logic on the interviewer's tablet is nearly identical to the logic used for web synchronization. The only differences are the arguments to *syncconnect*(). When using Bluetooth, *synconnect*() does not need a URL, username or password.

```
syncconnect(bluetooth)
```

Syncconnect() will scan all nearby Bluetooth devices and present a list to the interviewer who can choose which device to connect to. Alternatively, if you know ahead of time the name of the

device to connect to you may specify it as a second parameter. In this case *syncconnect*() will try to connect directly to the named device.

```
syncconnect(Bluetooth, "Supervisor03")
```

The rest of the logic is the same as for web synchronization except that the “from” paths change to match the paths on the supervisor device.

```
    // Connect to the supervisor device
    // We do not specify the device name to connect to
    // which allows the interviewer to pick the device
    // from a list of nearby devices.
    if syncconnect(bluetooth) = 1 then

        // Sync main data file.
        syncdata(PUT, POPSTANLFS_DICT);

        // Download latest application files from the supervisor.
        // The root file on the supervisor tablet is the
        // "PopstanLFS" folder so the from
        // folder starts from there and we need to add "Menu"
        // to get to the Menu programs.
        syncfile(GET, "Menu/Menu.pen", "Menu.pen");
        syncfile(GET, "Menu/Menu.pff", "Menu.pff");

        // Since the current application is in the Menu folder we need
        // to use "../Questionnaire go up one level and back down into
        // Questionnaire folder for the main questionnaire application files.
        syncfile(GET, "Questionnaire/PopstanLFS.pen",
                 "../ Questionnaire /PopstanLFS.pen");

        syncdisconnect();
    endif;
```

The logic for the supervisor is even simpler. We simply call the command *syncserver*() which runs the Bluetooth server and waits for connections.

```
    // Run the Bluetooth server to receive data from
    // interviewer.
    // Specify ".." as the root directory to make the
    // the "PopstanLFS" folder be the starting
    // point for all file get/put commands instead of the
    // menu folder.
    syncserver(Bluetooth, "..");
```

Syncserver() takes an optional second parameter which is the root directory for all *syncfile*() commands. The root directory is prepended to the file path on the server. So if the root directory is “C:/myroot” and the client calls syncfile(PUT, “myfile”, “files/myfile”) then the destination

path will be “C:/myroot/files/myfile”. By default, the root directory is the application directory of the server so adding “..” to it makes the root folder the parent directory of the application i.e. the PopstanLFS directory.