OpenHDS manual

Aurelio Di Pasquale, Rajib Mitra, Nicolas Maire
Swiss TPH, Socinstr 57, PO Box, 4002 Basel, Switzerland

Contents

Introduction 4
Administrator manual 4
Installing Minimal Ubuntu server 4
Installing Java 7 4
Installing Tomcat 4
Installing MySQL-J Connector 5
Deploying war file to Tomcat 5
Set up OpenHDS data 5
Installing MySQL Server 5
Grant outside access to MySQL 5
Install Apache2, PHP and PHPMyAdmin 6
Installing SSH-Server 6
Installing Mirth 6
Installation steps 6
Set-up Mirth Connect to use the MySQL database 6
Installing ODKAggregate 7
Installer procedure 7
Installation 7
Deploy ODK Aggregate with a Local Superuser account 7
Upload DSS Core XLSForms 8
Create Database and Views for Data Management 9
Custom Location hierarchy 11
Tablet setup 12
User manual 13
Field worker manual 13
Getting Started 13
Field Worker Login 14

Creating New Location 16
Language Selection 18
Standard Operating Procedures (SOP) 19
Baseline Visit 19
Flowchart 19
Outline 20
Follow-up Visit 20
Find Existing Location 20
Filter Existing Location 21
Create Visit 21
In Migration 22
Out Migration 28
Membership 29
Relationship 30
Pregnancy Observation 31
Clear Individual 32
Pregnancy Outcome 33
Death 34
Finish Visit 36
Data manager manual 38
Extra forms 38
Migration of legacy data to openHDS 38
Requirements 39
File Conversion and staging DSS database 39
Required tables for migration 39
Adding required attributes 40
Common steps for all tables 40
Steps for each table 41
Round 41
Location 41
Observation (maps to Visit) 42
Individual 43
Socialgroup 45
Relationship 45
Membership 46

Pregnancy observation (indvstatus) 48
Pregnancy outcome (pregoutcome) 49
Birth 49
Residency 50
Death 51
Outmigration 52
Inmigration 52
Data import into OpenHDS 53
OpenHDS virtual server setup 55
OpenHDS virtual server setup 55
openHDS mobile 56
Test data generation 56
Setup 56
Installing dependencies 56
Run fieldworker-simulator 56

Introduction

OpenHDS is a multi-component data platform for Health and Demographic Surveillance, based on client-server architecture and using tablet computers for point-of-capture digitization of data. This document provides an overview of the system. It covers the setup and administration of the server and tablet components of the platform. It contains guides for field workers, field supervisors, and data managers. Finally, it describes the setup and use of a virtualized OpenHDS system for testing and evaluation purposes.

Administrator manual

This section contains the instructions on how to set-up the server and additional software which will result in a fully functional OpenHDS-Server platform with ODKAggregate and MirthConnect installed. OpenHDS will connect to a local MySQL database and run under the Tomcat servlet container. We assume that you make an installation to a local domain “local”, and the hostname “data-management” for the server, and that you can reach the server under this domain name. Adjust these settings to your local environment.

Installing Minimal Ubuntu server

- Download the **Ubuntu Mini** ISO from
  https://help.ubuntu.com/community/Installation/MinimalCD
  (32-bit PC (x86) Ubuntu 14.04 "Trusty Tahr" LTS Minimal CD, Size is around 31MB)
- Install the operating-system with no localization options and don't install any packages.
- Set the hostname of the machine to **data-management.local.**

Installing Java 7

- Currently the only supported Version is Java 7. Different auxiliary software packages required to run a full-blown OpenHDS server have different Java requirements, version 7 is the consensus.
- Install Java with command: ‘sudo apt-get install openjdk-7-jre-headless’
- Set the environment variable *JAVA_HOME* to the Java installation folder (you might need to reboot of the system for the variable to be available).

Installing Tomcat

- Install Tomcat (check available versions for distro on https://launchpad.net/ubuntu/+source/tomcat6) with 'sudo apt-get install tomcat6' .
- Check if Tomcat is running with 'sudo service tomcat6 status'You can start the Tomcat service with 'sudo service tomcat6 start'
- If the service is erroring out with a message about *JAVA_HOME* not set, open **/etc/default/tomcat6** with 'sudo vi /etc/default/tomcat6/' , uncomment the line 'JAVA_HOME=...' and set it to folder of Java installation.
- Install *tomcat6-admin* with 'sudo apt-get install tomcat6-admin'.
- Configure users which have access to the Tomcat Manager. They are defined in **/etc/tomcat6/tomcat-users.xml**. Add a user with the *html-manager* role and reload the Tomcat-service (‘sudo service tomcat6 restart’). You should now be able to log in into the tomcat manager under: http://serverip:tomcat_port/manager
- The Tomcat log-files are storend undern **/var/log/tomcat6/**

Installing MySQL-J Connector

MySQL Connector/J is the official JDBC driver for MySQL. It is required for Tomcat in order to communicate with a MySQL server instance.

- Go to http://dev.mysql.com/downloads/connector/j/ and download the mysql-connector*.tar.gz.
- unpack the file to Ubuntu home (~) and chmod the mysql*.jar file to 777 or just executable.

OR

Install package 'sudo apt-get install libmysql-java' which will put the MySQL connector into '/usr/share/java'.

- cd to **/usr/share/tomcat6/lib**
- Create a symbolic link 'sudo ln -s mysql-connector-xyz.jar .', resp. cd /usr/share/tomcat6/lib
- > sudo ln -s ../../java/mysql.jar mysql.jar
- Restart the Tomcat-service with 'sudo service tomcat6 restart' .

Deploying war file to Tomcat

A war file is a Web Application Archive that can be deployed in Tomcat. OpenHDS comes packaged in this format. Ways to obtain the OpenHDS Web Application:

- You can build and the OpenHDS server manually from Source code available under https://github.com/SwissTPH/openhds-server (Maven build) OR
- Download a precompiled OpenHDS .war-file from https://github.com/SwissTPH .

- Deploy the war file through the Tomcat html manager http://data-managment.local:8080/manager

Set up OpenHDS data

Execute the SQL-script 'openhds-required-data.sql' which can be found in the openhds.war-file (open with 7-zip or equivalent) under WEB-INF\classes\. This will insert some required data into the openhds-databse, including a default administrator login (admin/data). It is highly recommended to change this login for a production setup!.

Installing MySQL Server

- Install the MySQL server with 'sudo apt-get install --no-install-recommends mysql-server'.
- Follow the set-up instructions during installation and create a root user during installation.

Grant outside access to MySQL

- Edit /etc/mysql/my.cnf and comment out line starting with *Bind-Address*
- Restart the MySQL-service with 'sudo service mysql restart'
- Login to MySQL-cli with: 'mysql -u root –p'
- Grant privileges with: 'GRANT ALL PRIVILEGES ON *.* TO 'data'@'%' WITH GRANT OPTION;'

Install Apache2, PHP and PHPMyAdmin

- Install Apache2 with 'sudo apt-get install apache2'.
- Install php for Apache with 'sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt'
- Install PHPMyAdmin with 'sudo apt-get install phpmyadmin'
- Select apache2 during installation.
- Select yes to dbconfig-common.
- Navigate to /etc/apache2 and open apache2.conf and make sure following line is available at the bottom of the file 'Include /etc/phpmyadmin/apache.conf'. If not, paste it into the file, save and restart the apache2 webservice with 'sudo service apache2 restart'.
- Direct your web browser to http://data-managment.local/ . A webpage should show up that displays "It works!" .
- PHPMyAdmin is now accessible under http:// data-managment.local /phpmyadmin

Installing SSH-Server

The SSH server installation is required to remotely access the server.

- Install ssh-server with: 'sudo apt-get install -y --no-install-recommends openssh-server'
- Service option: sudo service ssh status | start | stop | restart

Installing Mirth

- Download the Linux installer from http://www.mirthcorp.com/community/downloads (~116MB).
- Issue a 'chmod a+x mirth....sh' to mark the installer as executable.
- Run the installer with 'sudo ./mirth...sh' .
- Read through the agreements, scroll with enter and set-up as follows:

Installation steps

- Default install to **/usr/local/mirthconnect**
- Symlinks go to **/usr/local/bin**
- Set port number, e.g.port 8082 (since port 8080 is in use by tomcat).
- Set apps and log directory (/usr/local/mirthconnect/apps and /logs resp.)
- After installation, check if connection to http://data-management.local:8082 is okay.
- Launch mirth connect administrator and log in with admin/admin
- Create an admin account (admin/test) - no need to register.
- start Mirth Connect server by running 'sudo service mcservice start' (service is located in **/usr/local/mirthconnect**).

Set-up Mirth Connect to use the MySQL database

By default, Mirth will use the Derby database. We will need to change this to MySQL. First create a Mirth database user in PHPMyAdmin, e.g. with following script:

CREATE DATABASE mirthdb DEFAULT CHARACTER SET utf8;

GRANT ALL ON mirthdb.* TO mirthuser@'localhost' IDENTIFIED BY 'demodemo' WITH GRANT OPTION;
GRANT ALL ON mirthdb.* TO mirthuser@'%' IDENTIFIED BY 'demodemo' WITH GRANT OPTION;

Now open the file 'mirth.properties' in /opt/mirthconnect/conf for editing. Change the line starting with ‘database’ to ‘mysql’, change ‘database.url’ to 'jdbc:mysql://localhost:3306/mirthdb' (or your database name), and set the values for ‘database.username’ and ’database.password’. Afterwards restart the mirth connect service and and upon restart, the new database schema will be created.

Installing ODKAggregate

See also http://opendatakit.org/use/aggregate/tomcat-install/

- Download the Linux installer from http://opendatakit.org/downloads/ (~260MB)
- Change installer permissions with 'chmod 777 ODK...run'
- Run installer with './ODK...run'

Installer procedure

- Put file into: *~/*
- sql user: *odk_user* / *odk_password* (not recommended for production)
- Db name: *odk_prod*
- ODK Aggregate instance name: *odk*
- Enter valid gmail account
- Configuration done!

Installation

The installer ran in the previous step will create a new folder called ‘ODK Aggregate’ in the location you supplied in the previous step.

- Deploy the *ODKAggregate.war* with the Tomcat manager.
- Start the ODK Aggregate application in Tomcat manager if it is not already started.
- Navigate to 'http://data-management.local:8080/ODKAggregate/' in browser to browse to ODKAggregate

Note: If the installer quits the installation with the error: "Installer payload initialization failed. This is likely due to an incomplete or corrupt downloaded file." then try re-downloading the file again with *wget*. (*wget* for windows available at http://www.alexlomas.com/blog/2005/08/64_bit_wget_for_windows/)

Deploy ODK Aggregate with a Local Superuser account

Skip the following steps unless you are required to use an old version of ODKAggregate (from ODKAggregate v1.4.4. on and upwards, the installer will ask for a ODK Aggregate username for the super-user. The default password is aggregate. The following steps are thus not required anymore.)

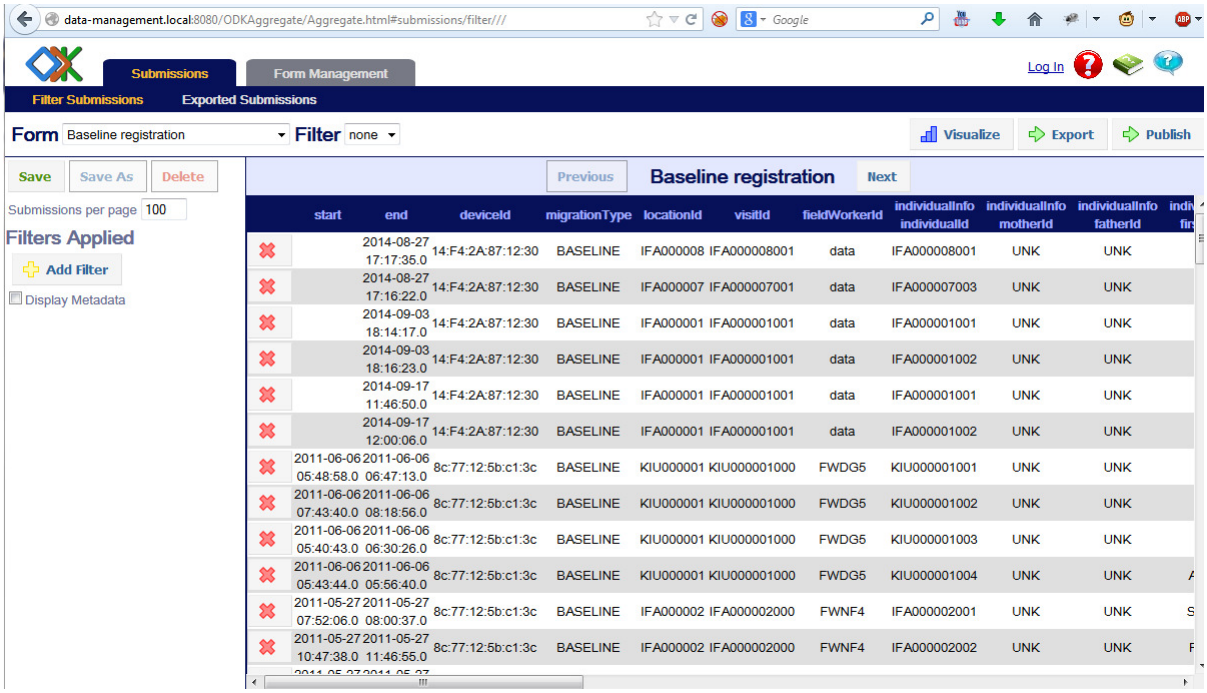
ODK Aggregate v.1.4.3 and lower by default require a Google e-mail account to set-up the installation. The steps provided here will create a local super-user so that no google mail account is required. This is especially helpful if you don't have/want a google account for the initial login or don't have an internet connection available:

- Open/extract the *ODKAggregate.war* that was created after executing the installer with e.g. 7zip.
- Navigate to the **WEB-INF\lib** folder
- Open *'ODKAggregate-settings.jar' (also with a zip program)*
- Open the file **security-properties** in a text editor
- Add admin username next to 'security.server.superUserUsername=' (e.g. *odk_admin* / *aggregate*)
- The default password is *aggregate*
- Repackage (zip up and put to according location) and redeploy the war file.
- Login over http://data-management.local:8080/ODKAggregate/

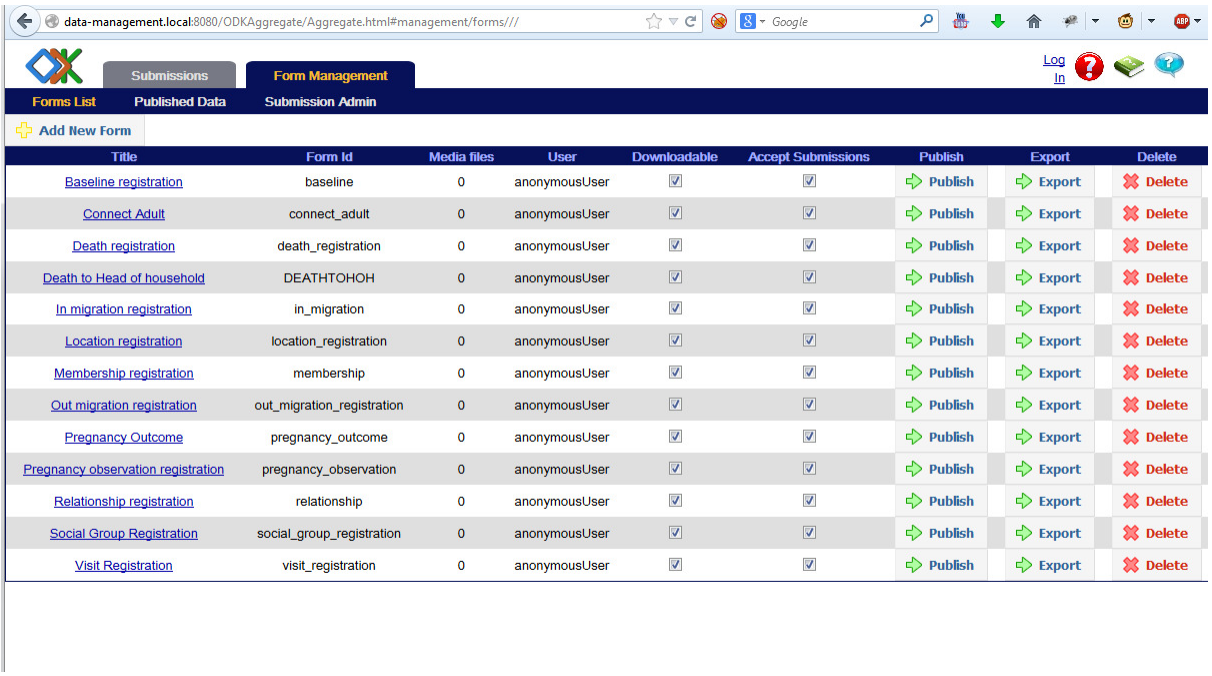
Upload DSS Core XLSForms

Call up http://data-management.local:8080/ODKAggregate/ in your Web browser.

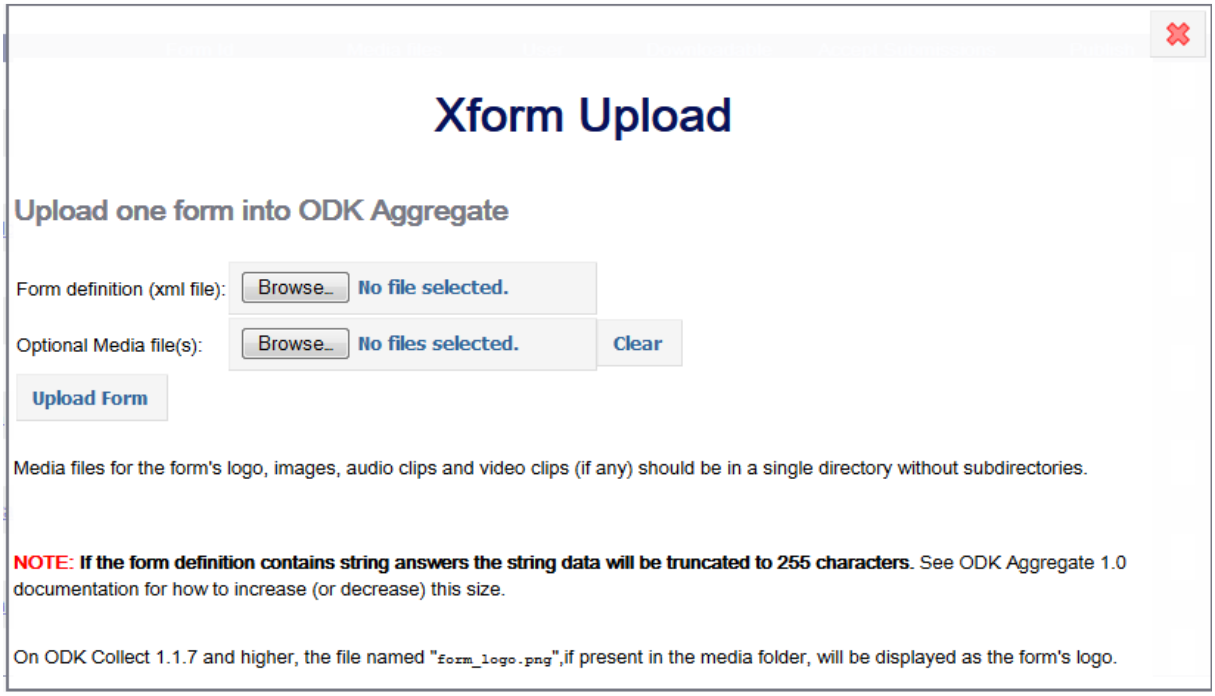
Log in with local account if required.



Switch to the 'Form Management' tab.



Click on add new form.



Click browse and select the XForm (.xml).

Click on 'Upload Form' to submit the form.

After a successful upload, the new form will appear in the 'Forms List'.

Create Database and Views for Data Management

This section provides SQL statements that will provide the Data Manager with a simplified database view of errors that occurred while sending data to OpenHDS. This allows for a fast and uncomplicated determination of cause of problematic data. These entries can then be corrected and resubmitted for a successful insertion.

After logging in as user **root** to the **odk** database, execute following SQL statements:

```
CREATE TABLE `errors` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `CHANNEL` varchar(30) DEFAULT NULL,
  `DATA` varchar(500) DEFAULT NULL,
  `ERROR` varchar(280) DEFAULT NULL,
  `exported` int(1) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=51322 DEFAULT CHARSET=latin1;
```

#UPDATE ROUND VIEWS

create view IN_MIGRATION_VIEW as select * from IN_MIGRATION_CORE where PROCESSED_BY_MIRTH=2;

create view LOCATION_VIEW as select * from LOCATION_REGISTRATION_CORE where PROCESSED_BY_MIRTH=2;create view DEATH_VIEW as select * from DEATH_REGISTRATION_CORE where PROCESSED_BY_MIRTH=2;

create view MEMBERSHIP_VIEW as select * from MEMBERSHIP_CORE where PROCESSED_BY_MIRTH=2;

create view OUT_MIGRATION_VIEW as select * from OUT_MIGRATION_REGISTRATION_CORE where PROCESSED_BY_MIRTH=2;

create view PREGNANCY_OBSERVATION_VIEW as select * from PREGNANCY_OBSERVATION_CORE where PROCESSED_BY_MIRTH=2;

create view PREGNANCY_OUTCOME_VIEW as select * from PREGNANCY_OUTCOME_CORE where PROCESSED_BY_MIRTH=2;

create view RELATIONSHIP_VIEW as select * from RELATIONSHIP_CORE where PROCESSED_BY_MIRTH=2;

create view SOCIALGROUP_VIEW as select * FROM SOCIAL_GROUP_REGISTRATION_CORE where PROCESSED_BY_MIRTH=2;

BASELINE ROUND VIEW

create view BASELINE_VIEW as select * from BASELINE_CORE where PROCESSED_BY_MIRTH=2;

```
CREATE USER 'datamanager'@'%' IDENTIFIED BY 'dataODKmanager';
```

GRANT SELECT, UPDATE ON DEATH_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON IN_MIGRATION_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON LOCATION_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON MEMBERSHIP_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON OUT_MIGRATION_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON PREGNANCY_OBSERVATION_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON PREGNANCY_OUTCOME_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON RELATIONSHIP_VIEW TO 'datamanager'@'%';

GRANT SELECT, UPDATE ON SOCIALGROUP_VIEW TO 'datamanager'@'%';

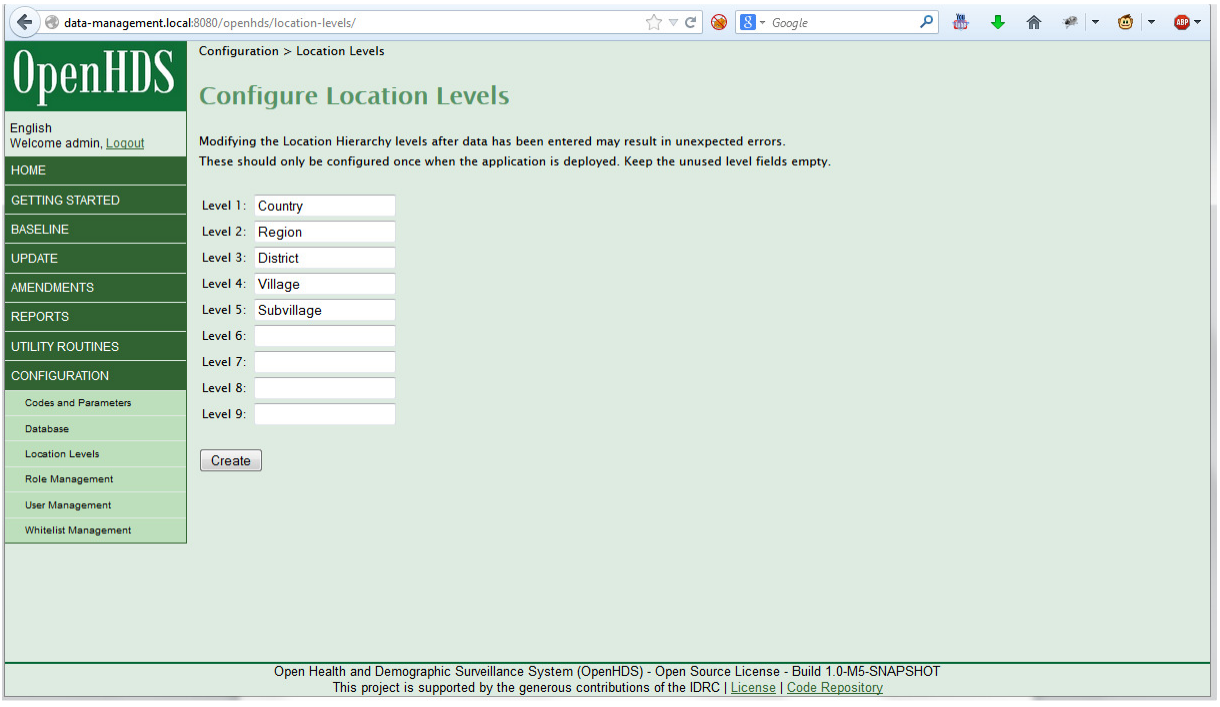
BASELINE ROUND PERMISSIONS

GRANT SELECT, UPDATE ON BASELINE_VIEW TO 'datamanager'@'%';

Custom Location hierarchy

By default openHDS has a location hierarchy that starts and the country level, goes down to region, District, Village and finally Subvillage. It is possible to modify these labels to have a custom location hierarchy where this is needed.

To customize the location hierarchy, open the openHDS web-application and log in.. Navigate to 'Configuration' and finally to 'Location Levels'.



Make the changes to the Labels and save with 'Create'.

You will need to stop and start the openhds-webapp in Tomcat for the changes to register.

IMPORTANT:

Once you have changed the location Levels, you'll also need to update the openHDS mobile application to handle these changes.
Open the strings.xml and find the entry 'STARTHIERARCHYLEVELNAME'. Change the value of the entry accordingly, e.g. from *Region* to *State* and safe the file.

Tablet setup

- Download the OpenHDS apk from https://github.com/SwissTPH/openhds .

-Installation of openhds mobile and ODK Collect (from the Google Play Store or from https://opendatakit.org/downloads/download-info/odk-collect-apk/).

-Configure server connection parameters and access credentials.

-Synchronize the applications with their server component.

User manual

Field worker manual

This section gives detailed descriptions on how a Field Worker will work with OpenHDS mobile application in the field.

Below are the procedures for OpenHDS mobile data collection using Android Tablet computers.

Getting Started

To get into the application you have to find and tap the icon which looks like Figure 1 below;

Figure 1

First for security purpose, Field worker will need to have login credentials to be able to access the application. Hence they will be encountered with the following page as shown in Figure 2 below:

A Field worker will always use the top button **“Log in as Field Worker”**.

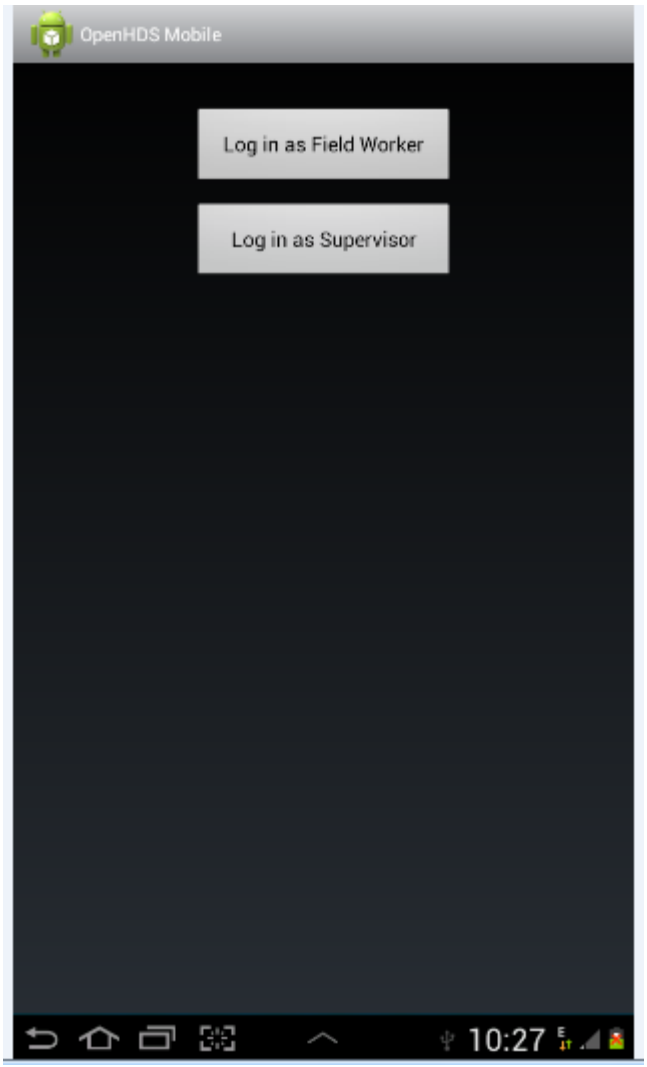


Figure 2

Field Worker Login

Below Figure 3 is the login page when pressing **"Log in as Field Worker".** For the first time using the device a Field worker should register their user name and password prior to log in. Therefore first s/he should enter Username and Password then tick the Register Field worker on device, and then once they successfully registered (Network is required for this procedure), thereafter they will need to login this time they will have to uncheck the Register Field worker on device

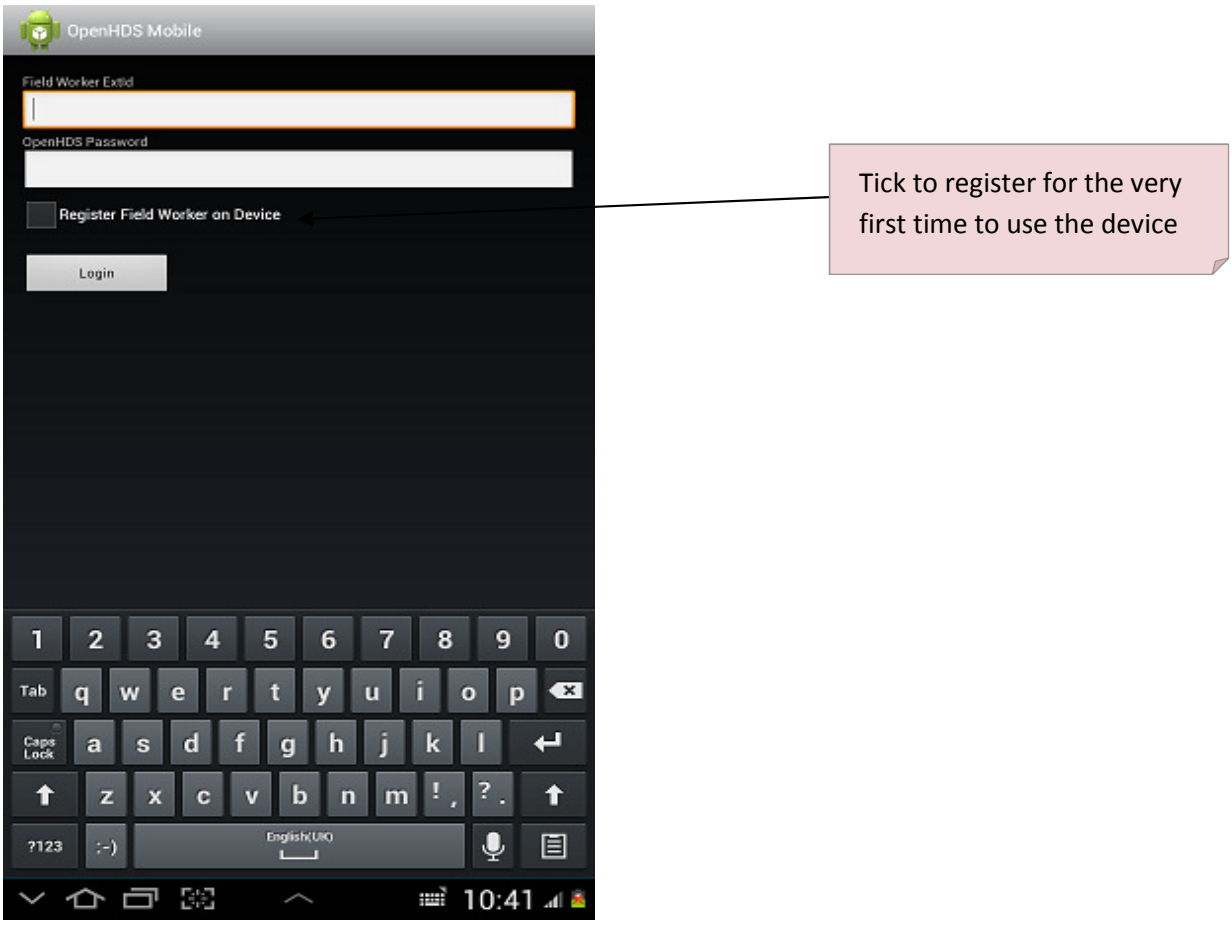


Figure 3

Below in Figure 4 shows the **main menu** display, on the left column is the information about location and individuals while on the right column is event information. Please note that most of the buttons are always disabled until it is right time for usage, for instance the **"Select Individual"** button cannot be active if the location information has not been select etc.

On the top right corner you will see the name of the Field worker logged in the device. Only two buttons are active at the main menu page. That means Field worker can only select the location for visiting, beginning with Region as the top level and/or Find the location Geo point of that particular location

Figure 4

Creating New Location

When Field worker is visiting a new household s/he will have to create a new location, to create a new location select round after select round tap **"Create Location"** button then fill in Name of location, type of location and tap **Record Location** to record GPS reading**.**

Figure 5

Once Field worker taps the **"Create Location"** button s/he will see the location page as depicted on Figure 6 below;

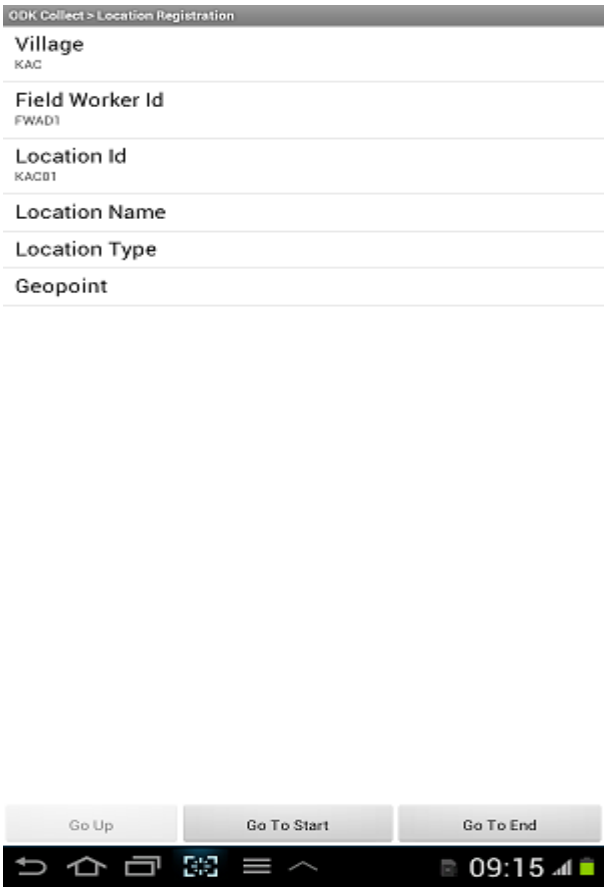


Figure 6

Most of the fields are pre-filled by the tablet, only two fields need to be filled in, these are Location Name, Location Type Geo point.

The device might take a bit longer to register the GPS coordinates accuracy depends on the number of available satellites but most of the time the best accuracy is when the reading is less than 6 meters accuracy, there you can press record location as per Figure 7. Please note the lower the numbers the better the accuracy.

Figure 7

Please note that to get the best reading is recommended tablet users to be outside the building:

Language Selection

Field workers have an option to choose the language they are comfortable to work with, therefore once they have open the form ready for data entry, they can select the language by tapping the **Menu** button as shown in figure 8 below;

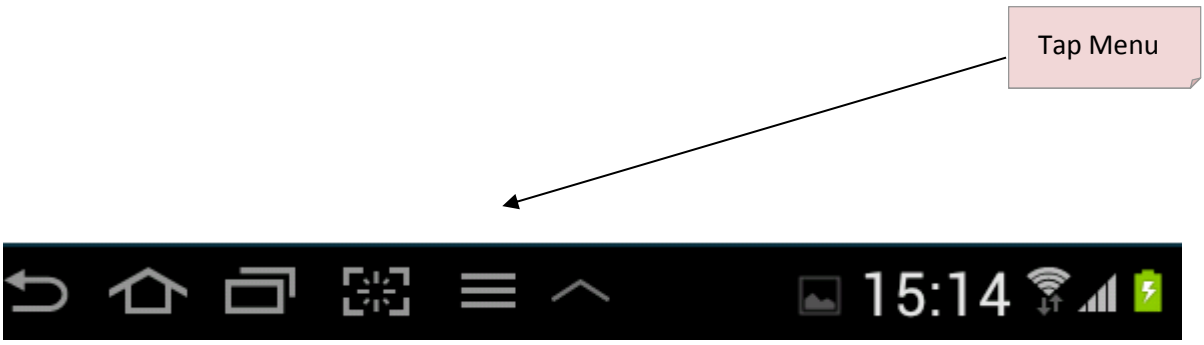


Figure 8

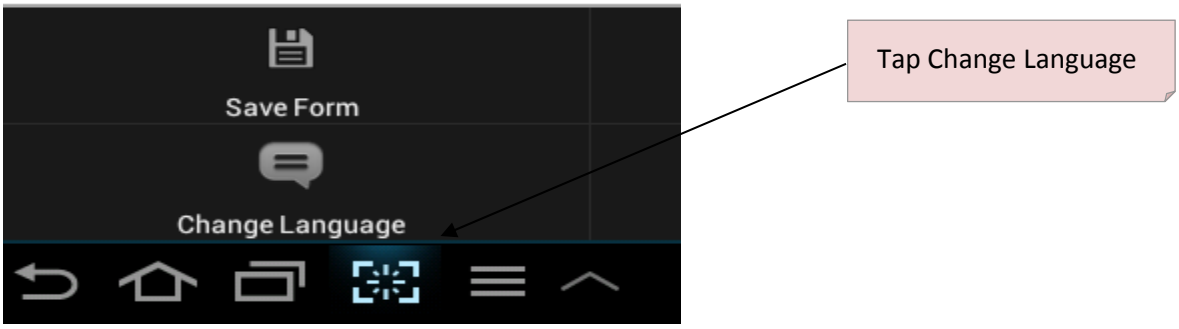


Figure 9

The tap change language to go to the Language selection menu as per Figure 10 below;

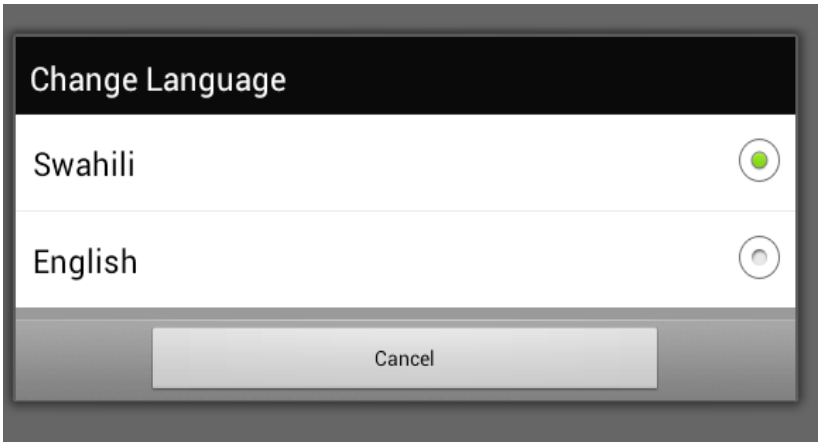


Figure 10

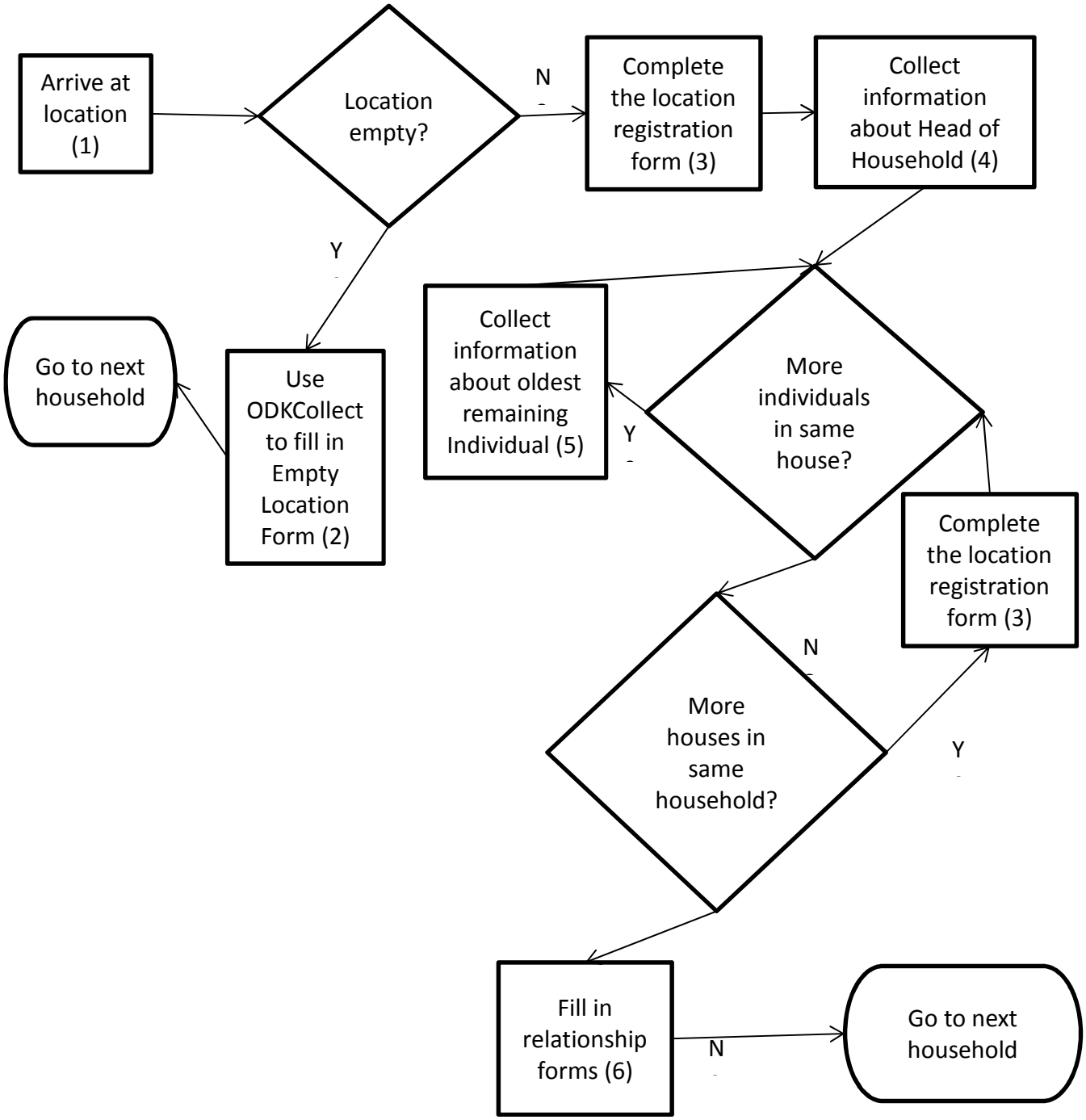
Standard Operating Procedures (SOP)

This section gives detailed descriptions on how the fieldworker has to handle each specific visit round and the included duties.

Baseline Visit

Flowchart

Following flowchart shows the basic workflow of a baseline visit:



Outline

(1) On arrival at the household, find the head of the household. If no one is in the location, an empty location form is completed to keep track of houses that need to be revisited. TODO: we need the head of household (not a different respondent) because of the consent, correct?

(2) The empty location form, unlike all other forms, needs to be filled in by starting ODKCollect, choosing fill blank form, and completing the questionnaire

(3) First collect the information about the location (house) in which the head of household lives. If this is the first house in the household, log in to openHDS, and find the village by going down the location hierarchy. Select “Create location” and complete the questionnaire. For further houses, come back to this step after completing (4) and (5) as shown in the flowchart.

(4) Record information about the head of household by first filling in the baseline form (select “Baseline “ to start the form). Select “no” in response to the prompts if mother and father are known. After completing the form chose “Membership”. Chose “Create” when asked whether to search for or create a household. Fill in the household information, use the last name of the head of household as household name. (TODO: family vs cohort definition?). Finish by selecting “Clear individual”. TODO: date of start of residency definition

(5) If there are more individuals living in the same house, register them in decreasing order of age. First fill in the Baseline form. If the parents are in the same household, respond yes to the prompt whether they are registered in the system, and search for the correct individual. After completing the Baseline form complete the Membership form.

(6) Fill in the relationship form for married individuals in the same household

Follow-up Visit

Find Existing Location

A Field worker should be able to explore the location for enumeration starting at Region level down to Household they wish to visit,

Figure 11

Filter Existing Location

Sometimes a Field Worker my encounter a very long list of locations. To quickly and accurately enable them select the desired location, they will use location filtering by tapping **"Filter Location"** button and the Figure 12 below shows the search location page for user to search the actual location they need to visit.

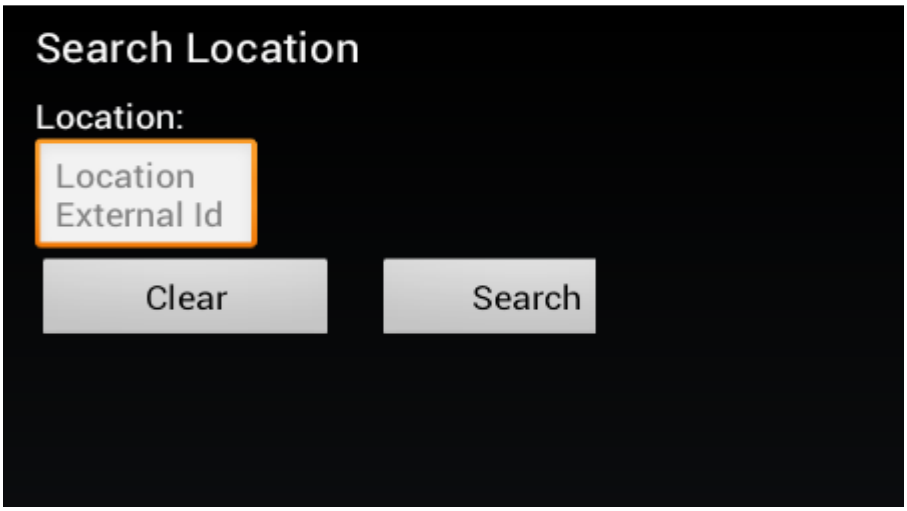


Figure 12

Create Visit

Every time a Field worker visits the household s/he will have to create a visit by tapping the **"Create Visit"** button. This can only be done once the location has been selected. So after tapping the visit the **Search for an Individual page** displays as per Figure 13 below. There a Field worker can tap Search individual after selecting location information and individual names, or just tap the search button without filing in any data for results. A successful search will populate the list of household members in that location ready for the Field worker to pick the individual who shall be interviewed.

Please note: if you are in the household page and unable to view household members, tap **"Select Individual"** button and all members will be displayed.

Figure 13

In Migration

In Migration can be in two forms. External from outside HDSS area or Internal from HDSS area:

External in Migration:

To In migrate an individual from outside the HDSS area, tap **In Migration** button. This button can only be active when the Field worker has created a visit. Once they tap that button the pop up message window will show up asking if it was **Internal** or **External**, the Field worker selects the external as per figure 9 below;

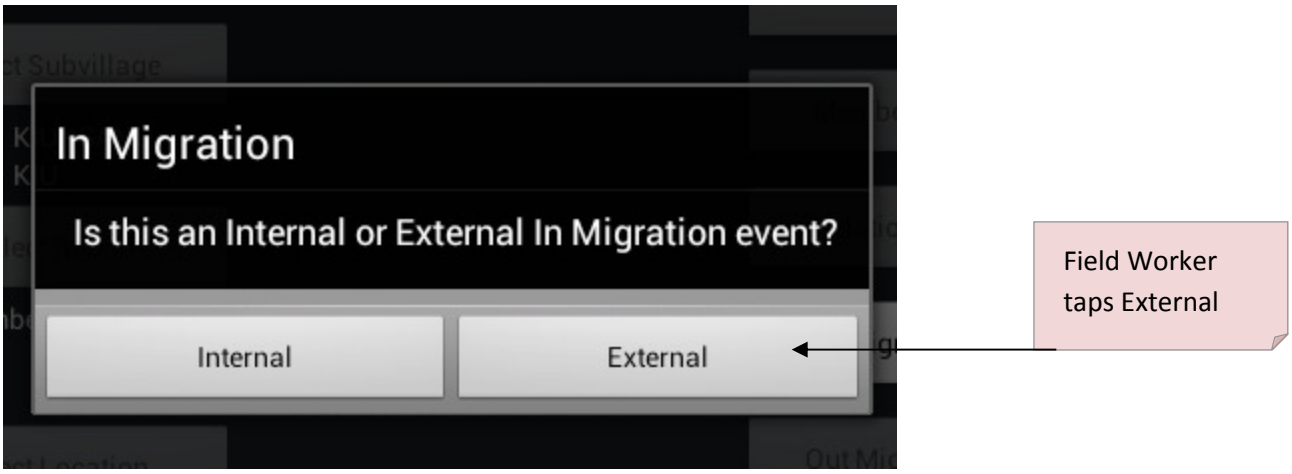


Figure 14

The tablet will try to link mother and father information of the in migrating individual to the existing individual, therefore it will ask if the mother and father of the in migrating individual are known and registered in the system as per Figures 15 and 16 below.

There he should search for the individual to in migrate from the tablet.

IMPORTANT: Please note that at some point when an individual is asked whether s/he has ever registered with the HDSS, s/he might say **NO** which at some point it might not be the truth. To avoid duplicate individual registration to HDSS system, Field worker **MUST** search for the individual's information to verify their existence in the tablet before attempting to add as new individual.

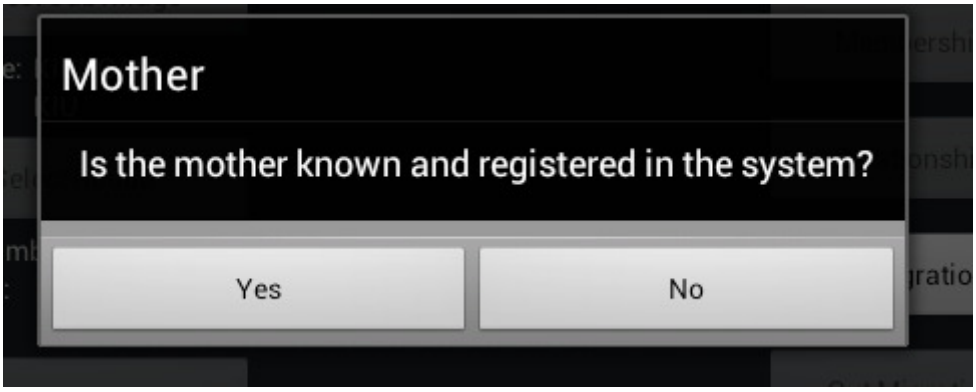


Figure 15

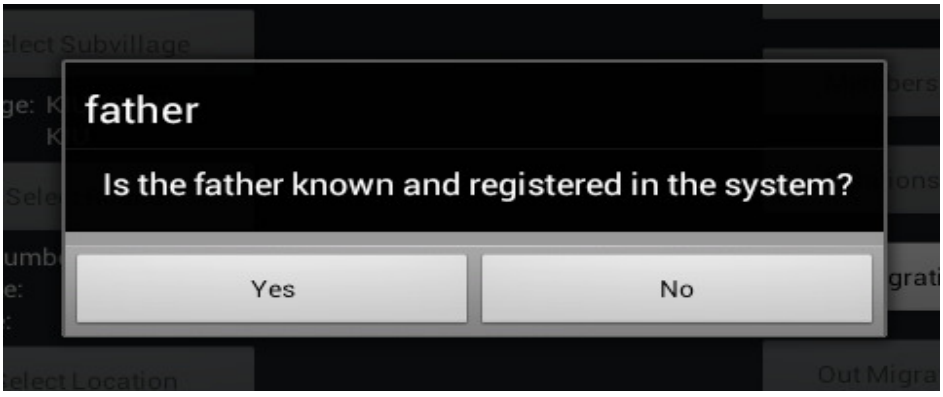


Figure 16

If either mother and/or father of the in migrating individual are known then the tablet will open the search for individual page ready for Field worker to search for the mother and father of the in migrating individual starting with mother and then mother to create the membership as per Figure 17 below:

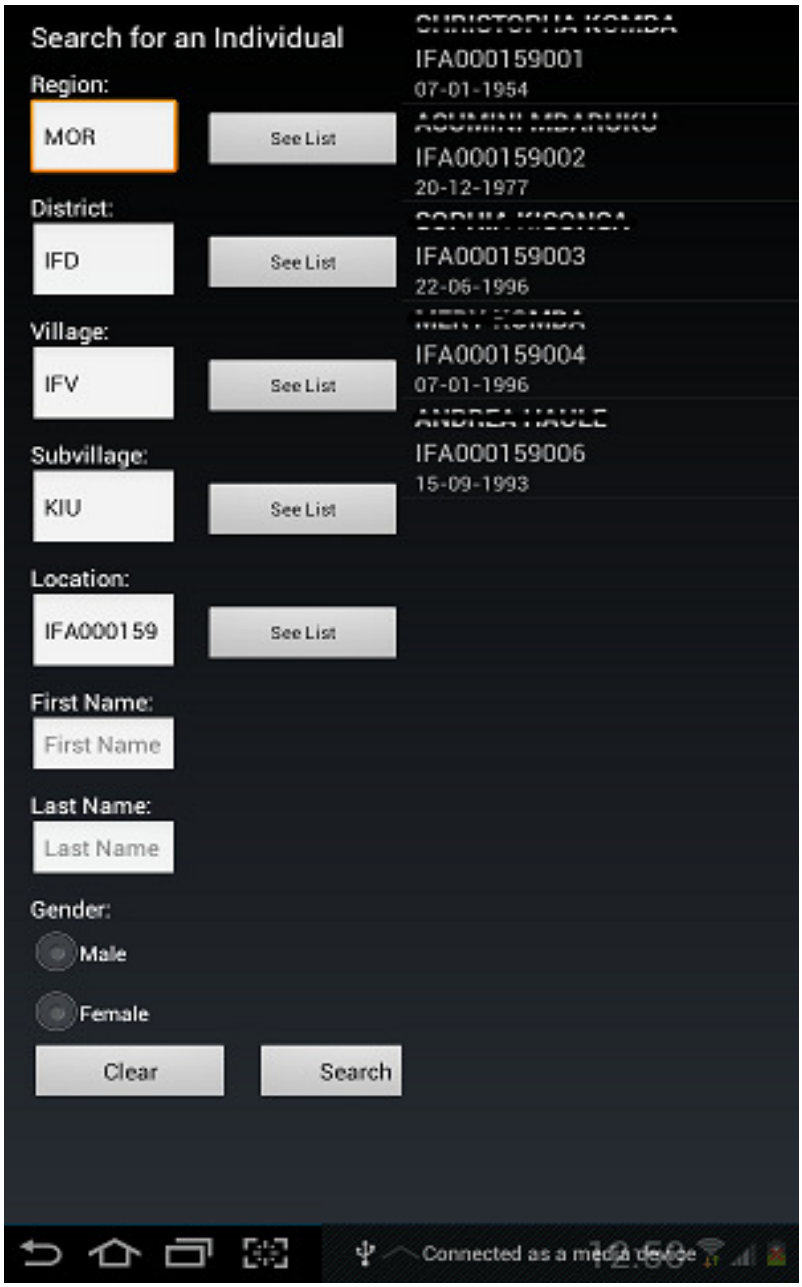


Figure 17

Please bear in mind that the searching functionality will allow Field worker to search from different location hierarchy that is to say they can search on region level to get more results or search on village level to get fewer results. Therefore if Field worker leaves blank in **Village** text box then the search will display all individuals for that district in the **Region.**

Once Field worker finds the matching mother or father the tablet will assign them to that individual and take Field worker on the page for registering the in migrating individual information as depicted in Figure 18 below;

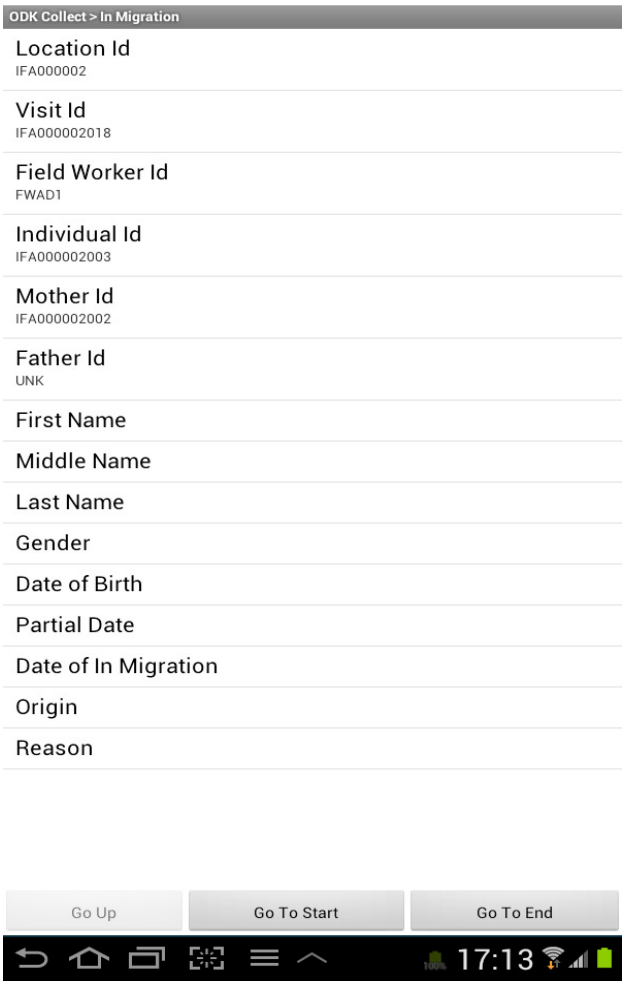


Figure 18

If both mother and father are unknown then the tablet will instead directly open the In Migration form with some pre-filled information ready for the Field worker to continue with filling in the names and other information of the individual as per Figure 18 above.

After external in-migration the Field worker has to create a membership. The pop up window will show up prompting the Field worker to create membership of the individual. Field Worker **MUST** create the membership. Below Figure 19 is the window asking Field worker to create membership. Also please follow the procedures for adding individual membership on the **membership** section;

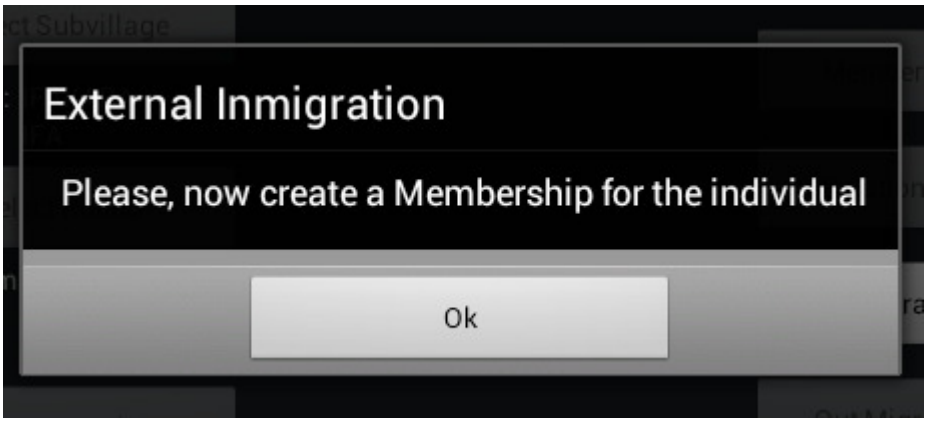


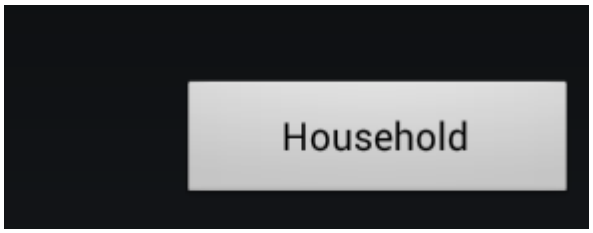
Figure 19

If we are migrating a social group in a new household first the field worker should migrate the head of the household and once done it then

Select the head of the head of the household and create Household (button Household).

Then create the membership for him (indicating him as the head)

then membership for all others individuals



Internal In Migration:

If the individual has migrated from internal the HDSS, once a pop up message shows up tap Internal to in migrate the individual as depicted in Figure 20 below;

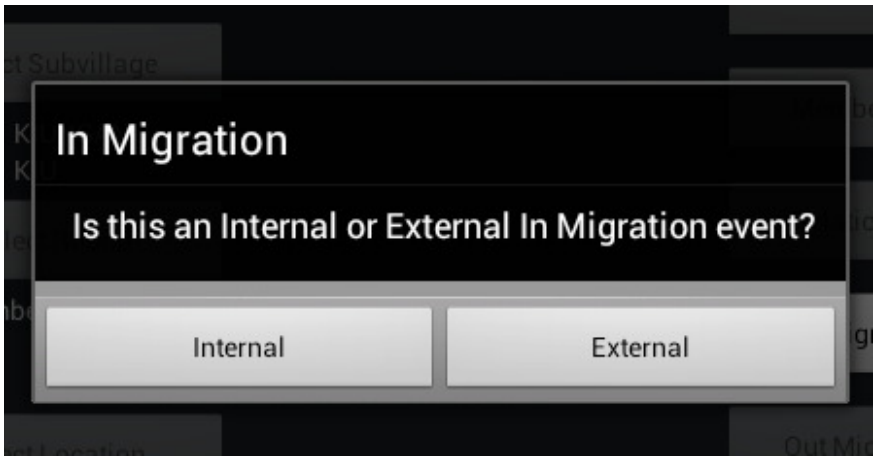


Figure 20

For the internal In Migration the individual to in migrate shall already be available in the tablet, therefore Field worker will be required to search for the individual to in migrate when the **"Search for an Individual"** page opens as shown in Figure 21 below. Every button next to the text box field will populate the list of available location information for Field worker to pick the necessary information for Field worker to search for the individual to in migrate

Field Worker can tap **See List** button to view the Location IDs of different places to search the individual to in migrate

Figure 21

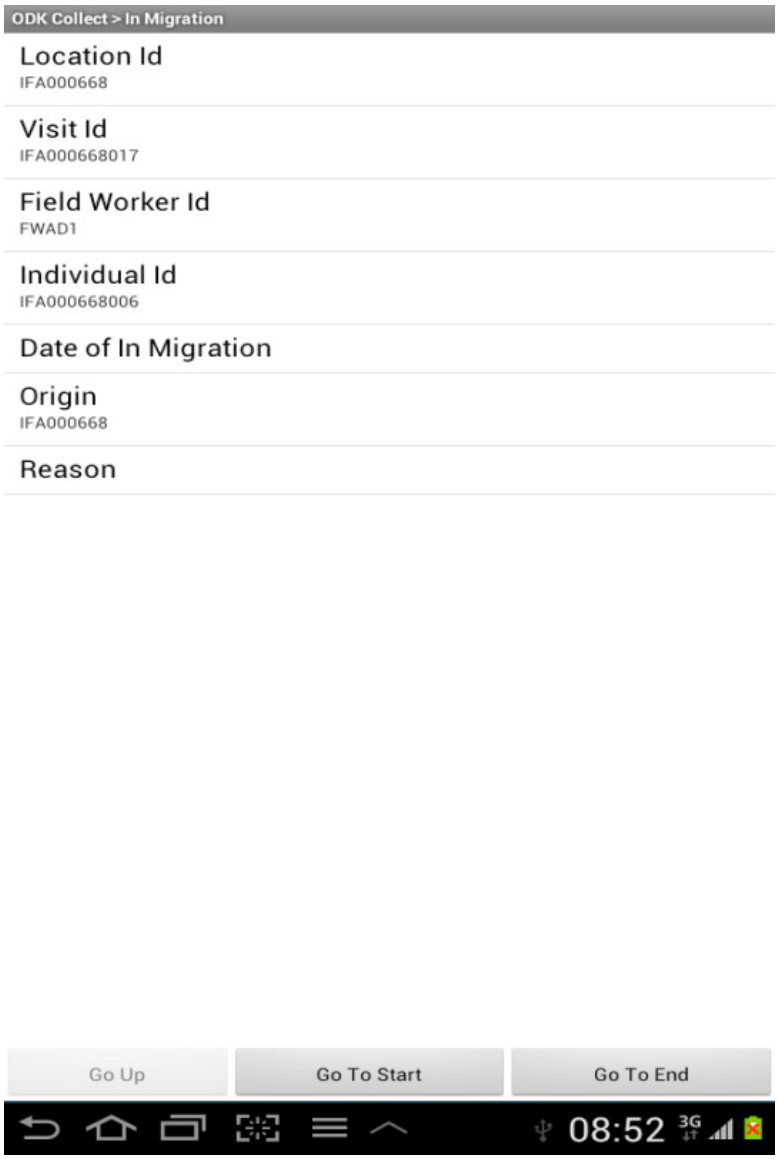


Figure 22

Once a Field worker finds the individual they wish to in migrate s/he will tap to proceed to the next page for finalizing the in migration process as displayed in Figure 22 above. Field worker should fill in and save the form and exit.

Out Migration

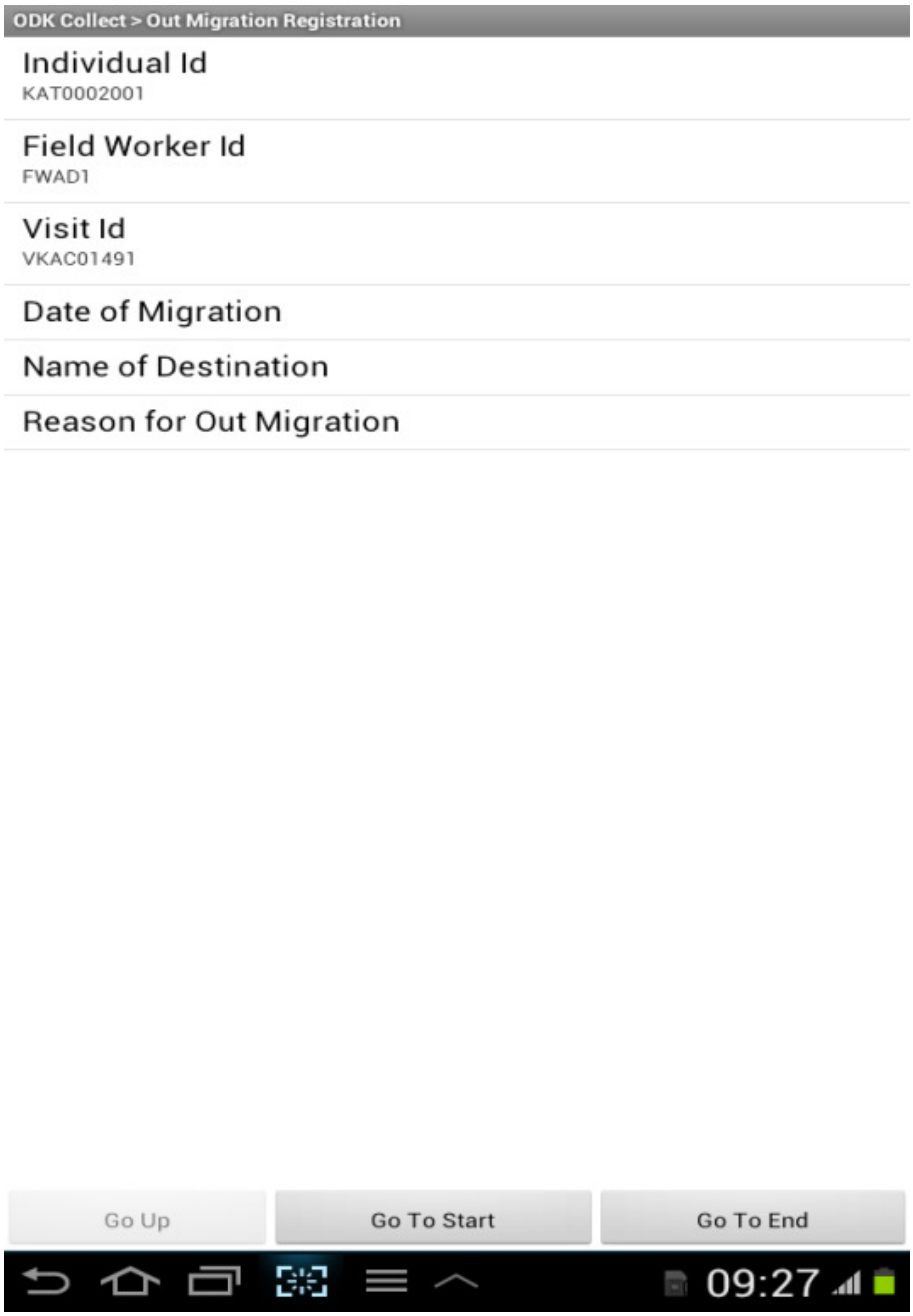
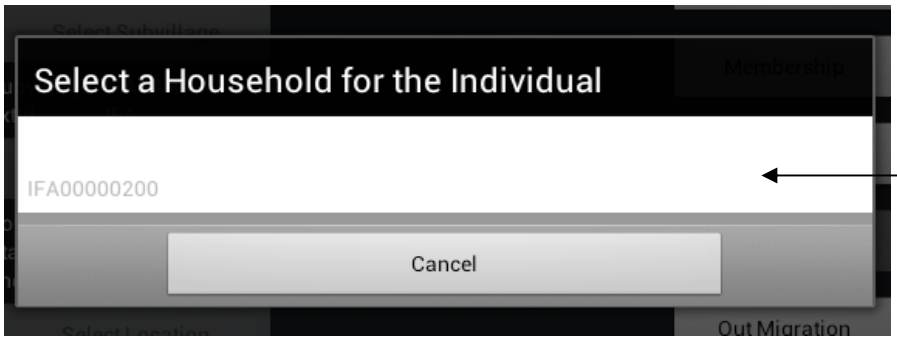


Figure 23

Most of the fields will be pre-filled except for Date of Out Migration, Name of destination and Reason for Out migration.

Membership

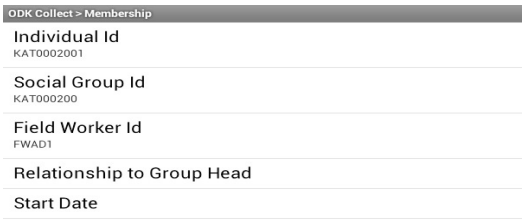
Membership **MUST** be created soon after the individual has been in migrated in the household otherwise the tablet will not allow Field worker to perform other events of that individual. To create membership Field worker should tap "**Create Membership"** button then tap location ID to select the household to in migrate the individual as per Figure 24 below depicting



Field Worker taps Household ID to pick the household for in migration

Figure 24

Field worker should start filling the blank fields as depicted in Figure 25 below, once finished should save the form and exit at the end of the form;



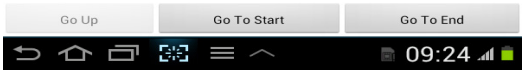


Figure 25

Relationship

To create the relationship tap **"Select Individual"** button and the list of all individual will be listed, and the Field worker will pick the household member they wish to add the relationship by tapping that member and next the Field worker will tap the "**Relationship",** then the Field worker will search for the individual to add the relationship with. The Field worker should press **"Search"** button to list individual and pick the one s/he is going to add the relationship and the Relationship page will come up for the Field worker for finalize the relationship by filling in *Relationship type* and *Start Date* as shown in Figure 26 below:

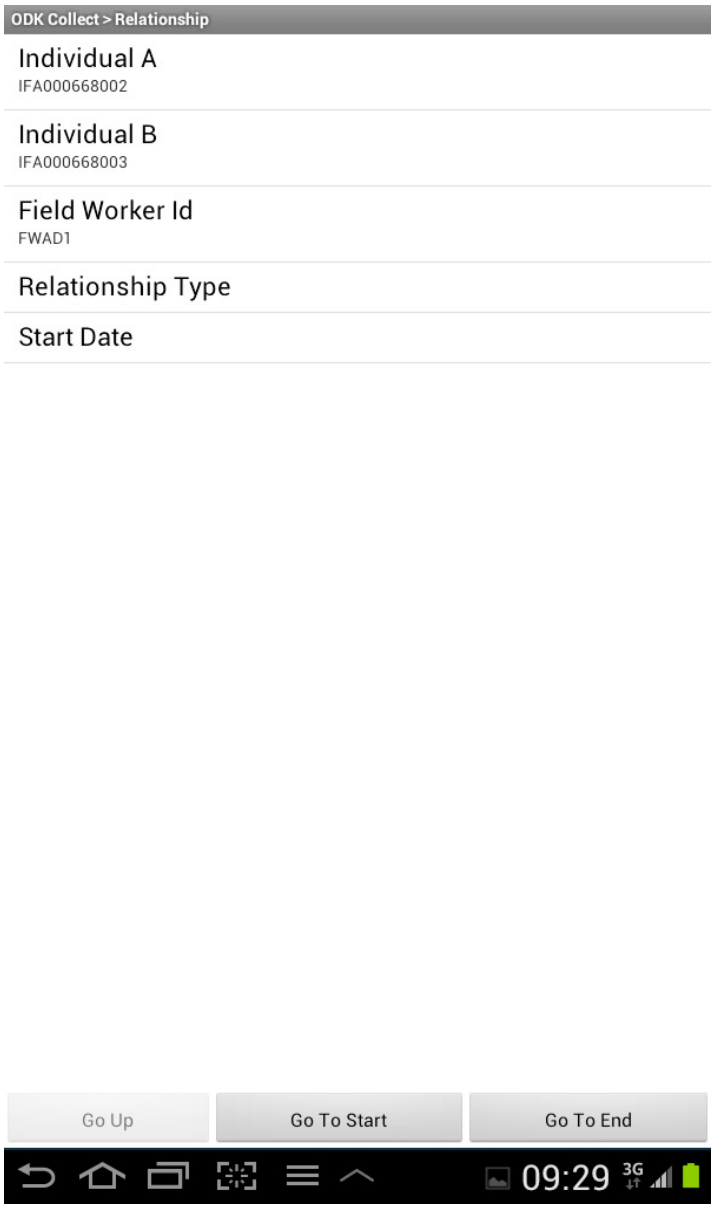


Figure 26

Pregnancy Observation

To create pregnancy observation in the household, tap the household member then tap the **"Pregnancy Observation"** button to update her pregnancy observation as per Figure 27 below;

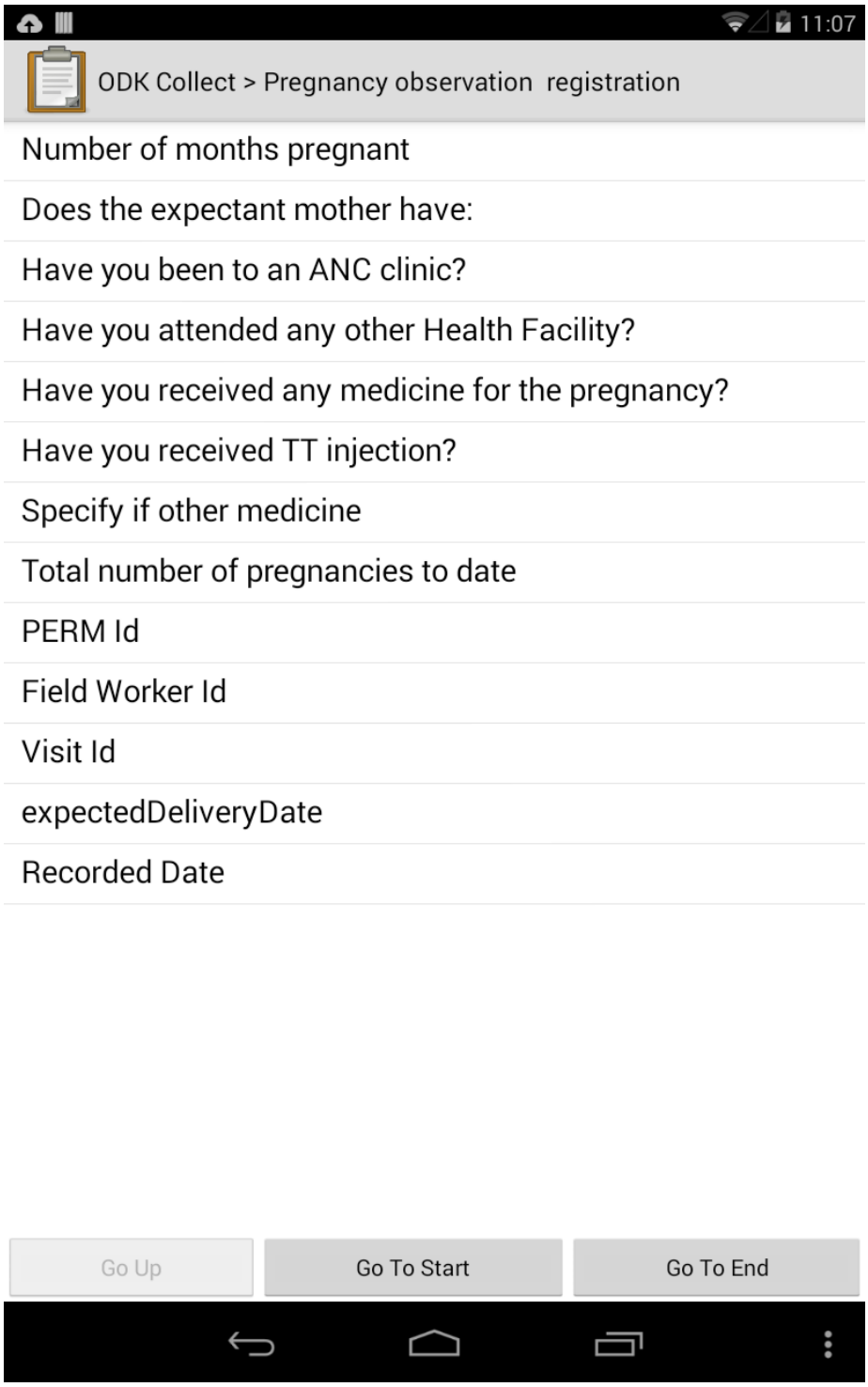


Figure 27

Clear Individual

Please note that if you have finished to add events of any individual in the household you can clear the individual for the purpose of select another individual by tapping the **"Clear Individual"** button as shown in Figure 28 below.

Figure 28

Pregnancy Outcome

To record a pregnancy outcome Field worker should tap the **"pregnancy Outcome"** button after selected the mother and this following window will display option to select the number of live births, tap the number of births as shown in Figure29 below:

Figure 29

Tap the father in the **Choose father**, window to link the new born with their father as depicted in Figure 30 below if that is the father of the new born (if the mother has a relationship with an individual, the individual will be suggested as first choice):

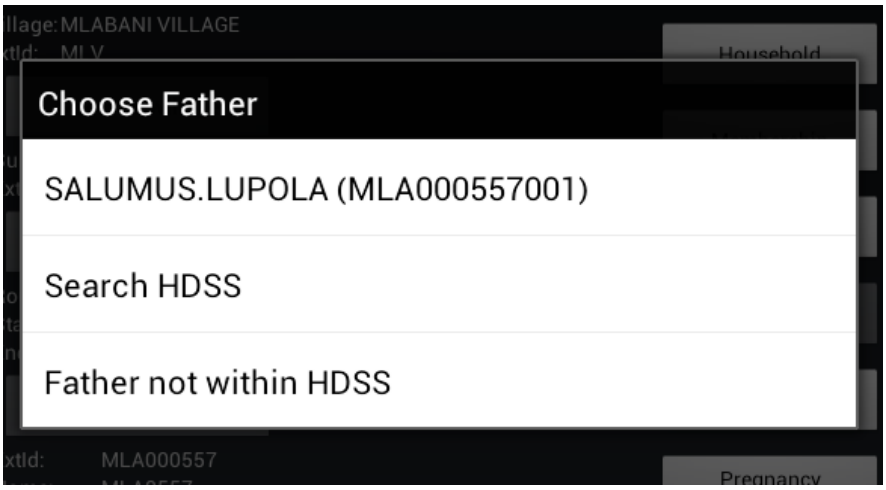


Figure 30

Please note, if there was more than one child born the process for fill the form will repeat according to the number of those births, fill in the pregnancy outcome accordingly, save form and exit.

Death

To register the death of individual once the Field worker is in select the deceased household member will click the death button and the form will open for the Field worker to fill in Date of Death, Place of Death and Cause of Death as per Figure 31 below.

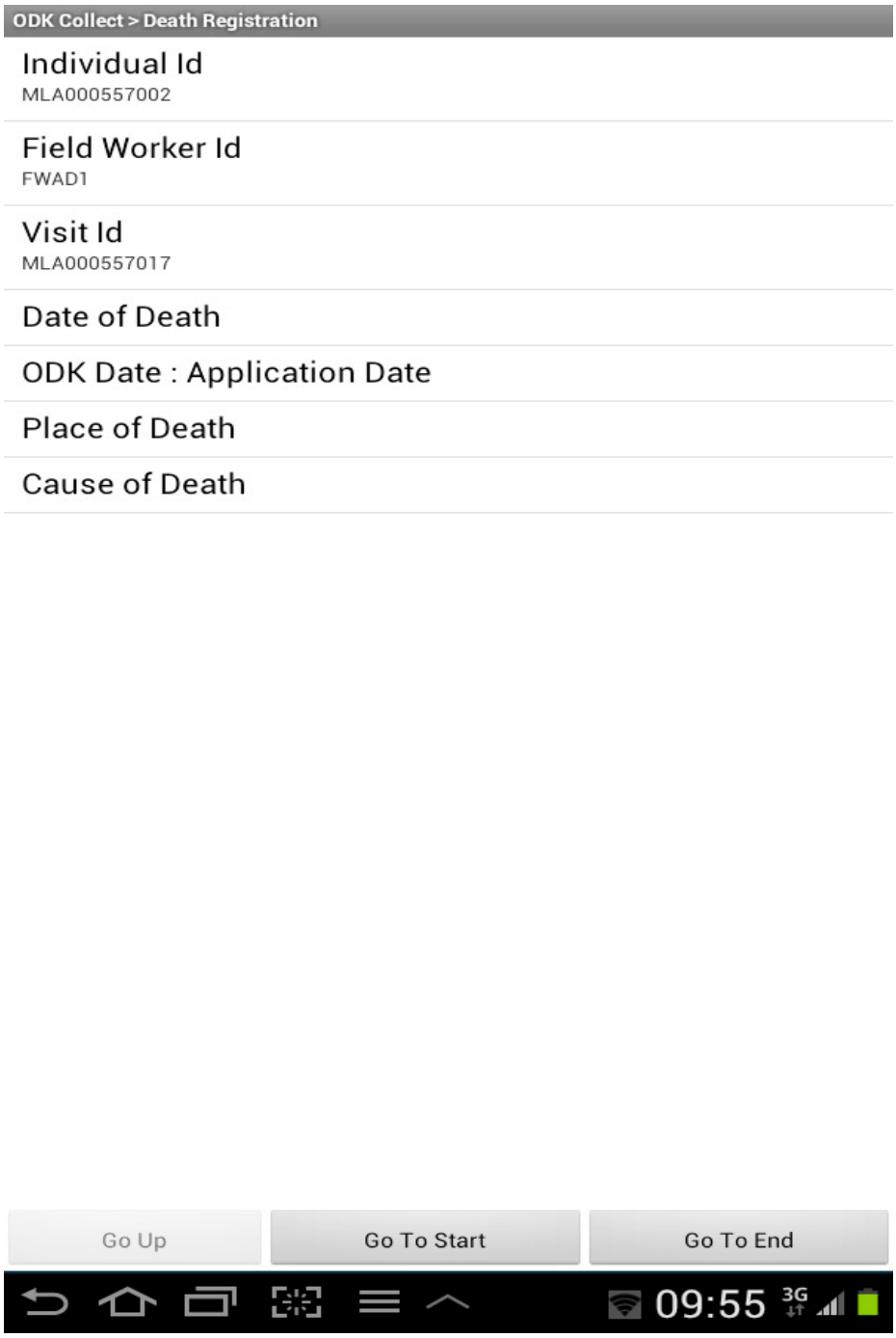
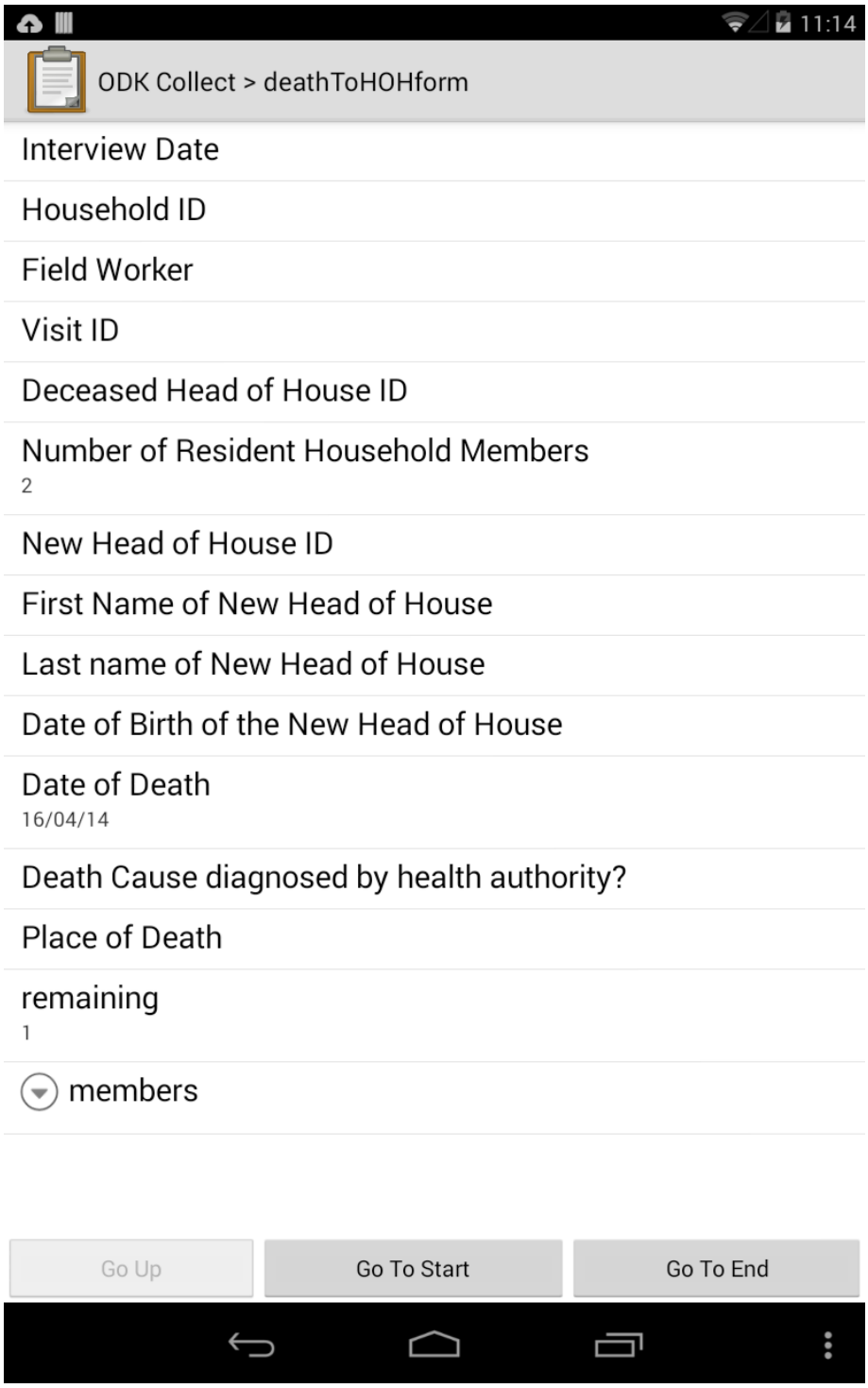


Figure 31

There's a particular case when the deceased person is the head of the household.

In this case the Death for Head of household form will automatically will be opened and there the Field worker has to indicate also the new Head of the household and the membership changes for all the members of the household as shown in the below picture.



Finish Visit

To finish visit can only be made when the Field worker has completely recorded all the events of the household and ready to leave the premises, otherwise Field Worker are not allow to tap the **Finish Visit** button. When Field worker finishes the visit the application will take them to the main menu, ready for them to visit the next household. To finish visit tap the **"Finish Visit"** as shown in Figure 32 below

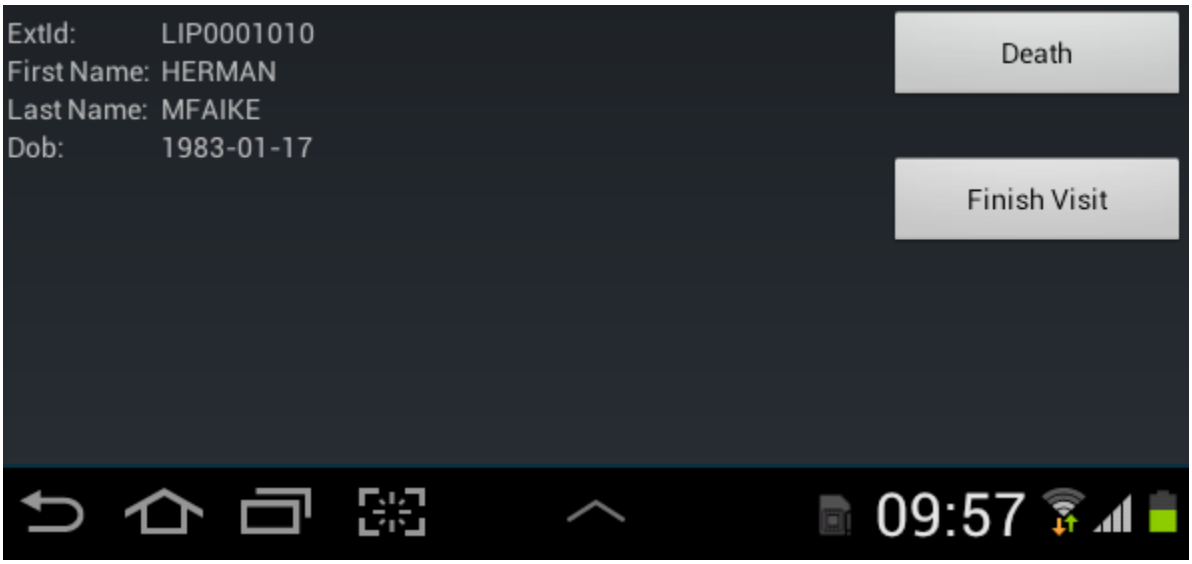


Figure 32

For the purpose of checking that the Field worker has not accidentally tapped the **"Finish Visit"** button, the pop up message will come up asking Field worker to confirm if they have finished capturing all the events for that particular household and they wish to finish the visit as per Figure 33 below displaying: Tap **Yes** to finish the visit, otherwise tap **cancel** to proceed with the visit

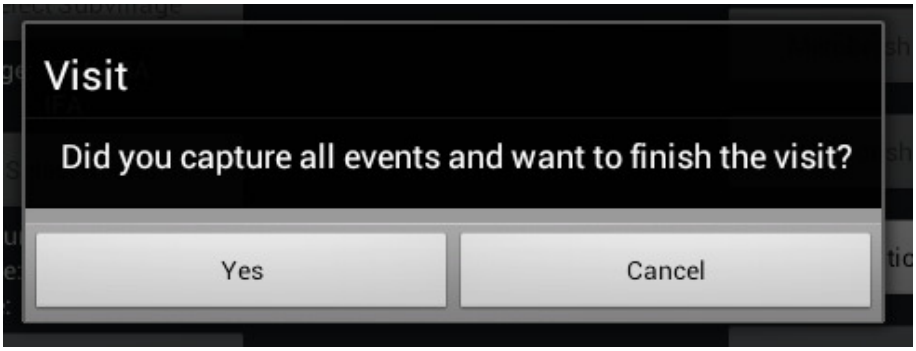


Figure 33

Data manager manual

Extra forms

The tablet component of openHDS can be extended to allow for data collection beyond the core HDSS events and entities, while making use of the demographic database stored on the tablet. Additional (ODK-) questionnaires can be added using the “Extra forms” functionality as follows:

On the server side select “ODK Forms” from the “Utility routines” menu, and define any extra forms required for the current round of enumeration. All forms have the following attributes:

-Name: (corresponds to jformId of the ODK uploaded file, and has to be unique)

-Active: you can define when a form is active or not (you can decide to deactivate during a certain period, and re-activate at a later stage)

-Gender: this is a filter, and allows specifying if the form is just for Males or Females or applying to all.

On the tablet side log in as Supervisor, where you will have the option to download the extra form list (this downloads only the form definition stubs. The real forms should be as usual downloaded from using ODKCollect)

When a Field worker starts the survey, once he get at Individual level he'll see on the top right of the screen a new Menu: "Extra forms". There he can search for all or search by name. Upon selection the corresponding form will appear already prefilled with all the IDs previously selected.

If the form is not physically present on the tablet the Field worker will get a warning message.

Every time a property of an extra form is changed (e.g. hidden or unhidden, or via a change of the filter rule), or if a new form is added, form definition stubs on the tablet have to be updated by repeating the download step described above.

Migration of legacy data to openHDS

This section explains the necessary steps to migrate from the HRS2 platform to its successor OpenHDS.

HRS2 was developed using Microsoft FoxPro. As such it also uses its underlying database structure. The database files are saved locally as separate DBF files for each table, which can be easily copied for data migration.

Attention!

To maintain the highest level of data quality, it is highly recommended that the data migration is only to be done in close collaboration with the Data-Manager (DM) of the migrating site, to avoid ambiguity.

Requirements

Before you start with this guide, please make sure you have all the required files and applications:

- HRS2 Database Dump (DBF-Files)
- OpenHDS with MySQL as database system

File Conversion and staging DSS database

All DBF-Files should be in one directory.

In MySQL (e.g. over the MySQL Workbench Application), we need a database named hrs_mysql_db. This database is only needed temporary and will be used to clean the data.

Running the script 01_hrs_mysql_db.sql will create this database.

Copy the conversion R script 02_dbf_mysql.R to the same directory file and edit it to configure the database's connection parameters.

Run the conversion R script that will convert the dbf to mysql, making sure the that the working directory matches the location of the dbf files. The script will first create the empty staging tables needed, and then populate them.

Required tables for migration

After the creation of the staging DSS database in MySQL, the following tables should be in the database (same name).

birth

death

individual

indvstatus

inmigration

location

membership

observation

outmigration

pregoutcome

relationship

residency

round

socialgroup

If a table is not there please check the reason.

If a table is there with a different name, please run the following alter table to change the name.

```
ALTER TABLE WRONGTABLENAME RENAME CORRECTTABLENAME;
```

Adding required attributes

All tables which are now populated from the HRS2 records in the dbf files need some modifications in order for the migration to OpenHDS to work. Some steps to all tables, table-specific alterations are described below.

Create Fieldworker FWAD1 (Admin Data) in openhds database;

insert into `fieldworker`(`uuid`,`deleted`,`insertDate`,`voidDate`,`voidReason`,`extId`,`firstName`,`lastName`,`insertBy_uuid`,`voidBy_uuid`) values ('0318a3bd496578ed014965a075dc001b',0,'2014-10-31 00:00:00',null,null,'FWAD1','Admin','Data','User 1',null);

Common steps for all tables

Add new column: **FIELDWORKER** VARCHAR(6)

Add new column: **processed_by_mirth** CHAR(1) DEFAULT ‘0’

Add new column: **id** INT NOT NULL AUTO_INCREMENT PRIMARY KEY(id)

Using the following script (change TABLE_NAME to the real table name)

ALTER TABLE TABLE_NAME

ADD FIELDWORKER VARCHAR(6) DEFAULT 'FWAD1',

ADD processed_by_mirth VARCHAR(1) DEFAULT '0',

ADD id INT AUTO_INCREMENT NOT NULL,

ADD PRIMARY KEY (id);

For each table check that the requirements for IDs defined in openHDS are respected:

Location Id 9 digits (e.g. KWM000087)

Visit ID 12 digits (e.g. KWM000087005 = Location ID + round 3 digit))

SocialGroup/Household ID 11 digits (e.g. KWM00008700 = Location ID + 00)

Individual ID 12 digits (e.g.KWM000087001 + Location ID + individual cardinality 001,002…)

In case it doesn't respect the requirement please pad with 0 (zero) to get the required length.

Steps for each table

Round

Run common SQL-queries.

If all worked ok then the following query should work correctly and all the columns should have a value.

select id, ROUND_NUM, str_to_date(START_DATE,'%Y-%m-%d') START_DATE, str_to_date(END_DATE,'%Y-%m-%d') END_DATE, REMARKS FROM round where processed_by_mirth=0

Location

Run common SQL-queries.

Extra Scripts:

Add new column: **altitude** CHAR(1)

Add new column: **latitude** CHAR(1)

Add new column: **longitude** CHAR(1)

Add new column: **accuracy** CHAR(1)

Add new column: **subvillageId** CHAR(3)

ALTER TABLE location

ADD altitude CHAR(1) default '0',

ADD latitude CHAR(1) default '0' ,

ADD longitude CHAR(1) default '0' ,

ADD accuracy CHAR(1) default '0' ,

ADD subvillageId VARCHAR(3) ;

Clean eventual duplications on the Location table.
select count(LOCATIONID),LOCATIONID from LOCATION group by LOCATIONID order by 1 desc;

ALTER TABLE location
ADD UNIQUE INDEX idx_locid (LOCATIONID);

- Before migrating the Locations, please collaborate with the DM to recreate the Location-Hierarchy!
- SubvillageID on the location table must have the correct value. Please update it with the values provided from the data managers.

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, subvillageId , LOCATIONID, FIELDWORKER, 'RUR' LOCATION_TYPE, ACCURACY, ALTITUDE, LONGITUDE, LATITUDE FROM location WHERE processed_by_mirth =0;

Please check that Location ID has the right format.

Location Id 9 digits (e.g. KWM000087)

In case not update it:

e.g. if Location ID is IFA0301 instead of IFA000301.

UPDATE LOCATION set LOCATIONID=CONCAT(LEFT(LOCATIONID,3),'00',RIGHT(LOCATIONID,4))

Observation (maps to Visit)

Run common SQL-queries.

Then:

ALTER TABLE observation

ADD INDEX idx_observeid (OBSERVEID);

ALTER TABLE observation

ADD INDEX idx_obslocation (LOCATIONID);

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, str_to_date(DATE,'%Y-%m-%d') DATE, OBSERVEID, LOCATIONID, FIELDWORKER, ROUND FROM observation WHERE processed_by_mirth=0;

Check that there are no records with OBSERVEID null.

If yes, having the Location ID and the Round number update the records accordingly

e.g. update observation set OBSERVEID=concat((LOCATIONID), LPAD(ROUND, 3, '0')) where OBSERVEID is null;

Please check that Location ID and OBSERVEID (VISITID) have the right format.

Location Id 9 digits (e.g. KWM000087)

Visit ID 12 digits (e.g. KWM000087005 = Location ID + round 3 digit))

e.g. if Location ID is IFA0301 instead of IFA000301.

UPDATE OBSERVATION set LOCATIONID=CONCAT(LEFT(LOCATIONID,3),'00',RIGHT(LOCATIONID,4));

e.g. if OBSERVEID ID is IFA024200 instead of IFA000024200.

UPDATE OBSERVATION set OBSERVEID=CONCAT(LEFT(OBSERVEID,3),'000',RIGHT(OBSERVEID,6)) ;

Verify and find/fix if there are LOCATIONS in the observation table that are not existing in the location table:

select LOCATIONID FROM observation where LOCATIONID not in (select LOCATIONID FROM location);

Individual

Run common SQL-queries.

Add new column: **approximate** INT DEFAULT 1 (maps to dobAspect)

Add new column: **FIRSTNAME** VARCHAR(50)

Add new column: **MIDDLENAME** VARCHAR(50)

Add new column: **LASTNAME** VARCHAR(50)

ALTER TABLE individual

ADD FIRSTNAME VARCHAR(50),

ADD MIDDLENAME VARCHAR(50) ,

ADD LASTNAME VARCHAR(50) ,

ADD approximate INT(1) DEFAULT 1;

- Since the name of an Individual in HRS2 is not separated into a First-, Middle- and Lastname as in OpenHDS, we have to split up this entry.
  For this step, please execute the following SQL query. This will also put an entry of UNKNOWN into the LASTNAME-column, in case the Lastname is still not set.

update individual SET
FIRSTNAME = SUBSTRING_INDEX(name, ' ', 1)

```
 ,MIDDLENAME = case when locate(" ",Ltrim(SUBSTRING(name, length(SUBSTRING_INDEX(name,'\ ', 1))+1)))=0
then ''
else SUBSTRING_INDEX(Ltrim(SUBSTRING(name, length(SUBSTRING_INDEX(name,'\ ', 1))+1)), ' ', 1)
end
 ,LASTNAME = case when locate(" ",Ltrim(SUBSTRING(name, length(SUBSTRING_INDEX(name,'\ ', 1))+1)))=0
then Ltrim(SUBSTRING(name, length(SUBSTRING_INDEX(name,'\ ', 1))+1))
else SUBSTRING_INDEX(Ltrim(SUBSTRING(name, length(SUBSTRING_INDEX(name,'\ ', 1))+1)), ' ', -1) end;

update individual set LASTNAME='UNKNOWN' where length(LASTNAME) =0;
```

- Replace a Father- and MotherID of '**Q**' with '**UNK'**.

```
UPDATE individual SET MotherID='UNK' where MotherID='Q';
UPDATE individual SET FatherID='UNK' where FatherID ='Q';

UPDATE individual SET MotherID='UNK' where MotherID is null;
UPDATE individual SET FatherID ='UNK' where FatherID is null;
```

Add index on INDIVIDID

```
ALTER TABLE individual
    ADD INDEX idx_individid (INDIVIDID);
```

Check that the individual IDs are unique:

```
select count(INDIVIDID), INDIVIDID from individual group by INDIVIDID order by 1 desc;
```

If all worked ok then the following query should work correctly and all the columns should have a value.

```
SELECT id, str_to_date(BIRTH_DATE,'%Y-%m-%d') BIRTH_DATE, FIRSTNAME,MIDDLENAME,
MOTHERID, INDIVIDID,  GENDER, FIELDWORKER, LASTNAME, approximate, FATHERID FROM
individual WHERE processed_by_mirth=0 order by BIRTH_DATE ASC;
```

Please check that Individual ID is 12 digits (e.g.KWM000087001)

e.g. if is VIS2287005 instead of VIS002287005

```
UPDATE INDIVIDUAL  set INDIVIDID=CONCAT(LEFT(INDIVIDID,3),'00',RIGHT(INDIVIDID,7)) ;
```

Same thing for MOTHER and FATHER ID:

```
UPDATE INDIVIDUAL  set MOTHERID=CONCAT(LEFT(MOTHERID,3),'00',RIGHT(MOTHERID,7))
where length(MOTHERID)>6;

UPDATE INDIVIDUAL  set FATHERID=CONCAT(LEFT(FATHERID,3),'00',RIGHT(FATHERID,7)) where
length(FATHERID)>6;
```

Socialgroup

Run common SQL-queries.

Create indexes

```
ALTER TABLE socialgroup
  ADD INDEX idx_sgid (SOCIALGPID);

ALTER TABLE socialgroup
  ADD INDEX idx_headid (HEADID);
```

Please check that head ID is 12 digits (e.g.KWM000087001)

```
UPDATE socialgroup  set HEADID=CONCAT(LEFT(HEADID,3),'00',RIGHT(HEADID,7)) ;
```

Please check that SocialGroups ID is 11 digits (e.g.KWM00008700)

```
UPDATE socialgroup  set SOCIALGPID=CONCAT(LEFT(SOCIALGPID,3),'00',RIGHT(SOCIALGPID,6)) ;
```

Be sure that all SocialGroups have a head known in individual table. Fix the one that result from the query below.

```
select * from socialgroup where HEADID not in (select INDIVIDID FROM individual);
```

If all worked ok then the following query should work correctly and all the columns should have a value.

```
select id, str_to_date(DE_DATE,'%Y-%m-%d') DE_DATE , SOCIALGPID , FIELDWORKER , NAME, TYPE ,
HEADID  FROM socialgroup WHERE processed_by_mirth=0;
```

Relationship

Run common SQL-queries.

create indexes:

```
ALTER TABLE relationship

  ADD INDEX IDX_RELIND (INDIVIDID);

ALTER TABLE relationship

  ADD INDEX IDX_RELIND2 (INDIVIDID2);
```

Update the column Relationship Type (TYPE) to accepted valid values:

1 - Never Married

2 - Married

3 - Separated/Divorced

4 - Widowed

5 - Living Together

UPDATE relationship set TYPE='2' WHERE TYPE='MRT';

Update the column EEVENTTYPE to accepted/valid values:

DTH - Death

OMG – Out Migration

DHH – Death of Head Household

NA – None

UPDATE relationship set EEVENTTYPE ='NA' WHERE EEVENTTYPE is null;

UPDATE relationship set EEVENTTYPE ='DTH' WHERE EEVENTTYPE ='WID';

Please check that Individual ID is 12 digits (e.g.KWM000087001)

UPDATE relationship set INDIVIDID=CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**)) ;

UPDATE relationship set INDIVIDID2=CONCAT(LEFT(INDIVIDID2,**3**),'00',RIGHT(INDIVIDID2,**7**)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, str_to_date(DE_DATE,'%Y-%m-%d') DE_DATE, str_to_date(SDATE,'%Y-%m-%d') SDATE, INDIVIDID2, FIELDWORKER, INDIVIDID, TYPE FROM relationship WHERE processed_by_mirth =**0**;

Membership

Run common SQL-queries.

create indexes:

ALTER TABLE membership

ADD INDEX IDX_MEMIND (INDIVIDID);

```
ALTER TABLE membership
 ADD INDEX IDX_RELSGP (SOCIALGPID);
```

Update the column Relation to Head (RLTN_HEAD) to accepted/valid values:

1 - Head

2 - Spouse

3 - Son/Daughter

4 - Brother/Sister

5 - Parent

6 - Grandchild

7 - Not Related

8 - Other Relative

9 - Don't Know

```
UPDATE membership  set RLTN_HEAD='1' WHERE RLTN_HEAD='ONE';
UPDATE membership  set RLTN_HEAD='2' WHERE RLTN_HEAD='WIF';
UPDATE membership  set RLTN_HEAD='3' WHERE RLTN_HEAD='CHD';
UPDATE membership  set RLTN_HEAD='4' WHERE RLTN_HEAD IN ('BRO','SIS');
UPDATE membership  set RLTN_HEAD='5' WHERE RLTN_HEAD='PAR';
UPDATE membership  set RLTN_HEAD='7' WHERE RLTN_HEAD='OTH';
UPDATE membership  set RLTN_HEAD='7' WHERE RLTN_HEAD='OTH';
UPDATE membership  set RLTN_HEAD='8' WHERE RLTN_HEAD='OT';
UPDATE membership  set RLTN_HEAD='9' WHERE RLTN_HEAD='UNK';
```

Update the column EEVENTTYPE to accepted/valid values:

DTH - Death

OMG – Out Migration

DHH – Death of Head Household

NA – None

```
UPDATE membership  set EEVENTTYPE='OMG' WHERE EEVENTTYPE ='EXT';
```

UPDATE membership set EEVENTTYPE='NA' WHERE EEVENTTYPE IS NULL;

Update the column SEVENTTYPE to accepted/valid values:

BIR - Birth

IMG – IN Migration

MAR - Marriage

ENU - Enumeration

UPDATE membership set SEVENTTYPE='OMG' WHERE SEVENTTYPE ='ENT';

Please check that INDIVIDUAL ID is 12 digits (e.g.KWM000087001)

UPDATE membership set INDIVIDID =CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

Please check that SocialGroups ID is 11 digits (e.g.KWM00008700)

UPDATE membership set SOCIALGPID=CONCAT(LEFT(SOCIALGPID,3),'00',RIGHT(SOCIALGPID,6)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, str_to_date(DE_DATE,'%Y-%m-%d') DE_DATE, str_to_date(SDATE,'%Y-%m-%d') SDATE, FIELDWORKER, SOCIALGPID, RLTN_HEAD, INDIVIDID,SEVENTTYPE,EEVENTTYPE,str_to_date(EDATE,'%Y-%m-%d') EDATE FROM membership WHERE processed_by_mirth =0 order by SDATE asc;

Pregnancy observation (indvstatus)

Run common SQL-queries.

create indexes:

ALTER TABLE indvstatus

ADD INDEX IDX_POBIND (INDIVIDID);

Please check that INDIVIDUAL ID is 12 digits (e.g.KWM000087001)

UPDATE indvstatus set INDIVIDID =CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

Please check that Visit ID 12 digits (e.g. KWM000087005 = Location ID + round 3 digit))

UPDATE indvstatus set OBSERVEID=CONCAT(LEFT(OBSERVEID,**3**),'000',RIGHT(OBSERVEID,**6**)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, OBSERVEID,DATE_ADD(str_to_date(**DATE**,'%Y-%m-%d'), INTERVAL **9**-CAST(PREG_MONTH AS SIGNED) MONTH) EXPECTED_DELIVERY_DATE, FIELDWORKER,str_to_date(**DATE**,'%Y-%m-%d') **DATE**, INDIVIDID FROM indvstatus WHERE processed_by_mirth =0

Pregnancy outcome (pregoutcome)

Run common SQL-queries.

create indexes:

ALTER TABLE pregoutcome

ADD INDEX idx_proind (INDIVIDID);

ALTER TABLE pregoutcome

ADD INDEX idx_provis (OBSERVEID);

Update FATHERID to UNK if equals to 'Q':

UPDATE pregoutcome set FATHERID='UNK' where FATHERID='Q';

Please check that INDIVIDUAL ID/FATHERID is 12 digits (e.g.KWM000087001)

UPDATE pregoutcome set INDIVIDID =CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

UPDATE pregoutcome set FATHERID =CONCAT(LEFT(FATHERID,3),'00',RIGHT(FATHERID,7)) where LENGTH(FATHERID)>6;

Please check that Visit ID 12 digits (e.g. KWM000087005 = Location ID + round 3 digit))

UPDATE pregoutcome set OBSERVEID=CONCAT(LEFT(OBSERVEID,**3**),'000',RIGHT(OBSERVEID,**6**)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, OBSERVEID, INDIVIDID, **1** PARTIAL_DATE,FIELDWORKER,str_to_date(**DATE**,'%Y-%m-%d') **DATE**, FATHERID FROM pregoutcome WHERE processed_by_mirth =0

Birth

Run common SQL-queries.

Please check that INDIVIDUAL ID is 12 digits (e.g.KWM000087001)

UPDATE birth set INDIVIDID =CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

Please check that SocialGroups ID is 11 digits (e.g.KWM00008700)

UPDATE birth set SOCIALGPID=CONCAT(LEFT(SOCIALGPID,3),'00',RIGHT(SOCIALGPID,6)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT FIRSTNAME, TYPE, SOCIALGPID , GENDER, **9** RELATIONSHIP_TO_GROUP_HEAD, a.INDIVIDID INDIVIDID, LASTNAME FROM birth a, individual b where a.INDIVIDID = b.INDIVIDID

Residency

Run common SQL-queries.

Create indexes:

ALTER TABLE residency

ADD INDEX idx_resind (INDIVIDID);

ALTER TABLE residency

ADD INDEX idx_resloc (LOCATIONID);

Please check that Individual ID is 12 digits (e.g.KWM000087001)

e.g. if is VIS2287005 instead of VIS002287005

UPDATE residency set INDIVIDID=CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

Please check that Location Id 9 digits (e.g. KWM000087)

UPDATE residency set LOCATIONID=CONCAT(LEFT(LOCATIONID,3),'00',RIGHT(LOCATIONID,4));

Check if data entries are clean. If entries are found that are not consistent, please show them to the DM so that they are fixed! (endDate<startDate, endDate or startdate in the future or before the start of DSS, etc etc)

Update the column EEVENTTYPE to accepted/valid values:

DTH - Death

OMG – Out Migration

NA – None

```
UPDATE residency set EEVENTTYPE='NA' where EEVENTTYPE is null;
```

```
UPDATE residency set EEVENTTYPE='OMG' where EEVENTTYPE ='EXT';
```

Update the column SEVENTTYPE to accepted/valid values:

BIR - Birth

IMG – IN Migration

ENU - Enumeration

UPDATE residency set SEVENTTYPE='IMG' where SEVENTTYPE ='ENT';

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, str_to_date(DE_DATE,'%Y-%m-%d') DE_DATE, str_to_date(SDATE,'%Y-%m-%d') SDATE, FIELDWORKER, LOCATIONID, INDIVIDID,SEVENTTYPE,EEVENTTYPE,str_to_date(EDATE,'%Y-%m-%d') EDATE FROM residency WHERE processed_by_mirth =0 order by SDATE asc;

Death

Run common SQL-queries.

When updating the Death locations, please collaborate with the DM to make sure numbers and locations correspond accordingly!

Common SQL-Scripts.

Reason and **Place** should be 2 columns of the death table.

If not rename the columns correspondent to these one.

e.g.

ALTER TABLE death

CHANGE DTHPLACE PLACE **VARCHAR**(**99**),

CHANGE DTHCAUSE REASON **VARCHAR**(**99**);

Then

```
update death set PLACE='UNK' where PLACE is null;

update death set REASON='UNK' where REASON is null;
```

Please check that INDIVIDUAL ID is 12 digits (e.g.KWM000087001)

UPDATE death set INDIVIDID =CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

Please check that Visit ID 12 digits (e.g. KWM000087005 = Location ID + round 3 digit))

UPDATE death set OBSERVEID=CONCAT(LEFT(OBSERVEID,**3**),'000',RIGHT(OBSERVEID,**6**)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, OBSERVEID, REASON, FIELDWORKER, str_to_date(**DATE**,'%Y-%m-%d') **DATE**, PLACE, INDIVIDID FROM death where processed_by_mirth =0;

Outmigration

Run common SQL-queries.

create indexes:

ALTER TABLE outmigration

ADD INDEX idx_omgind (INDIVIDID);

ALTER TABLE outmigration

ADD INDEX idx_omgvis (OBSERVEID);

REASON should be a column of the outmigration table. If has another name please change it to REASON.

E.G. ALTER TABLE outmigration

CHANGE OUTMGREASO REASON **VARCHAR**(**99**);

Usually this column has a numeric value, please update to the correspondent String value.

E.G. update outmigration set REASON='OTHER' where REASON='6';

Please check that INDIVIDUAL ID is 12 digits (e.g.KWM000087001)

UPDATE outmigration set INDIVIDID =CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

Please check that Visit ID 12 digits (e.g. KWM000087005 = Location ID + round 3 digit))

UPDATE outmigration set OBSERVEID=CONCAT(LEFT(OBSERVEID,**3**),'000',RIGHT(OBSERVEID,**6**)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, OBSERVEID, REGION_NAM, FIELDWORKER, str_to_date(**DATE**,'%Y-%m-%d') **DATE**, REASON, INDIVIDID FROM outmigration WHERE processed_by_mirth =**0**;

Inmigration

Run common SQL-queries.

create indexes:

ALTER TABLE inmigration

ADD INDEX idx_imgind (INDIVIDID);

ALTER TABLE inmigration

ADD INDEX idx_imgvis (OBSERVEID);

REASON should be a column of the inmigration table. If has another name please change it to REASON.

E.G. ALTER TABLE inmigration

CHANGE IMGREASON REASON **VARCHAR**(**99**);

Usually this column has a numeric value, please update to the correspondent String value.

E.G. update inmigration set REASON='OTHER' where REASON='6';

Please check that INDIVIDUAL ID is 12 digits (e.g.KWM000087001)

UPDATE inmigration set INDIVIDID =CONCAT(LEFT(INDIVIDID,**3**),'00',RIGHT(INDIVIDID,**7**));

Please check that Visit ID 12 digits (e.g. KWM000087005 = Location ID + round 3 digit))

UPDATE inmigration set OBSERVEID=CONCAT(LEFT(OBSERVEID,**3**),'000',RIGHT(OBSERVEID,**6**)) ;

If all worked ok then the following query should work correctly and all the columns should have a value.

SELECT id, REASON, str_to_date(**DATE**,'%Y-%m-%d') **DATE**, INDIVIDID, REGION_NAM MOVED_FROM , OBSERVEID, INTERNAL, LOCATIONID, FIELDWORKER FROM inmigration where processed_by_mirth=**0**;

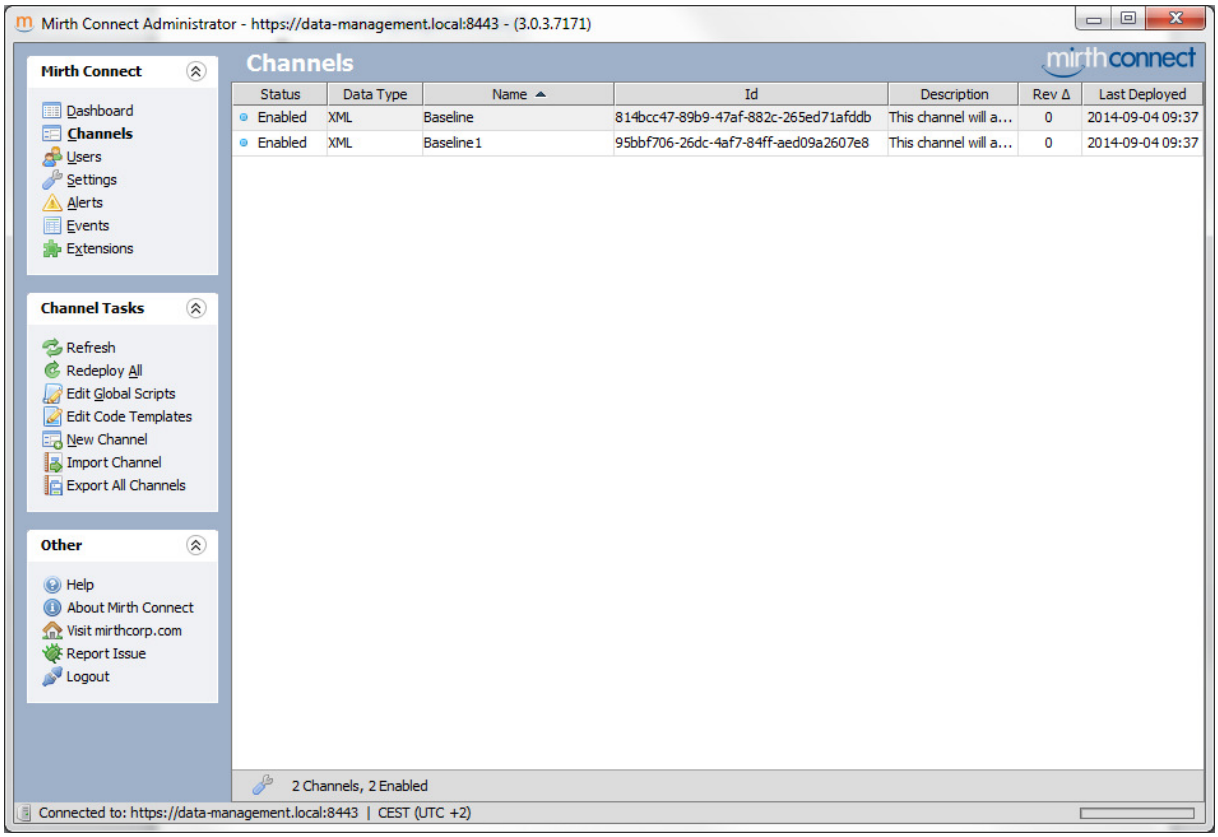
Data import into OpenHDS

Location Hierarchy should be defined and present on OpenHDS.

Create table errors on the HRS database:

```
CREATE TABLE `errors` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `CHANNEL` varchar(30) DEFAULT NULL,
  `DATA` varchar(500) DEFAULT NULL,
  `ERROR` varchar(250) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=119 DEFAULT CHARSET=latin1;
```

Open Mirth Connect Administrator.



Import the appropriate Mirth Channel (Migration Channel) that handles the migration process and the 'database error writer' channel.

Configure the connection to openhds DB and application, the HRS db connection in the global script.

Import the code template.

Run the channels.

After the records are processed you can export the error list by table just running the following query;

E:G: select CHANNEL tableName, DATA, ERROR FROM errors where CHANNEL='Visit';

Here you can see the error ant fix them on the HRS table if possible.

All the record not processed will have the flag processed_by_mirth=2 (errored).

If you fix the error and reset the flag to 0 (unprocessed) and re-run the channel, Mirth will try to re-process the record and if the correction was done correctly the record will go to openhds and the flag will be updated to 1 (transferred).

Please after you export the data and fix it **DON'T forget to delete the data from the error table. Otherwise on next export your list of errors will show them again even if processed.**

So after fixing please run on the HRS DB:

E.G. delete from errors where CHANNEL='Visit';

OpenHDS virtual server setup

In order to lower the entry barrier for parties interested in the evaluation of the software platform, we have compiled an all-in-one package based on virtualization technology, which removes the need for a full-blown IT infrastructure to test the basic features. The following sections describe the setup of this system on a regular PC or Laptop computer.

OpenHDS virtual server setup

Download and install Virtualbox (https://www.virtualbox.org/wiki/Downloads). Download the openhds_server.ova (https://www.wuala.com/swisstph-phc/openhds/) and import this into your local Virtualbox host. This is an Ubuntu 12.4 system, with all necessary components of openHDS installed (openDHS server, ODKAggregate, MirthConnect, openHDS mobile), and initialized as a functional openHDS server.

Make sure the network settings of the virtual machine enable you to connect from your computer. You may need to adapt the network settings in the virtual machine manager, and/or the network configuration of the Ubuntu system (which is set to the static IP address 192.168.56.102). The configuration of the test data generator (below) assumes that the server is accessible under the name data-management.local. Either change the configuration of the test data generator, or map this name to the IP of the virtual server. On a Windows 7 host system, this can be done by adding the following line to the file “c:\Windows\System32\drivers\etc\hosts”

192.168.56.102 data-management.local

All services on the server are running after booting the image. The openHDS database comes pre-populated with a synthetic test data set.

URLS:

- Tomcat is available under http://data-management.local:8080/

- openHds : http:// data-management.local:8080/openhds/

- ODKAggregate : http://data-management.local:8080/ODKAggregate/

The mirth connect administrator tool can be reached from the host system via: https:// data-management.local:8443 / (create exception)

From there you can start the java webstart app. It will fail due to insufficient permissions, so we'll need to add this url to a whitelist. Control Pane -> Java -> Security -> Edit Sitelist -> Add -> Insert https://data-management.local:8443 -> Ok -> Ok.

Most necessary users (Linux, mysql, mirthConnect Admin) are called data, with password data. The openHDS administrative user is called admin, with password test. This image is intended purely for development and testing, THE VIRTUAL MACHINE SHOULD NOT BE ACCESSIBLE ON A SHARED NETWORK INTERFACE AT ANY TIME.

openHDS mobile

An emulated version of openHDS mobile is already packaged with the virtual box server image (the guest system) for testing and evaluation purposes, which is sufficient for most evaluation purposes. Launch the emulator (TODO: work out an intuitive way to launch this). All the configuration settings and software is ready to connect to the openHDS server components.

Test data generation

https://github.com/SwissTPH/openhds-sim/ is a Python program that emulates a tablet computer with ODKCollect installed, and which can be used to generated test data sets in openHDS. The Python program is also pre-installed on the virtual image. Not that you can also use the openhds-sim software to test-drive your local OpenHDS server installation. Install the software by following these setup instructions.

Setup

- Download latest Fieldworker-Simulator from https://github.com/SwissTPH/openhds-sim
  e.g. 'wget https://github.com/SwissTPH/openhds-sim/archive/master.zip'
- 'sudo apt-get update'
- Download zip-program 'sudo apt-get install zip'
- Extract zip file with 'zip master.zip'

Installing dependencies

- install 'sudo apt-get install --no-install-dependencies python-mysqldb'
- install 'sudo apt-get install --no-install-dependencies python-numpy'
- install 'sudo apt-get install --no-install-dependencies python-matplotlib'
- install 'sudo apt-get install --no-install-dependencies python-lxml '

Run fieldworker-simulator

- Change folder 'cd openhds-sim-master'
- python fieldwork_simulator.py

The software comes with configuration files which allow the simulation of a baseline round, and 2 update rounds, and therefore can be used to generate a small test data set. At this stage, the events reported are not necessarily meaningful, but they are meant to be compatible with the consistency checks imposed by openHDS.

Start the main program fieldwork_simulator.py: Optionally edit the configuration file which is stored under /home/data/ openhds-sim/conf/site.json, and start the script by double-clicking on the shortcut "fieldwork_simulator" on the

Events submitted for the baseline round are now transferred from ODKCollect to openHDS. Once this process finishes, the data generator will submit events for the subsequent rounds (and pause again once this is over), until all events have been transferred.

The test data set can then be viewed in openHDS.