

OpenHDS manual

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

Contents

Introduction.....	5
Administrator manual	5
Hardware requirements.....	5
Installing Minimal Ubuntu server.....	5
Installing Java 7.....	6
Installing MySQL Server.....	6
Installing Apache2, PHP and PHPMyAdmin	6
Set up MySQL-user and databases.....	7
Add user and grant access rights in MySQL cli	7
Add user and grant access rights in PhpMyAdmin.....	8
Optional: Grant outside access to MySQL.....	10
Installing Tomcat	11
Installing MySQL-J Connector.....	11
Installing SSH-Server.....	12
Installing OpenHDS.....	12
Deploying OpenHDS in Tomcat	12
Set up OpenHDS required data	13
Changing the Language in OpenHDS.....	14
Code and Parameters.....	16
Installing Mirth	17
Installation steps	17
Set-up Mirth Connect to use the MySQL database.....	18
Importing MirthConnect channels	19
Installing ODKAggregate.....	21
Run the installer	21
Installation.....	22

Upload HDSS Core XLSForms.....	23
Create Database and Views for Data Management.....	25
Datamanager MySQL User	26
Custom Location hierarchy.....	27
Tablet setup.....	29
Set-up OpenHDS Mobile.....	29
Set-up ODKCollect	32
Synchronizing OpenHDS Mobile.....	35
Synchronizing ODKCollect	38
User manual	41
Field worker manual.....	41
Getting Started	41
Changing Language in OpenHDS Mobile	42
Field Worker Login	43
Creating New Location	44
Language Selection.....	46
Tablet workflows	46
Baseline Visit	46
Follow-up Visit.....	49
Data manager manual	66
Fieldworkers creation.....	66
Collection of non-core variables in (“Extra forms”)	66
Extra-form data integration	67
XLSForm.....	67
Extra Form execution order.....	68
Export Extra Form submissions	70
Error Handling	73
Data flows in OpenHDS	74
Migration of legacy data to OpenHDS.....	75
Requirements	76
File Conversion and staging HDSS database.....	76
Required tables for migration	77
Adding required attributes.....	78
Common steps for all tables.....	78
Steps for each table.....	78

Round	78
Location	79
Observation (maps to Visit).....	80
Individual	81
Socialgroup	82
Relationship.....	83
Membership	84
Pregnancy observation (indvstatus).....	86
Pregnancy outcome (pregoutcome)	86
Birth.....	87
Residency.....	87
Death.....	88
Outmigration	89
Inmigration	90
Error table.....	91
Data import into OpenHDS.....	91
Transformation using MirthConnect.....	91
Mirth-Channels.....	93
Set up an alert for errors on Extra-Forms in Mirth:	94
Migrating from older version of openhds SQL ALTER script	95
OpenHDS evaluation platform	96
OpenHDS in VirtualBox.....	96

Revision history:

Document Version	OpenHDS Release	Date	Modified by	Changes
1.0	1.2	01.10.2014	Swiss TPH	Initial version of OpenHDS manual
1.1	1.2	19.11.2014	MTR	Added sections on how to create MySQL user and Mobile application installation and set-up. Other small changes.
1.2	1.2	23.11.2014	MTR	Extended section on how to deploy the Amazon EC2 OpenHDS Evaluation platform
1.3	1.2	23.11.2014	NMA	Added section on data flows and error resolution
1.4	1.3	27.11.2014	ADP	Expanded Section on Extra forms and section on Task generation before Syncing the tablets.
1.5	1.3	02.12.2014	ADP	Expanded error handling section. Added Update visit workflow.
1.6	1.3.2	15.05.2015	ADP	Updated Login mobile section.
1.7	1.3.3	6.8.2015	Swiss TPH	Added section on configuring location hierarchy Various updates due to software prerequisite changes
1.8	1.4 exp	11.08.2015	MTR	Added section about extra-form functionality
1.9	1.4	22.09.2015	NMA	Updated URLs
1.10	1.4	6.10.2015	NMA	Minor fixes and updates
1.11	1.5	31.03.2016	MTR	Extended Sync OpenHDS Mobile and Extra form section
1.12	1.5	27.7.2016	NMA	Added installation instructions for Windows
1.13	1.5	3.8.2016	NMA	Removed EC2 instructions

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.

OpenHDS was developed by University of Calabar, Nigeria; University of Southern Maine, US; and Ifakara Health Institute, Tanzania; and first deployed in Akpabuyo HDSS, Cross River, Nigeria. All groups continue contributing to the development of OpenHDS.

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 inside the Tomcat servlet container. We will also cover how to install and set-up the mobile apps used for mobile data collection, in particular OpenHDS Mobile and ODKCollect.

For the purpose of documenting the setup of OpenHDS, we use the same usernames, passwords and hostnames as in the evaluation platforms described in section “OpenHDS evaluation platforms”. Specifically, we describe how to 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. The standard username for system services is “data”, with password “data”.

It is highly recommended, for production systems even essential, that you change the logins used in this guide, and set password which comply with security standards!

Hardware requirements

Server hardware requirements depend on the size of data to be managed, and on the rate of data acquisition. The size of the surveyed population and the age of the HDSS site are the most important determinants of the hardware requirement. For a typical site with 10'000s of individuals which have been under surveillance for some years, the following server specification was found to be adequate:

CPU: Intel Xeon E5-2609

Memory: 8GB RDIMM, 1600MT/s

Storage: 900GB, SAS 6Gbit/s

For the data collection, tablet computers running the Android OS are required. The minimal supported version of Android is 4.4.2 (API 19).

Installing Minimal Ubuntu server

The recommended server operating system for running OpenHDS is Linux. Here we document the setup on the Basis of the Ubuntu Linux distribution. Windows as a server operating system is currently not supported. Instructions for setting up the software pre-requisites on a Windows server

are provided at the end of the following sections where applicable. However, so far there is very limited experience using OpenHDS on Windows servers.

- Download the **Ubuntu Mini** ISO from
<https://help.ubuntu.com/community/Installation/MinimalCD>
Ubuntu 14.04 "Trusty Tahr" LTS Minimal CD 64-bit PC (amd64, x86_64) is the recommended version for production systems ([the size is approximately 37 MB](#)). The 32-bit PC (x86) Ubuntu 14.04 "Trusty Tahr" LTS Minimal CD, is used for the evaluation platform.
- Install the operating-system with no localization options and don't install any packages.
- During installation, set the hostname of the machine to '**data-management.local**'. If you missed to change this during setup, you can change the hostname afterwards by editing the file 'etc/hostname'.

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 the command 'sudo apt-get install openjdk-7-jre-headless' from a terminal.
- **Optional:** 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).

Windows: Install Java as documented here: <http://docs.oracle.com/javase/7/docs/webnotes/install/>

Installing MySQL Server

- Install the MySQL server with 'sudo apt-get install mysql-server'.
- Follow the set-up instructions during installation and set the password of the root user during installation to '**data**', leave the rest to its default.

Windows: Follow the instructions provided here:

<http://dev.mysql.com/doc/refman/5.7/en/windows-installation.html> to install the MySQL server

Installing Apache2, PHP and PHPMyAdmin

This step is only required if you want to administer the MySQL database with a graphical user interface. If you are familiar with the mysql Command Line Interface (CLI) and the MySQL syntax, you don't can also insert all the statements from the command line.

- Run the command 'sudo apt-get install phpmyadmin' in a terminal
- This command will install the phpmyadmin package. This package depends on other packages to be installed (e.g. apache2 and php), and it will also install all the missing packages if they are not available on the system (which is the case if you followed this guide).
- During the installation choose MySQL as the database and say yes to set-up the config through dbconfig-common.
- Set the root password to 'test'
- Set up a password for the PhpMyAdmin-application (e.g. 'test')
- PhpMyAdmin is now accessible under <http://data-management.local/phpmyadmin>

Windows: You can skip this step on a Windows server

Set up MySQL-user and databases

Add user and grant access rights in MySQL cli

Open a Terminal:

Log in:

- ‘mysql –uroot –pdata’ this will open the MySQL CLI, login in as user root with password test
- The MySQL command prompt should appear

Create user:

- ‘CREATE USER ‘data’@‘%’ IDENTIFIED BY ‘data’;’ this command will create a new MySQL user data with the password data.

Create databases:

- CREATE DATABASE IF NOT EXISTS ‘openhds’;
- CREATE DATABASE IF NOT EXISTS ‘odk_prod’;

Grant access privileges to user:

- ‘GRANT ALL ON *.* TO ‘data’@‘%’;’ this command will grant the user data access to all databases with all access rights
- flush privileges;

**(YOU SHOULD CHANGE THE SPECIFIC ACCESS RIGHT FOR A PRODUCTION SERVER ENVIRONMENT
TO SOMETHING STRICTER!)**

Add user and grant access rights in PhpMyAdmin

Open the PhpMyAdmin site <http://data-management.local/phpmyadmin>

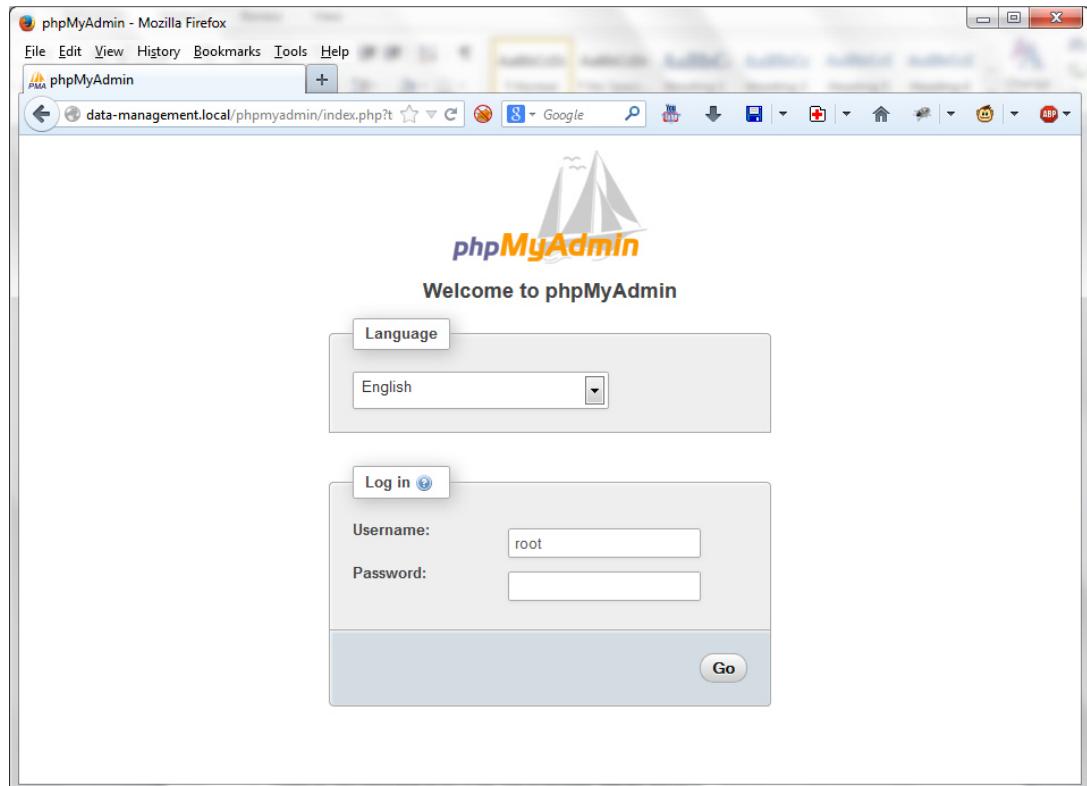


Figure 1

Log in as user 'root' with password 'test'.

From the main menu, switch to the User section

A screenshot of the PhpMyAdmin interface showing the "Users overview" section. The left sidebar lists databases: "dubai", "hrs_mysql_db", "information_schema", "iSHARETraining", "mirthdb", "mysql", "odk_prod", "openhds", and "performance_schema". The main panel shows a table of users with the following data:

User	Host	Password	Global privileges	Grant	Action
data	%	Yes	ALL PRIVILEGES	Yes	
datamanager	%	Yes	USAGE	No	
debian-sys-maint	localhost	Yes	ALL PRIVILEGES	Yes	
mirthuser	%	Yes	USAGE	No	
mirthuser	localhost	Yes	USAGE	No	
odk_user	localhost	Yes	USAGE	No	
root	127.0.0.1	Yes	ALL PRIVILEGES	Yes	
root	::1	Yes	ALL PRIVILEGES	Yes	
root	data-management	Yes	ALL PRIVILEGES	Yes	
root	localhost	Yes	ALL PRIVILEGES	Yes	

Figure 2

From there, click on ‘Add user’ (Refer to Figure 2 bottom). A screen like in Figure 3 should appear.

The screenshot shows the 'Add user' form in phpMyAdmin. In the 'Login Information' section, the 'User name:' field is set to 'data', 'Host:' is 'Any host', and 'Password:' and 'Re-type:' fields both contain 'data'. There is a 'Generate password:' button and a 'Generate' link. In the 'Database for user' section, there are two unchecked checkboxes: 'Create database with same name and grant all privileges' and 'Grant all privileges on wildcard name (username_%)'.

Figure 3

Set the ‘User name’ to ‘data’ and the password fields to ‘data’. Under Global privileges click on ‘Check All’ (refer to Figure 4).

The screenshot shows the 'Global privileges' section in phpMyAdmin. At the top, there is a checkbox for 'Grant all privileges on wildcard name (username_%)' which is unchecked. Below it, a red box highlights the 'Global privileges (Check All / Uncheck All)' button. The interface is divided into three main sections: 'Data' (with checkboxes for SELECT, INSERT, UPDATE, DELETE, and FILE), 'Structure' (with checkboxes for CREATE, ALTER, INDEX, DROP, and other structure-related permissions), and 'Administration' (with checkboxes for GRANT, SUPER, and various administrative privileges). A note at the bottom states: 'Note: MySQL privilege names are expressed in English'.

Figure 4

Create the new user and permissions by clicking on 'Go'.

Create databases in PHPMyAdmin

Switch to the Databases section. Under 'Create database' input the database name (openhds and then odk_prod) and click on create to create the database (refer to Figure 5).

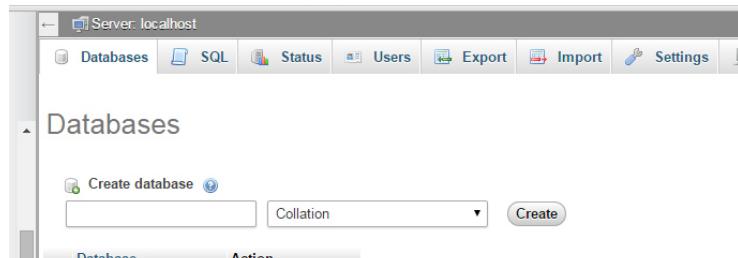


Figure 5

Optional: Grant outside access to MySQL

This step is optional. You can use this to access the MySQL Server instance from outside the server (non-localhost), so e.g. you can use Fat Clients like the MySQL Workbench Application to administer the MySQL database instead of PhpMyAdmin.

- Edit /etc/mysql/my.cnf and either remove or comment out line starting with *Bind-Address* with a '#'
- Restart the MySQL-service with 'sudo service mysql restart'

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 it is not already running.
- 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** (open the file e.g. with sudo vi tomcat-users.xml).
- Add a new role *html-gui* and a user in this role to above mentioned file in the <tomcat-users> section:

```
<role rolename="manager-gui" />
<user username="data" password="data" roles="manager-gui" />
```
- 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 stored under **/var/log/tomcat6/** (if needed)

Recommended:

Increase Tomcat Memory allocation.

When encountering huge amounts of data (e.g. Individuals data), Tomcat runs out of memory (since it defaults to use only 128MB). It is thus advised to increase the amount of memory that Tomcat can use.

To do this, edit the configuration in 'etc/default/tomcat6'. Find the line starting with 'JAVA_OPTS=' , and add the values '-Xmx1024M -Xms1024M'. After saving the file, restart the Tomcat server with 'sudo service tomcat6 restart' to apply the change.

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 to your Ubuntu home (e.g. with wget)
- extract the file (in Ubuntu: 'tar -xvzf mysql*.tar.gz')
- and chmod the mysql*.jar file to 777 or just executable

OR AS AN ALTERNATIVE TO DOWNLOADING FROM MYSQL.COM (UNDER UBUNTU):

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

THEN:

- 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'

Windows: Follow the instructions on <https://tomcat.apache.org/tomcat-6.0-doc/setup.html#Windows> to install Tomcat 6. Then download mysql-connector-java-* .zip from <http://dev.mysql.com/downloads/connector/j/>, and copy the extracted jar file to the lib directory in your Tomcat installation directory.

Installing SSH-Server

The SSH server installation is a good way to remotely access the server.

- Install ssh-server with: 'sudo apt-get install -y openssh-server'
- Service option: sudo service ssh status | start | stop | restart
- Now you will be able to remotely connect to your server over SSH for example with Putty which can be downloaded from <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Installing OpenHDS

A *.war file is a Web Application Archive. This is the format that OpenHDS comes packaged in when you download our binary distribution. It is ready to be deployed to the Tomcat Servlet Application Server. Ways to obtain the OpenHDS Web Application are:

- 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/openhds>

In both cases, you will need to make sure that the database connection parameters (hostname, user, password) are set correctly for your server. To do this, open the OpenHDS.war file with an archiving program like 7-zip, and edit the file WEB-INF/classes/database.properties. The values you are most likely to adapt are for the variables dbURL, dbUser, and dbPass.

Now you are ready to deploy the war file through the Tomcat html manager.

Deploying OpenHDS in Tomcat

Open the Tomcat manager under <http://data-managment.local:8080/manager> and log in as data/data.

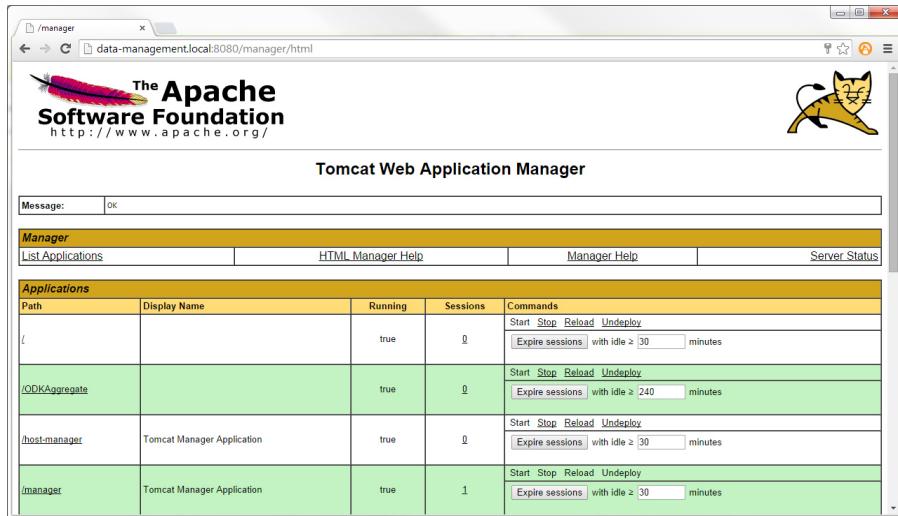


Figure 6

Upload and submit *openhds.war* file.

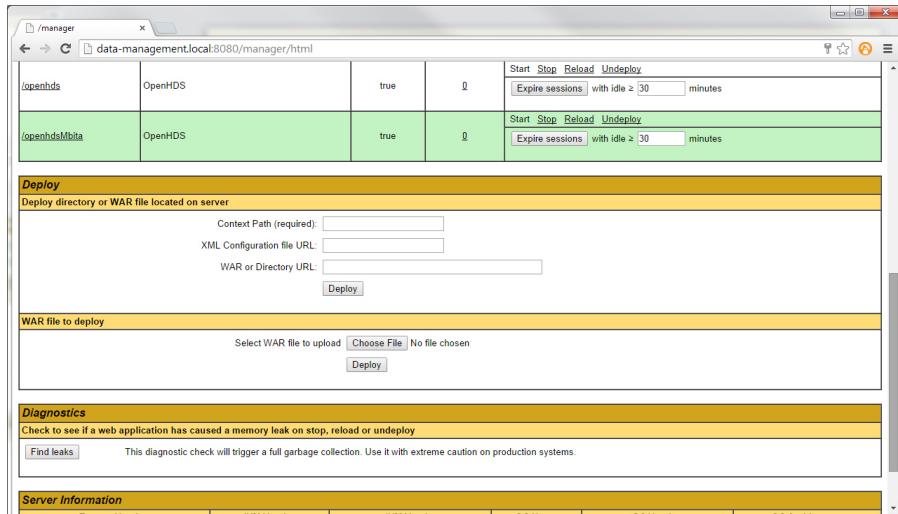


Figure 7

Check if the application package was successfully deployed and is running. If the openhds application is not listed as “running”, check the tomcat log file.

Set up OpenHDS required data

After you have deployed the OpenHDS application you will need to run one additional SQL script that will insert some required data into the openhds-database, including a default administrator login (**admin/test**).

Find the SQL-script ‘openhds-required-data.sql’ which can be found in the openhds.war-file (open with 7zip, WinRar or equivalent) under WEB-INF\classes\.

If you are using phpMyadmin, log in as user data. Then select the openhds database and select the SQL section. Copy/paste the contents of the SQL file into the ‘Run SQL queries’ window and submit with ‘Go’.

The screenshot shows the phpMyAdmin interface for the 'openhds' database. The left sidebar lists tables such as 'adultvpm', 'AsyncTask', 'death', 'demrates', 'extension', 'fieldworker', 'form', 'individual', 'immigration', 'location', 'locationhierarchy', 'locationhierarchylevel', 'membership', 'neonatalvpm', 'note', 'outcome', 'outmigration', 'postneonatalvpm', and 'pregnancyobservation'. The main area is titled 'Run SQL query/queries on database openhds:' and contains the following SQL script:

```

1 INSERT INTO `user` (uuid,firstname,lastname,description,username,password,IsActive) VALUES ('Administrator User', 'admin', 'test', 0, false);
2 INSERT INTO `user_roles` (user_uuid, role_uuid) VALUES ('User 1', 'ROLE1');
3 
4 -- Location Hierarchy root
5 INSERT INTO `locationhierarchy` (uuid, name, extId, level_uuid, parent_uuid) VALUES ('hierarchy_root', 'Root', NULL, NULL, NULL);
6 
7 -- Field Worker
8 INSERT INTO `fieldworker` (uuid, extId, firstname, lastname, deleted) VALUES ('UnknownFieldWorker', 'UNKNOWNFIELDWORKER', 'Unknown', 'Field Worker', false);
9 
10 -- Unknown Individual: This should always be pre-populated
11 INSERT INTO `individual` (uuid, extId, firstName, middleName, lastName, gender, dob, mother_uuid, father_uuid, inserted, updated) VALUES ('Unknown Individual', 'UNK', 'Unknown', 'NULL', 'UNKNOWN', 'MALE', '1900-12-19 15:07:43', '15:07:43', 'PENDING', 'NULL', 'NULL', false, 'UnknownFieldWorker');
12 
13 INSERT INTO `whitelist` (uuid, address) VALUES ('LOCALHOST1', '127.0.0.1');
14 
15 INSERT INTO `whitelist` (uuid, address) VALUES ('LOCALHOST2', 'localhost');

```

Below the SQL editor, there are buttons for 'Clear', '[Delimiter :]', 'Show this query here again' (checked), 'Retain query box', and a 'Go' button.

Figure 8

Under the Ubuntu terminal you can extract the war file with ‘unzip openhds.war’. Then issue a ‘mysql –udata –pdata openhds < /path/to/sql/file/openhds-required-data.sql’ to run the SQL script.

Changing the Language in OpenHDS

The OpenHDS server platform is localized in several languages. At the moment, the supported languages include English, French, Portuguese and Swahili.

To switch the language the OpenHDS Application is displayed in:

1. Click on the ‘Select language’ link on either the login page or from the top of the menu once logged in (see Figure 9).

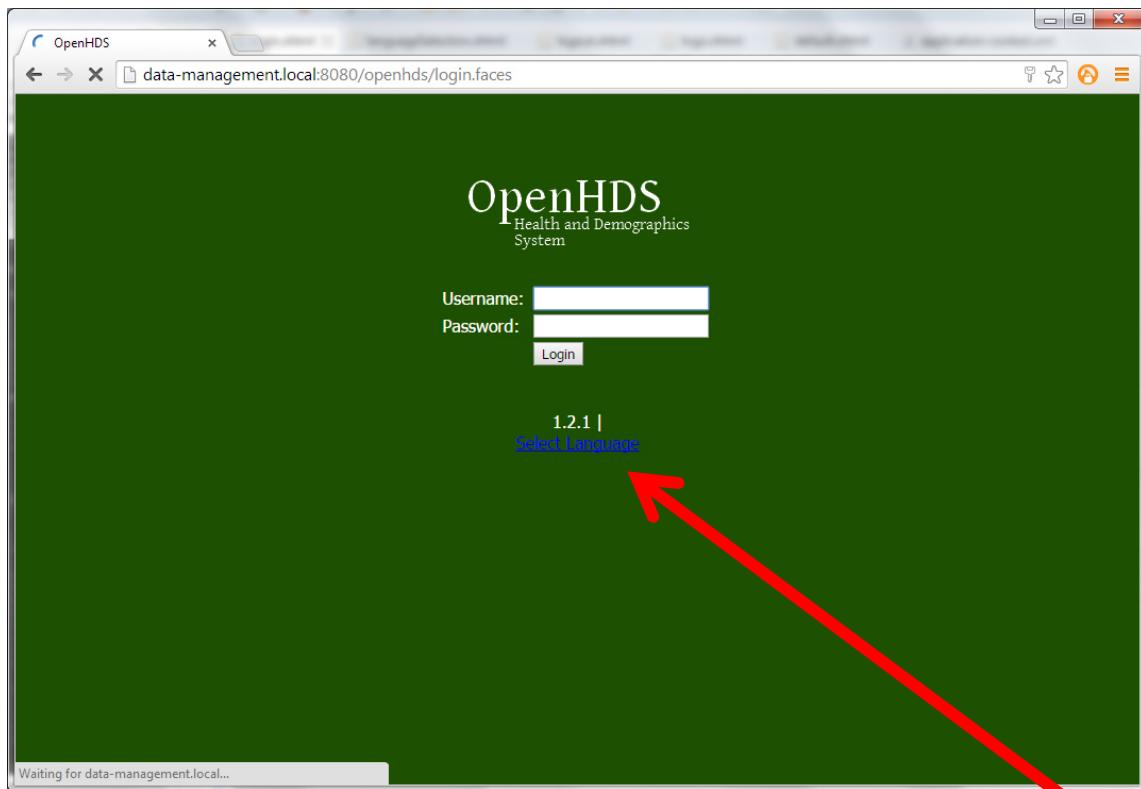


Figure 9

2. On the Internationalization page, select your display language and click on 'Change Locale' to save the settings (refer to Figure 10).

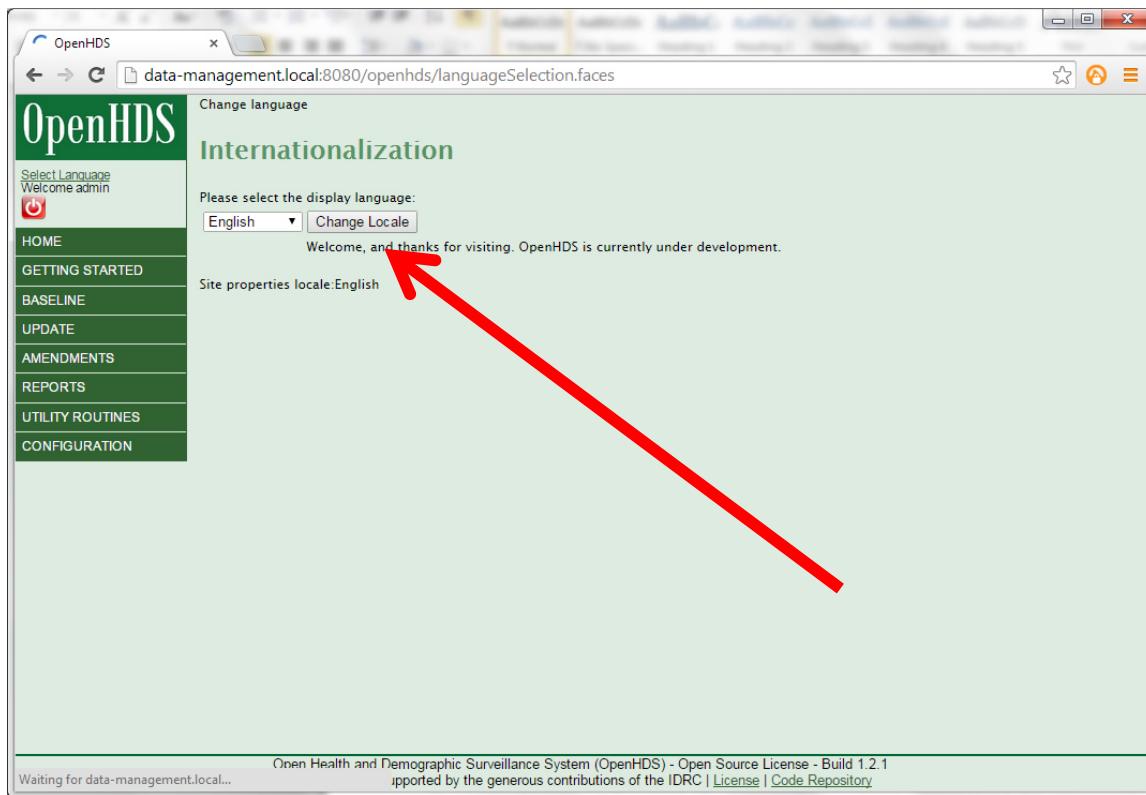


Figure 10

Code and Parameters

In the configuration section of OpenHDS we need to choose the code and parameters used in the system (Figure 11).

The following parameters can be set:

- ✓ Unknown Identifier:
- ✓ Not Applicable:
- ✓ Male:
- ✓ Female:
- ✓ Yes:
- ✓ No:
- ✓ Birth:
- ✓ Death:
- ✓ In Migration:
- ✓ Out Migration:
- ✓ Enumeration:
- ✓ Marriage:
- ✓ Live Birth:
- ✓ Still Birth:
- ✓ Miscarriage:
- ✓ Abortion:
- ✓ Data Status Warning:
- ✓ Data Status Valid:
- ✓ Data Status Fatal:
- ✓ Data Status Void:
- ✓ Data Status Pending:
- ✓ Data Status Closed:
- ✓ Locale:
- ✓ Date Format:
- ✓ Use Autocomplete:
- ✓ Visit At:
- ✓ Minimum Age of Parenthood:
- ✓ Minimum Age of Household Head:
- ✓ Minimum Age of Marriage:
- ✓ Minimum Age of Pregnancy:
- ✓ Earliest Enumeration Date (dd-MM-yyyy)
- ✓ Earliest Event Date (dd-MM-yyyy)

Those parameters then will be used in the web application and in the OpenHDS mobile.

Especially important for the data collection are the following parameters:

1. Visit At: this parameter can have the two values “location” or “Socialgroup”. These two values basically are the two standards between East Africa (visit at location level) or West Africa (visit at Social Group level). When the tablet it’s synced this parameter will influence

the logic in the tablet. The Visit ID will be Location ID or SocialGroup ID + round Number, and the individuals shown from the tablet will be the one of the location selected or the one of the social group visited depending on the visit level.

2. The Minimum Age parameters all will be considered in the tablet logic (e.g. a child cannot be chosen as household head, and a woman of less than n years cannot have a pregnancy, etc)
3. Earliest Event Date: a not compulsory field. This if set, can be used in the forms if we want to force a lower limit to dates of events (e.g Migration can not be before the start of the study or of a specific round etc)

Figure 11

Installing Mirth

MirthConnect is used to transfer data into OpenHDS by making web service calls. This is used for submitting data during the baseline and update rounds, but also for ingesting legacy data from a system like HRS2. Make sure you have an updated Java with Java Web Start to run the Mirth Connect Administrator Application.

- 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 the *Mirth Connect Administrator* from above site
- Log in with admin/admin
- Create an admin account (data/data) - no need to register.
- You can start/stop/etc the Mirth Connect service by running 'sudo service mcservice start|stop|status' (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 data@'%' IDENTIFIED BY 'data' WITH GRANT OPTION;
```

Now open the file 'mirth.properties' in / usr/local/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'.

Relevant lines to change

```
# options: derby, mysql, postgres, oracle, sqlserver

database = mysql

# examples:
# Derby      jdbc:derby:${dir.appdata}/mirthdb;create=true
# PostgreSQL  jdbc:postgresql://localhost:5432/mirthdb
# MySQL       jdbc:mysql://localhost:3306/mirthdb
# Oracle      jdbc:oracle:thin:@localhost:1521:DB
# SQLServer   jdbc:jtds:sqlserver://localhost:1433/mirthdb

database.url = jdbc:mysql://localhost:3306/mirthdb

# if using a custom driver, specify it here
#database.driver =

# maximum number of connections allowed for the connection pool
database.max-connections = 10

# database credentials
database.username = data
database.password = data
```

- Afterwards restart the mirth connect service (sudo service mcservice restart)
- After the restart, the new database schema will be created and you can log in to the Mirth Connect Administrator with **data/data**

Windows: Download and install the MirthConnect software from <https://www.mirth.com/Downloads>. Launch MirthConnect Server Manager from Start Menu. Stop

the service (on the “Service” panel). Change the database type to mysql: select “mysql” in the dropdown menu on the “Database” panel, and provide the mysql connection credentials. Start the service again on the “Service” panel.

Importing MirthConnect channels

Download the latest release of the channel definitions, code templates, global scripts, and alerts from the OpenHDS source repository: <https://github.com/SwissTPH/openhds>. Using the MirthConnect Administrator application, import these resources into your MirthConnect environment.

To do that launch Mirth Connect Administrator and log in.

Click on Channels in the menu.

On the “Channel Task” in the Menu then click on “Import Channel” and select the relevant Channel you want to import and when the channel is open click on the “Save Changes” on the channel task list.

If you are running new DSS and you need to start with a “Baseline scenario” you will import the following channels:

- Baseline
- Baseline Households

If you are already on a running DSS (Update Rounds) you will import the following channels:

- Update Events
- Update Household

In both scenario then you need to import also the channels:

- Database error writer
- File error create send

After you finish importing all the channels and saving them, you need to click on “Edit Global Scripts” on the “Channel Tasks” and then click “Import Scripts”.

You’ll be prompted to choose a script. Select the file Global Scripts.xml.

You’ll get to the following screen where you need to configure your database connection parameters and a local path to a folder where to store the “error file” produced by mirth.

```

1 // This script executes once when all channels start up from a redeploy
2 // You only have access to the globalMap here to persist data
3
4
5 // odk information
6 globalMap.put("ODK_DATABASE_URL", "jdbc:mysql://localhost/odk_prod");
7 globalMap.put("ODK_DATABASE_USER", "data");
8 globalMap.put("ODK_DATABASE_PASSWORD", "data");
9
10
11
12 // openhds information
13 globalMap.put("OPENHDS_PATH", "http://localhost:8080/openhds/api/test");
14 globalMap.put("OPENHDS_USER", "admin");
15 globalMap.put("OPENHDS_PASSWORD", "test");
16 globalMap.put("ERROR_PATH", "home/users/Desktop/mirth/output/errors.csv");
17 globalMap.put("OUTPUT_PATH", "home/users/Desktop/mirth/output");
18
19
20 // odk tables
21 globalMap.put("DEATH_TABLE", "DEATH_REGISTRATION_CORE");
22 globalMap.put("IN_MIGRATION_TABLE", "IN_MIGRATION_CORE");
23 globalMap.put("LOCATION_TABLE", "LOCATION_REGISTRATION_CORE");
24 globalMap.put("MEMBERSHIP_TABLE", "MEMBERSHIP_CORE");
25 globalMap.put("OUT_MIGRATION_TABLE", "OUT_MIGRATION_REGISTRATION_CORE");
26 globalMap.put("PREGNANCY_OBS_TABLE", "PREGNANCY_OBSERVATION_CORE");
27 globalMap.put("PREGNANCY_OUTCOME_TABLE", "PREGNANCY_OUTCOME_CORE");
28 globalMap.put("OUTCOMES_TABLE", "PREGNANCY_OUTCOME_OUTCOMES");
29 globalMap.put("RELATIONSHIP_TABLE", "RELATIONSHIP_CORE");
30 globalMap.put("SOCIAL_GROUP_TABLE", "SOCIAL_GROUP_REGISTRATION_CORE");
31 globalMap.put("VISIT_TABLE", "VISIT_REGISTRATION_CORE");
32 globalMap.put("BASELINE_TABLE", "BASELINE_CORE");
33
34 return;
35
36
37
38
39
40
41
42
43
44
45

```

After you have configured the variables values then you need to click on the Script Tasks Menu the item “Save Scripts”.

Then click again on Channels and then on “Edit Code Templates” in the Channel Tasks menu.

After done this please click on “Import Code template” and choose the file “CodeTemplates.xml”.

Click then on Save code templates.

Next step is to click on “Alerts” on the Mirth connect Menu.

In the Alert Tasks menu click on “Import Alert” menu item and select the file “Events Connection Alert.xml”.

Double click on the “Events Connection Alert” and on the channel panel on the right select the channels “Baseline” and “Baseline Households” (if in Baseline Scenario) or “Update Events” or “Update Households” (if in Update rounds scenario) and click on the enable button.

Save the alert (Ignore any eventual warning message).

Last step is to configure the SMTP connection on the channel “File Error CreateSend”.

Go on the Channels list and double click on the “File Error CreateSend” channel.

Select the “Destinations” tab. You’ll see the following screen.

Please check with your IT department the SMTP host.

In the fields “From” you will enter the sender email.

In the fields “To” you will enter the emails of the people that need to receive the error file generated by mirth. You can have multiple addresses comma (,) separated.

Installing ODKAggregate

ODKAggregate is the server component that acts as a repository for electronic forms used in the data collection (which can be synchronized to tablets), and as the recipient and storage for completed forms which are submitted from tablets.

ODKAggregate provides an installer that will create the war-file with settings preconfigured according to your settings during the installation steps. The installer is available for Windows, Linux 32 or 64 and also Mac OS. This guide will assume that you download and run the Linux installer.

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

- Download the appropriate (32 or 64 bit, depending on your system) Linux installer from <http://opendatakit.org/downloads/> (~260MB). If you’re under Ubuntu, you can issue a ‘wget <http://url.to/file/filename.sth>’ to download the file
- Change the installer permissions with ‘chmod 777 ODK[...].run’. This will set the permissions for the installer such that it is allowed to be executed.
- Run the installer with ‘./ODK[...].run’

Run the installer

The ODKAggregate installer will ask for user input to set up the database connection and other configuration options.

- Put the installer into e.g.: ~/ (the tilde stands for the user home, e.g. /home/data)

- Set the output folder to `~/ODK`
- The port you can leave at 8080
- The database server is hosted at 127.0.0.1
- When asked for the public internet address, insert ‘data-management.local’
- When asked for the SQL user, use: `data / data` (not recommended for production)
- Db name: `odk_prod`
- ODKAggregate instance name: `odk_prod`
- Configuration

Installation

The installer ran in the previous step will create a new folder called ‘ODKAggregate’ in the install folder that you supplied in the first step. You will find the `ODKAggregate.war` file there and an SQL script `create_db_and_user.sql`.

- Open PhpMyAdmin and run theSQL script to generate the odk database.
- Deploy the `ODKAggregate.war` with the Tomcat manager.
- Start the ODKAggregate application in Tomcat manager if it is not already started.
- Navigate to '`http://data-management.local:8080/ODKAggregate/`' in a web-browser to browse to ODKAggregate
- You should now be able to log in by using ‘Aggregate Password’ with the username you supplied in the installer and the password ‘`aggregate`’. You can later change the password by going to the ‘Site Admin’ section.

Figure 12 ODKAggregate ‘Site Admin’ section

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/)

Upload HDSS Core XLSForms

This section covers the steps to obtain and upload the OpenHDS core forms, which the OpenHDS application needs for its core functionality.

Download the OpenHDS Core Forms from <https://github.com/SwissTPH/openhds> (xlsx format). These forms can be modified if required, for example by adding translation to additional languages, or variables which are collected beyond the core set (however, those additional variables are not automatically integrated with the OpenHDS database). The form templates must be converted into the XForms format prior to upload them into ODKAggregate.

To convert the xlsx file into a proper Xform there are several options. We suggest the following two:

- <http://opendatakit.org/xiframe/> (online conversion)
- <https://github.com/UW-ICTD/xlsform.exe/blob/master/README.md> (windows offline converter)

Open up <http://data-management.local:8080/ODKAggregate/> in your Web browser.

You can log in with the local ODK account if required, but you shouldn't need it to upload Forms, since by default it should be allowed for anonymous users.

	start	end	deviceld	migrationType	locationId	visitId	fieldWorkerId	individualInfo	individualInfo	individualInfo	individualInfo	individualInfo
X	2014-08-27 17:17:35.0		14:F4:2A:87:12:30	BASELINE	IFA000008	IFA000008001	data	IFA000008001	UNK	UNK		
X	2014-08-27 17:16:22.0		14:F4:2A:87:12:30	BASELINE	IFA000007	IFA000007001	data	IFA000007003	UNK	UNK		
X	2014-09-03 18:14:17.0		14:F4:2A:87:12:30	BASELINE	IFA000001	IFA000001001	data	IFA000001001	UNK	UNK		
X	2014-09-03 18:16:23.0		14:F4:2A:87:12:30	BASELINE	IFA000001	IFA000001001	data	IFA000001002	UNK	UNK		
X	2014-09-17 11:46:50.0		14:F4:2A:87:12:30	BASELINE	IFA000001	IFA000001001	data	IFA000001001	UNK	UNK		
X	2014-09-17 12:00:06.0		14:F4:2A:87:12:30	BASELINE	IFA000001	IFA000001001	data	IFA000001002	UNK	UNK		
X	2011-06-06 2011-06-06 05:48:58.0 06:47:13.0		8c:77:12:5b:c1:3c	BASELINE	KIU000001	KIU000001000	FWDG5	KIU000001001	UNK	UNK		
X	2011-06-06 2011-06-06 07:43:40.0 08:18:56.0		8c:77:12:5b:c1:3c	BASELINE	KIU000001	KIU000001000	FWDG5	KIU000001002	UNK	UNK		
X	2011-06-06 2011-06-06 05:40:43.0 06:30:26.0		8c:77:12:5b:c1:3c	BASELINE	KIU000001	KIU000001000	FWDG5	KIU000001003	UNK	UNK		
X	2011-06-06 2011-06-06 05:43:44.0 05:56:40.0		8c:77:12:5b:c1:3c	BASELINE	KIU000001	KIU000001000	FWDG5	KIU000001004	UNK	UNK		
X	2011-05-27 2011-05-27 07:52:06.0 08:00:37.0		8c:77:12:5b:c1:3c	BASELINE	IFA000002	IFA000002000	FWNF4	IFA000002001	UNK	UNK	S	
X	2011-05-27 2011-05-27 10:47:38.0 11:46:55.0		8c:77:12:5b:c1:3c	BASELINE	IFA000002	IFA000002000	FWNF4	IFA000002002	UNK	UNK	F	
	2011-05-27 2011-05-27											

Figure 13

Switch to the 'Form Management' tab.

Title	Form Id	Media files	User	Downloadable	Accept Submissions	Publish	Export	Delete
Baseline registration	baseline	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Connect Adult	connect_adult	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Death registration	death_registration	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Death to Head of household	DEATHTOHOH	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
In migration registration	in_migration	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Location registration	location_registration	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Membership registration	membership	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Out migration registration	out_migration_registration	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Pregnancy Outcome	pregnancy_outcome	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Pregnancy observation registration	pregnancy_observation	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Relationship registration	relationship	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Social Group Registration	social_group_registration	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Visit Registration	visit_registration	0	anonymousUser	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			

Figure 14

Click on add new form.

Xform Upload

Upload one form into ODK Aggregate

Form definition (xml file): No file selected.

Optional Media file(s): No files selected.

Media files for the form's logo, images, audio clips and video clips (if any) should be in a single directory without subdirectories.

NOTE: If the form definition contains string answers the string data will be truncated to 255 characters. See ODK Aggregate 1.0 documentation for how to increase (or decrease) this size.

On ODK Collect 1.1.7 and higher, the file named "form_logo.png", if present in the media folder, will be displayed as the form's logo.

Figure 15

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'.

Upload the other forms, one by one.

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 into OpenHDS.

After logging in as user **root** select the **odk_prod** database and execute following SQL script:

Error table and views

```
CREATE TABLE `errors` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `CHANNEL` varchar(30) DEFAULT NULL,
  `DATA` varchar(1000) DEFAULT NULL,
  `ERROR` varchar(280) DEFAULT NULL,
  `exported` int(1) DEFAULT '0',
  `inserted_timestamp` timestamp DEFAULT CURRENT_TIMESTAMP,
  `COMMENT` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 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;
CREATE VIEW DEATHTOHOH_VIEW AS SELECT * FROM DEATHTOHOH_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 errors 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'@'%';
GRANT SELECT, UPDATE ON DEATHTOHOH_VIEW TO 'datamanager'@'%';
# BASELINE ROUND PERMISSIONS
GRANT SELECT, UPDATE ON BASELINE_VIEW TO 'datamanager'@'%';

```

Datamanager MySQL User

The above script will also create a MySQL user ‘datamanager’ with the password ‘dataODKmanager’. This special user only has access to these error views and error table. With these he can modify and reset the erroneous data. Refer to section ‘Error handling’ in the Data Management section below for more details.

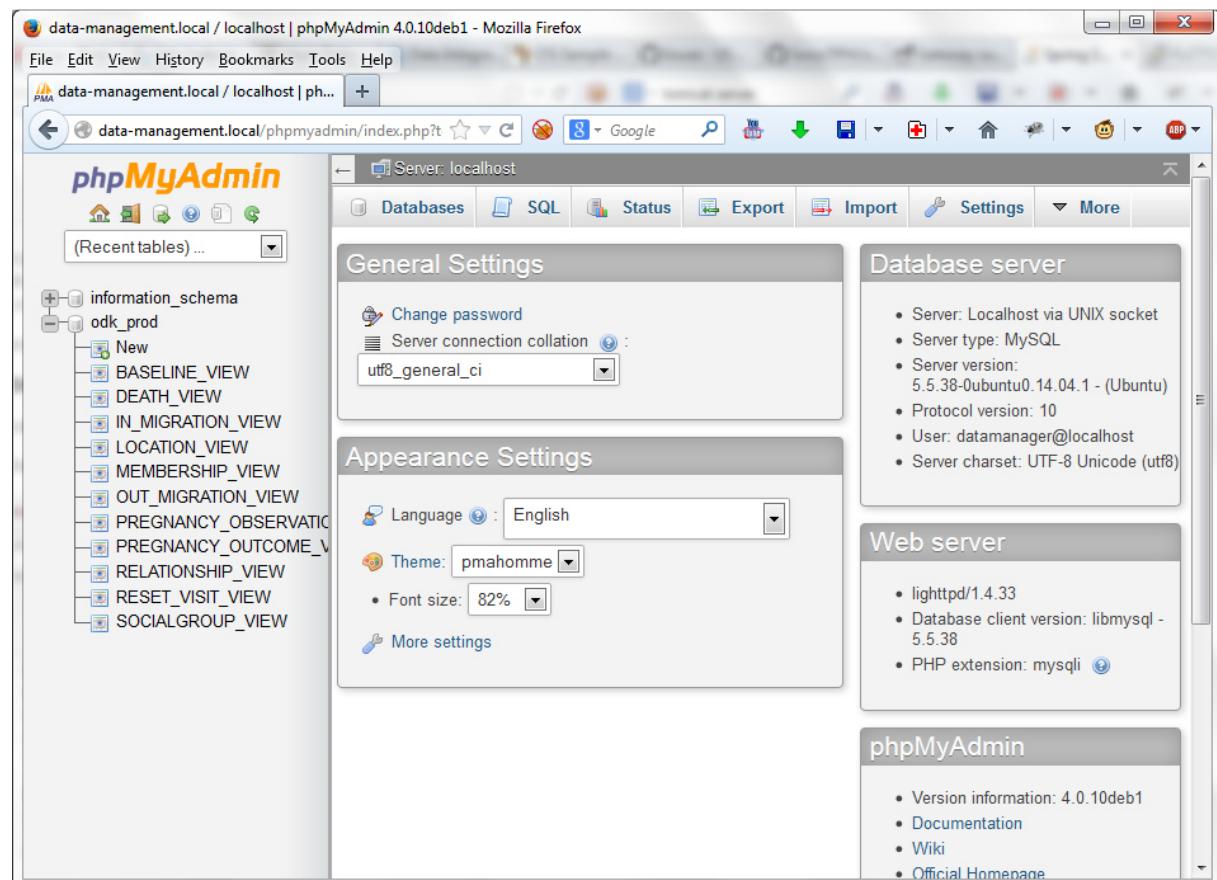


Figure 16 The datamanager MySQL user only has access to the error views

Custom Location hierarchy

By default OpenHDS has a location hierarchy that starts at 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. Then navigate to 'Configuration' and finally to 'Location Levels'.

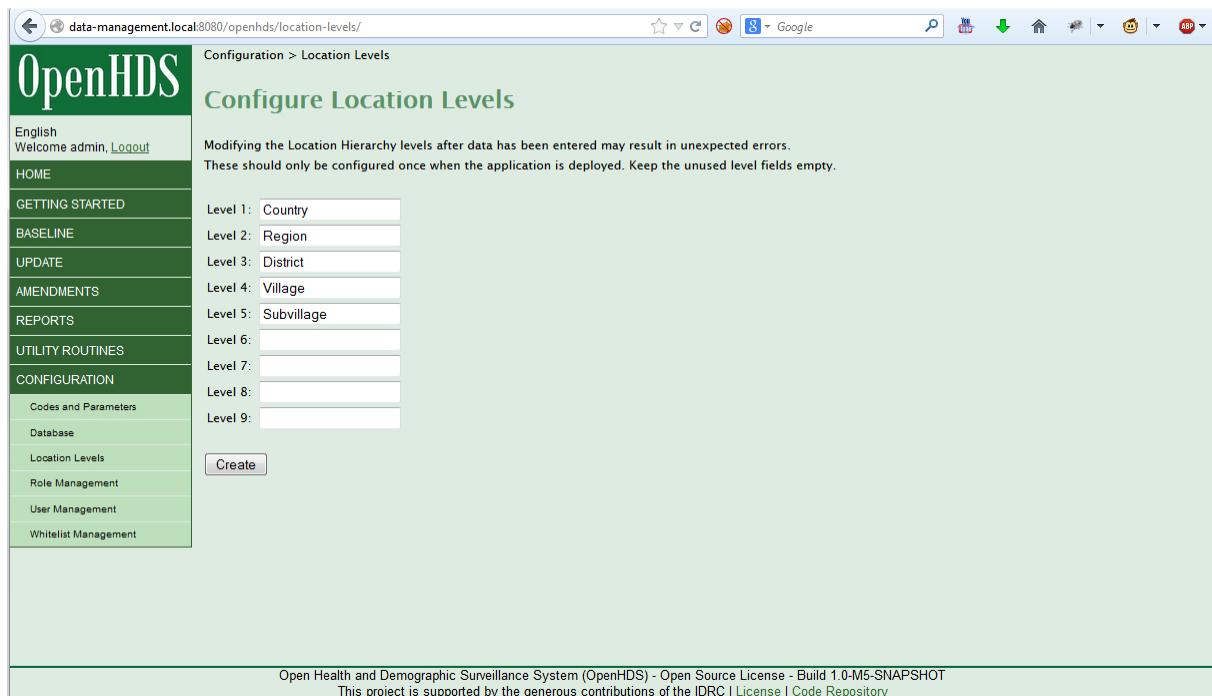


Figure 17

The OpenHDS system can manage up to 9 levels in the location hierarchy.

To change the labels edit the Level entries and then save with the button 'Create'.

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

Next you need to populate the location hierarchy. Navigate to Utility Routines->Location Hierarchy and select Create after filling in a location. Start at the top of the hierarchy (e.g. country), and work your way down to build a hierarchy tree by specifying the parent of each location (e.g. the village a sub-village belongs to).

IMPORTANT:

Note that the definition of your location hierarchy forms the basis of a standard ID naming convention which is used throughout in all OpenHDS systems. IDs for the core entities of the HDSS data base are constructed as follows:

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...)

Once you have changed the location Levels, you'll also need to update the OpenHDS mobile application to handle these changes.

Re-syncing the tablet (see section Tablet setup) through the sync database function will bring the new levels automatically in the mobile application. Buttons will be shown or hidden in the tablet according to the configuration on the server

Tablet setup

- Download the OpenHDS apk from <https://github.com/SwissTPH/openhds> . If you should into installation troubles, make sure that Installation from untrusted sources is enabled in the settings under security.
- Download ODKCollect (from the Google Play Store or from <https://opendatakit.org/downloads/download-info/odk-collect-apk/>).
- Install OpenHDS mobile and ODKCollect

Set-up OpenHDS Mobile

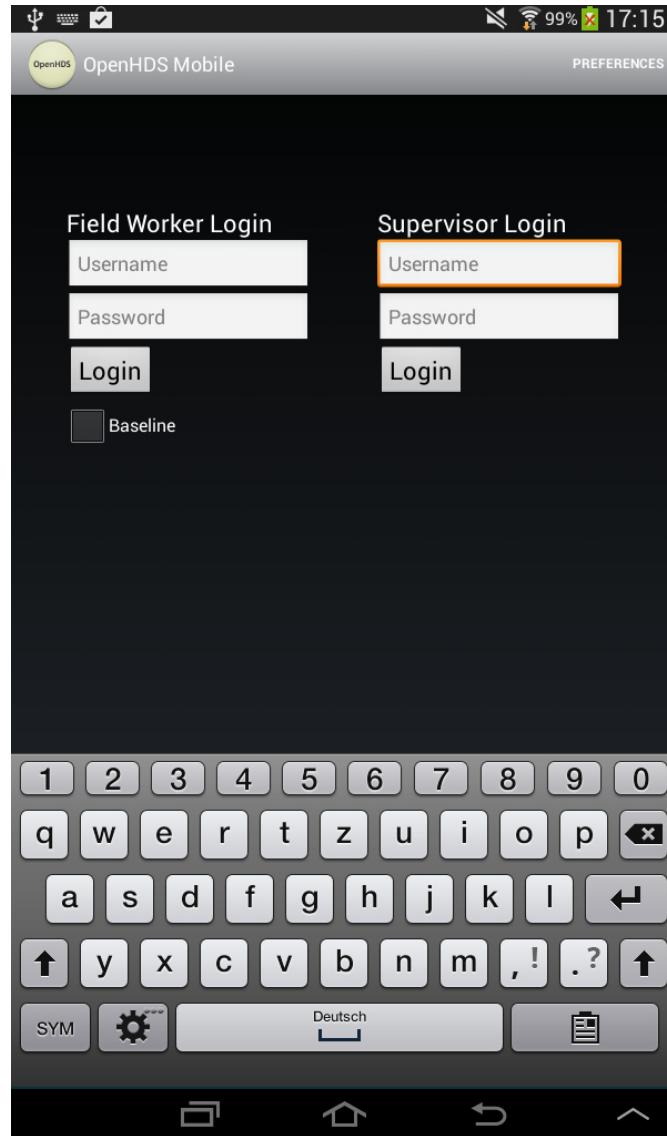


Figure 18

Tap the «*Preferences*» button at the upper right hand corner

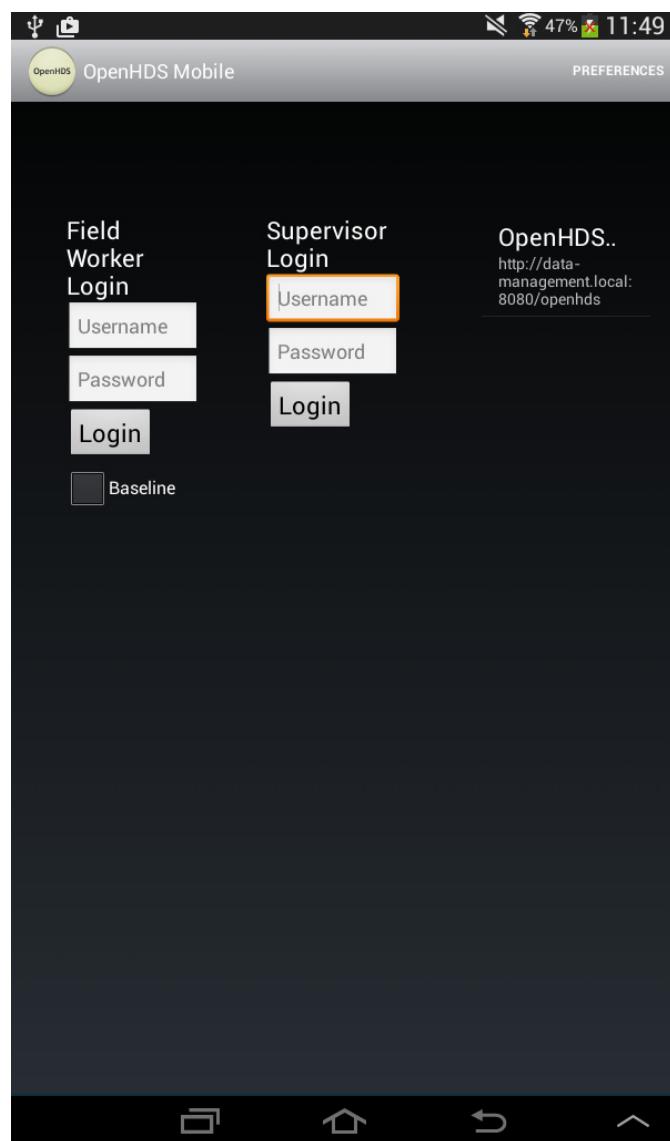


Figure 19

Click on the URL to start editing the entry

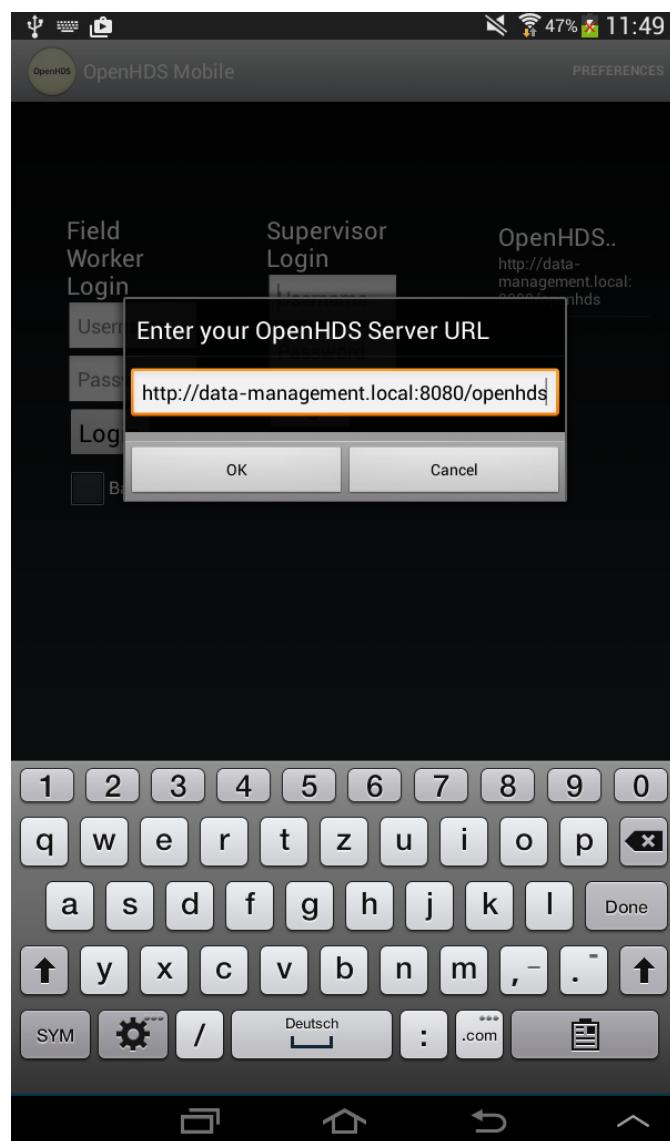


Figure 20

Enter the URL to the OpenHDS server and confirm your input with a click on «OK»

Set-up ODKCollect

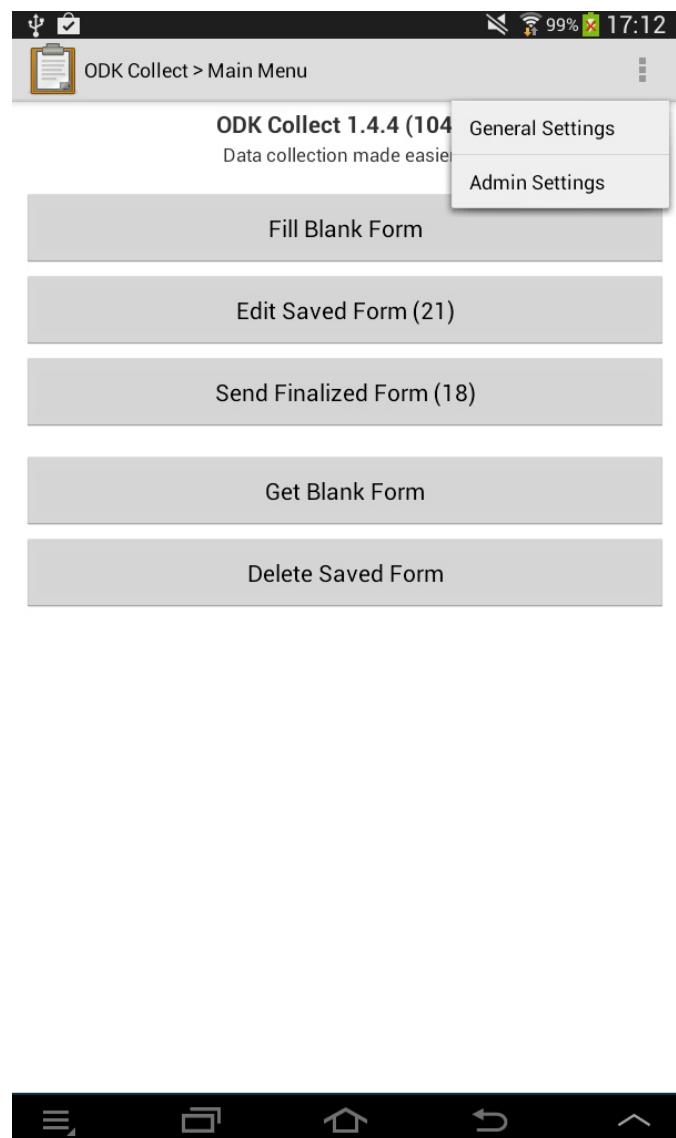


Figure 21

Open the Menu at the upper right hand corner (or tap the hardware menu button on your device respectively) and select «*General Settings*»

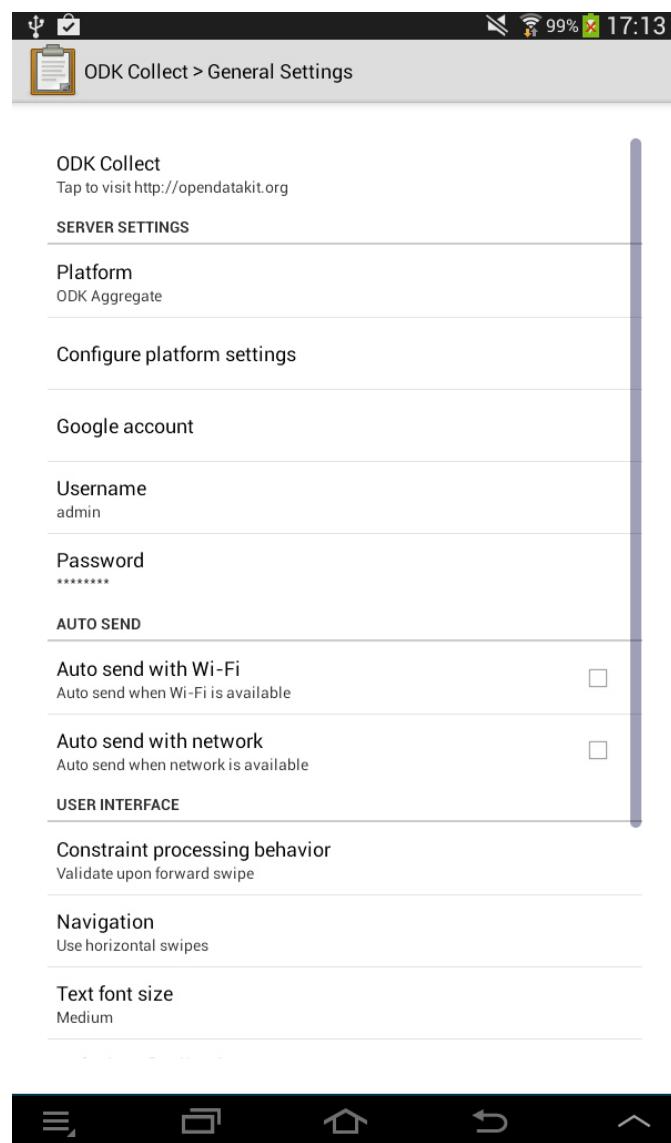


Figure 22

Select the option «*Configure platform settings*»

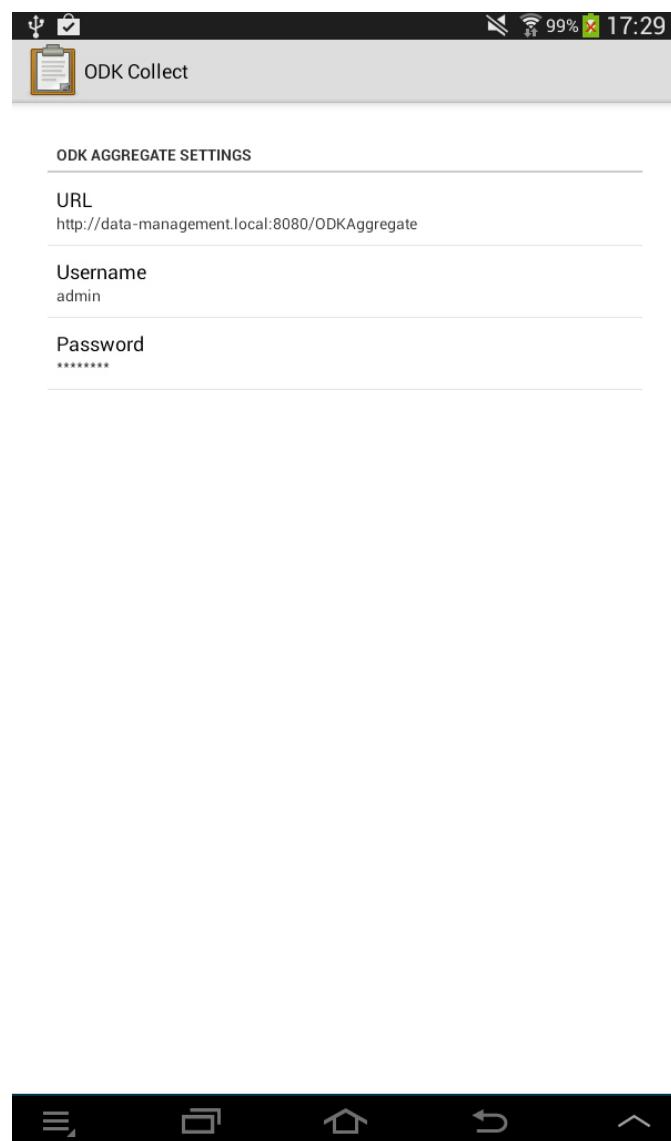


Figure 23

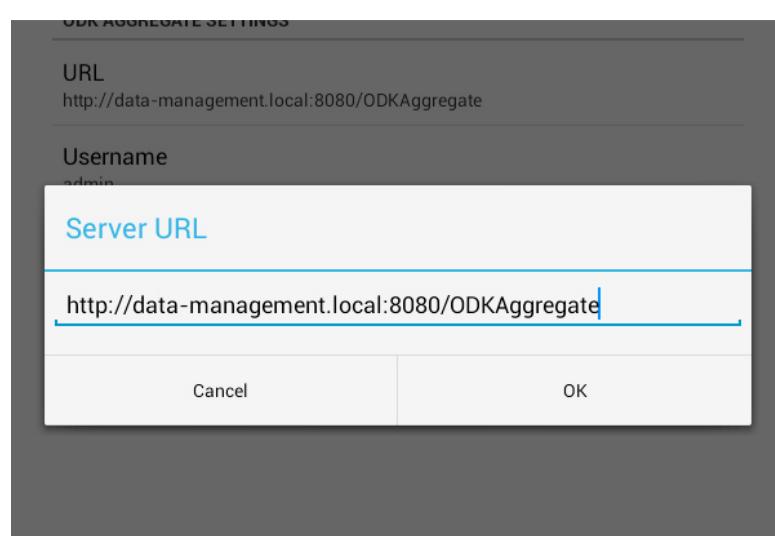


Figure 24

Enter the URL to the ODKAggregate server and also provide a valid ODKAggregate Username/Password combination.

Synchronizing OpenHDS Mobile

NOTE:

Before you can start the synchronization with OpenHDS Mobile, make sure you have set-up OpenHDS as described in the section “Set up OpenHDS”.

Before starting syncing the tablets it is mandatory to prepare the data on the server to be synced to the tablet.

So the following steps must be executed on OpenHDS server (Web interface):

- 1) Login as admin to OpenHDS server.
- 2) Click on the menu Item Utility Routines → Round Codes
 - a. Create the round (0= Baseline, 1= First follow up etc. etc.)
- 3) Click on the menu item Utility→Tasks
 - a. In the field Round number enter the number of the round just created and click on Start Visit Task. Wait till you see that the task is ended as in the figure below

Task Name	Total Items Processed	Start Time Stamp	End Time Stamp	MDS Hash
Visit Task	7017	12-09-2014 11:56:02	12-09-2014 11:56:08	e8b49b011e0972b593b27989452460a7

- b. Now Click on the button Start Individual Task and wait till it finish as shown in the following figure.

Task Name	Total Items Processed	Start Time Stamp	End Time Stamp	MDS Hash
Visit Task	7017	12-09-2014 11:56:02	12-09-2014 11:56:08	e8b49b011e0972b593b27989452460a7
Individual Task	33325	25-11-2014 07:31:28	25-11-2014 07:34:26	027e9cb4567270b35f2da7fce5a73582

- c. Click on the button Start Location Task and wait till it finish as shown in the following figure.

Task Name	Total Items Processed	Start Time Stamp	End Time Stamp	MDS Hash
Visit Task	7017	12-09-2014 11:56:02	12-09-2014 11:56:08	e8b49b011e0972b593b27989452460a7
Individual Task	33325	25-11-2014 07:31:28	25-11-2014 07:34:26	027e9cb4567270b35f2da7fce5a73582
Location Task	8747	25-11-2014 07:35:11	25-11-2014 07:35:25	6f85d39a494f19a074150fea9a26a735

- d. Click on the button Start Relationship Task and wait till it finish as shown in the following figure.

Task Name	Total Items Processed	Start Time Stamp	End Time Stamp	MDS Hash
Visit Task	7017	12-09-2014 11:56:02	12-09-2014 11:56:08	e8b49b011e0972b593b27989452460a7
Individual Task	33325	25-11-2014 07:31:28	25-11-2014 07:34:26	027e9cb4567270b35f2da7fce5a73582
Location Task	8747	25-11-2014 07:35:11	25-11-2014 07:35:25	6f85d39a494f19a074150fea9a26a735
Relationship Task	4384	25-11-2014 07:35:30	25-11-2014 07:35:36	ea0619acaddee30d6432eac92317b27bd

- e. Finally Click on the button Start Socialgroup Task and wait till it finish as shown in the following figure.

Task Name	Total Items Processed	Start Time Stamp	End Time Stamp	MDS Hash
Visit Task	7017	12-09-2014 11:56:02	12-09-2014 11:56:08	e8b49b011e0972b593b27989452460a7
Individual Task	33325	25-11-2014 07:31:28	25-11-2014 07:34:26	027e9cb4567270b35f2da7fce5a73582
Location Task	8747	25-11-2014 07:35:11	25-11-2014 07:35:25	6f85d39a494f19a074150fea9a26a735
Relationship Task	4384	25-11-2014 07:35:30	25-11-2014 07:35:36	ea0619acadee30d6432eac92317b27bd
Social Group Task	5461	25-11-2014 07:35:46	25-11-2014 07:35:51	cc5f854bf001a898947600e092cdabbd

The tasks must be executed in the order described.

They are separated because in case you need to amend an individual or a Location or Relationship etc. it is only necessary to re-run only the task involved and not all.

Now you can start with the synchronization process.

In the «*Supervisor Login*» section, provide the user credentials and click on «*Login*»

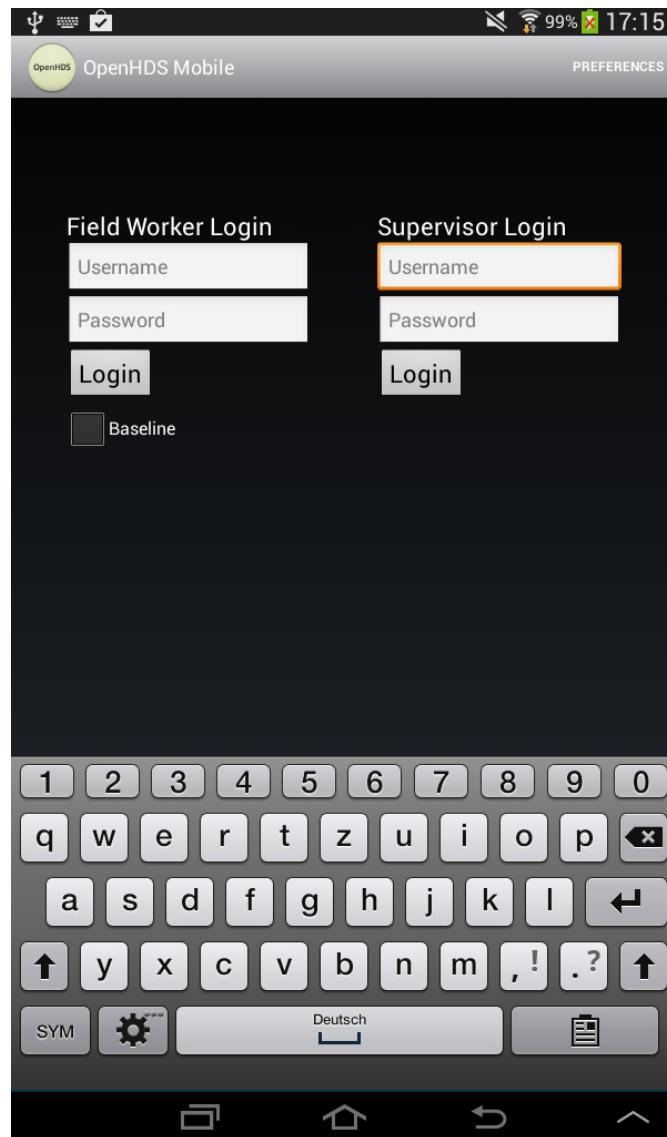


Figure 25

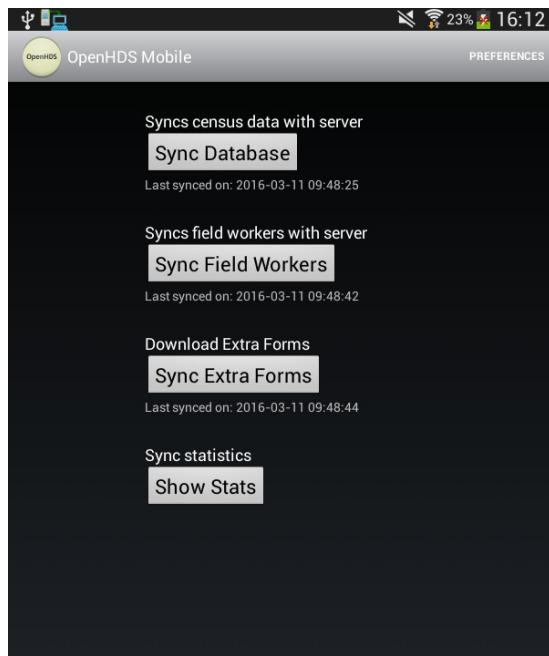


Figure 26

Select «*Sync Database*» to download the OpenHDS Entities onto the tablet

Select «*Sync Field Workers*» to download the OpenHDS Field Workers onto the tablet

Select «*Sync Extra Forms*» to download the Extra Form definitions onto the tablet

After every synchronization step, the “last sync” label will update to reflect the last time of synchronization. In addition, a small pop-up window will display the number of currently stored entities on the table and those available on the server. If this number doesn’t match, are displayed in red, otherwise in green. If entries are incomplete, try re-syncing them again.

Please make sure that all entries are in green before sending out the fieldworker to collect data!



Figure 27 Sync statistics

Synchronizing ODKCollect

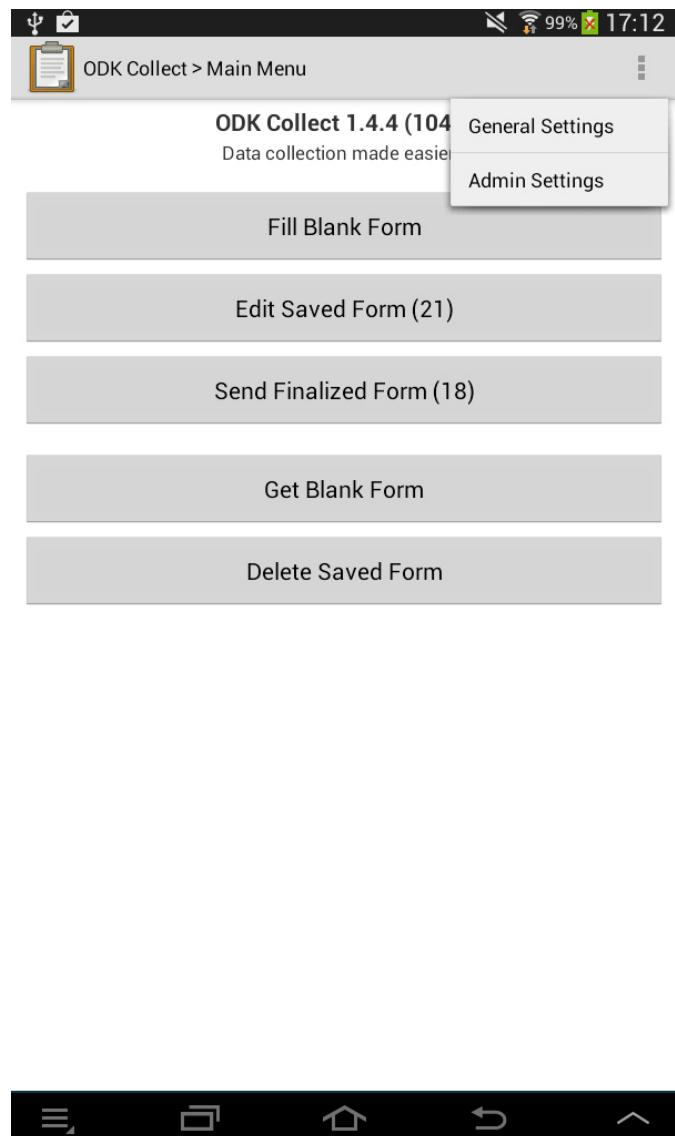


Figure 28

Select «*Get Blank Form*»

ODKCollect will now query for all available Forms directly on the server and display them in an overview

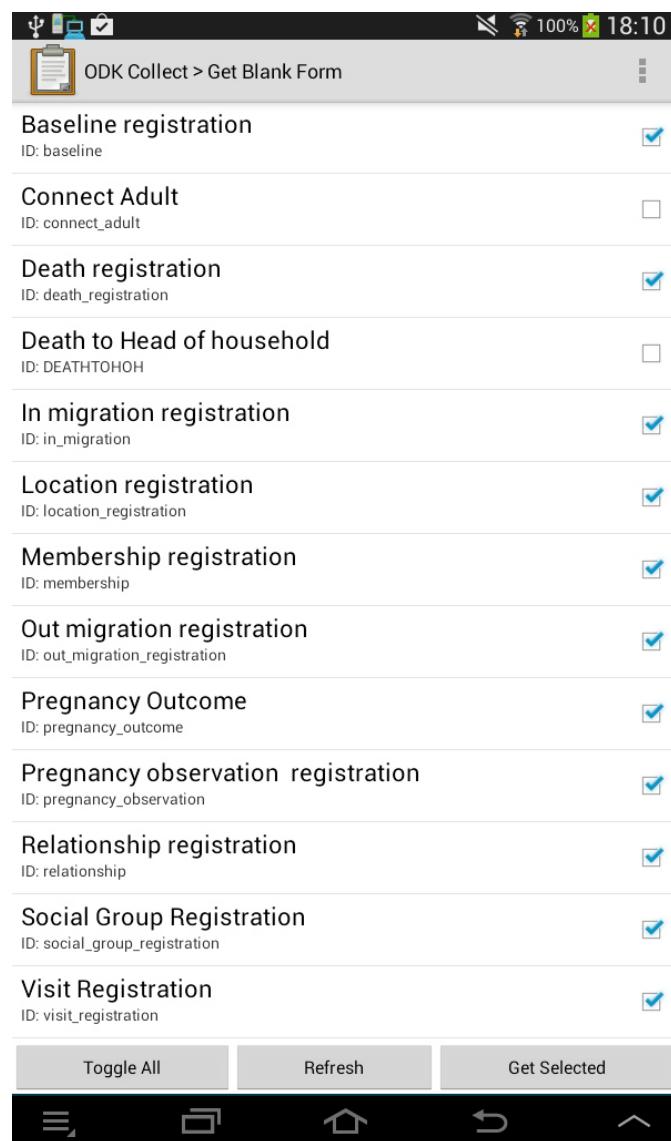


Figure 29

Select the Forms you want to synchronize with the tablet from the list

Click on «*Get Selected*» to start the import process

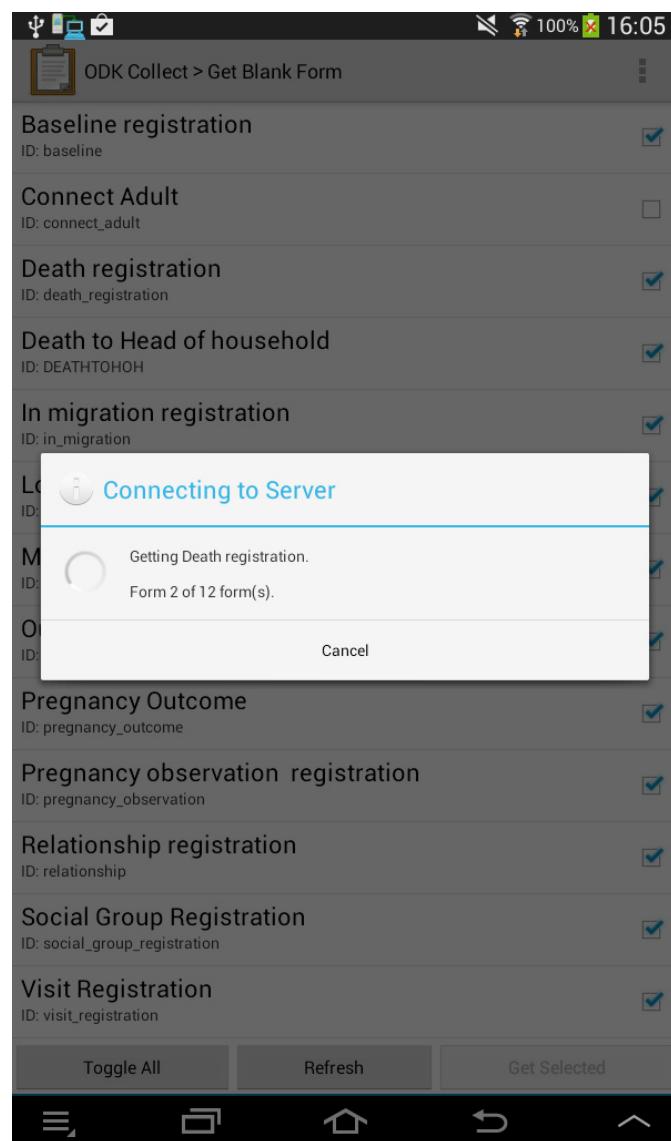


Figure 30

The selected forms are transferred and imported one by one into ODKCollect

The imported forms are displayed. Make sure there were no errors.

User manual

Field worker manual

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

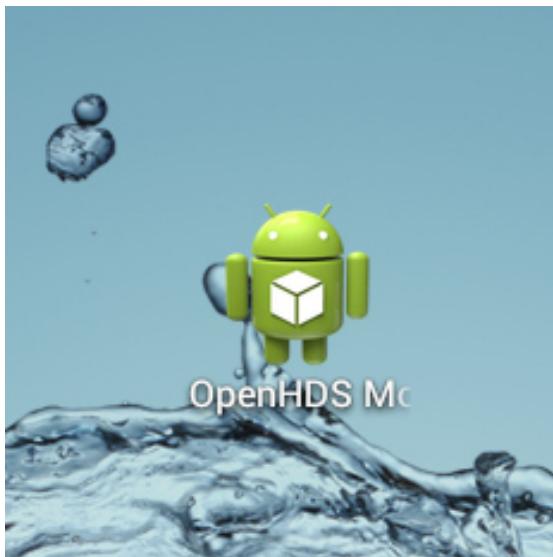


Figure 31

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 31.

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 33**Error!**

Reference source not found. below:

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

Changing Language in OpenHDS Mobile

In OpenHDS Mobile it is also possible to have the interface displayed in the language of your choice. At the moment we have implemented support for these languages:

English, Spanish and Portuguese.

To select a different display language, open the Preferences menu in the login screen and click on Language. In the List select the language that you want the application to be displayed in (Figure 32). The language switch is immediate.

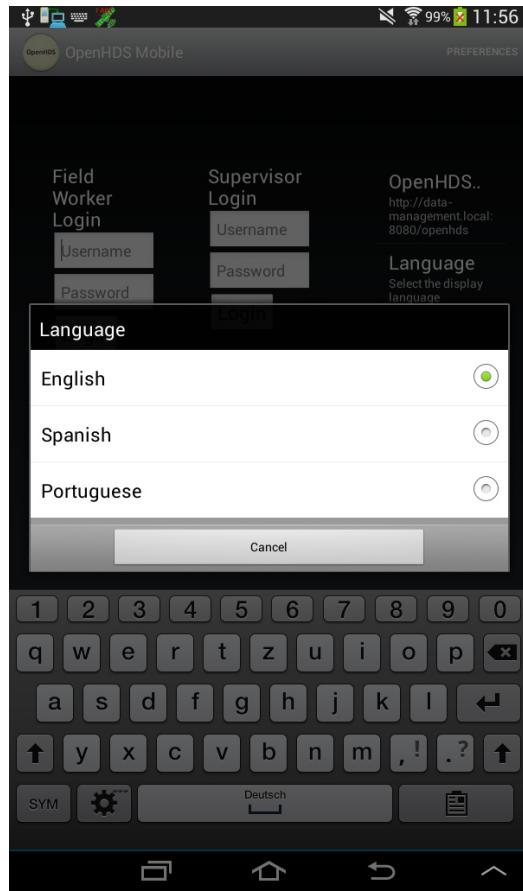


Figure 32

Field Worker Login

Figure 33 is the login page “Field Worker Login”.

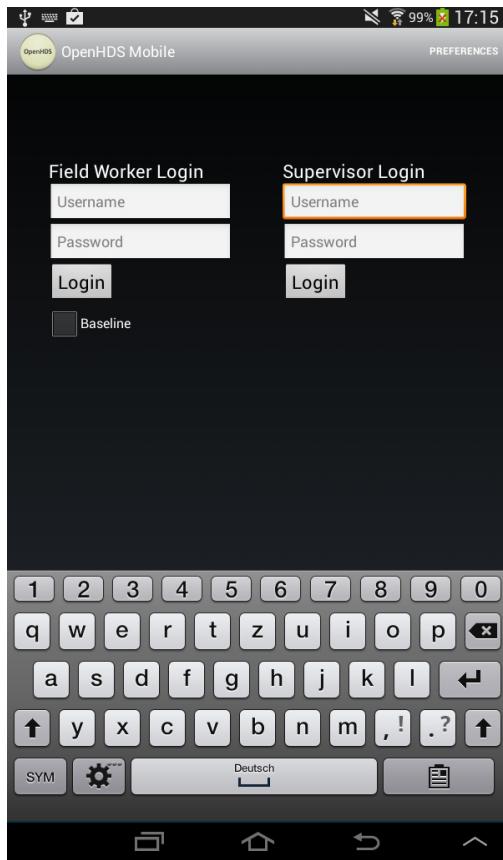


Figure 33

Figure 34 shows the **main menu** display, on the left column is the information about location and individuals while on the right column is event information. 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

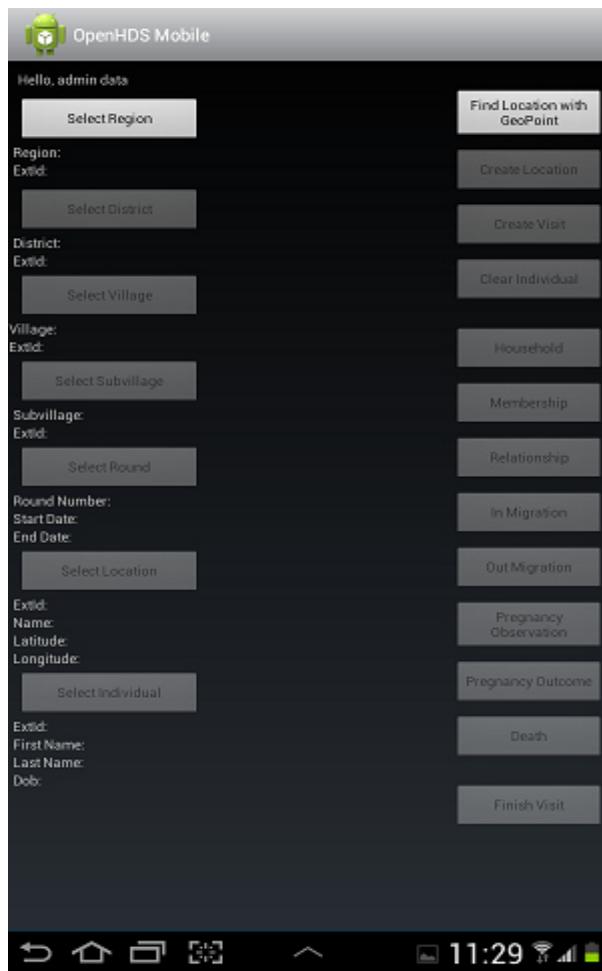


Figure 34

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 35

Once Field worker taps the “**Create Location**” button s/he will see the location page as depicted on Figure 36 below;

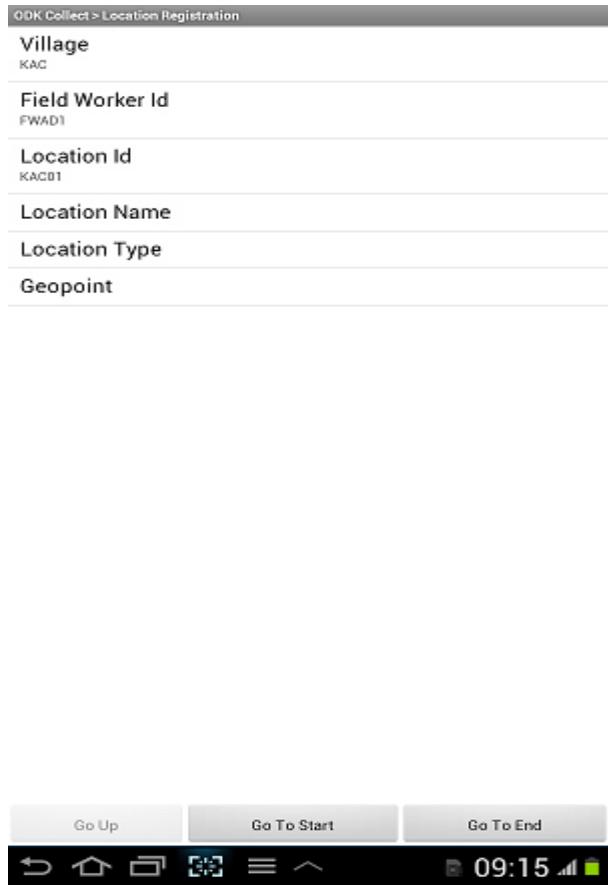


Figure 36

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 37. Please note the lower the numbers the better the accuracy.



Figure 37

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 38 below;

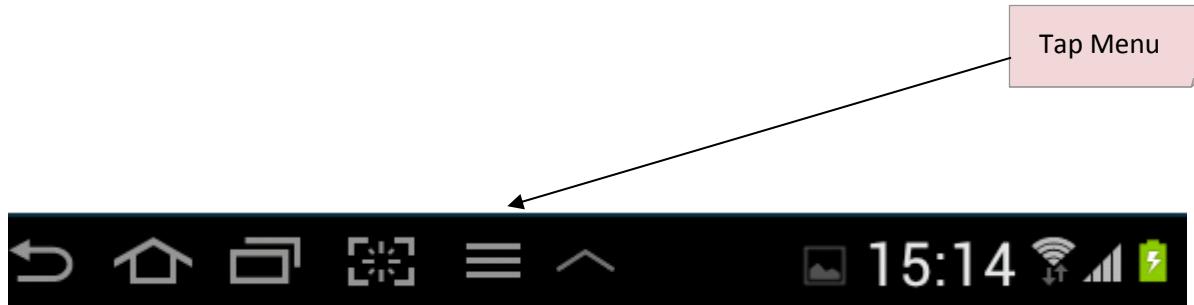


Figure 38

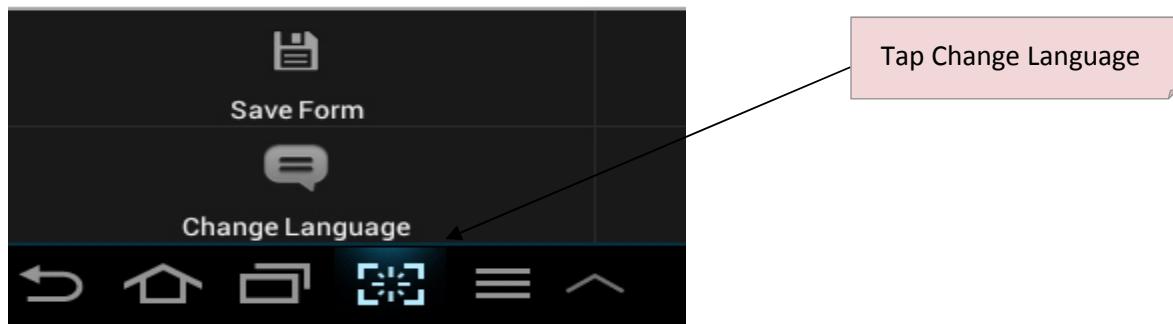


Figure 39

The tap change language to go to the Language selection menu as per Figure 40 below;



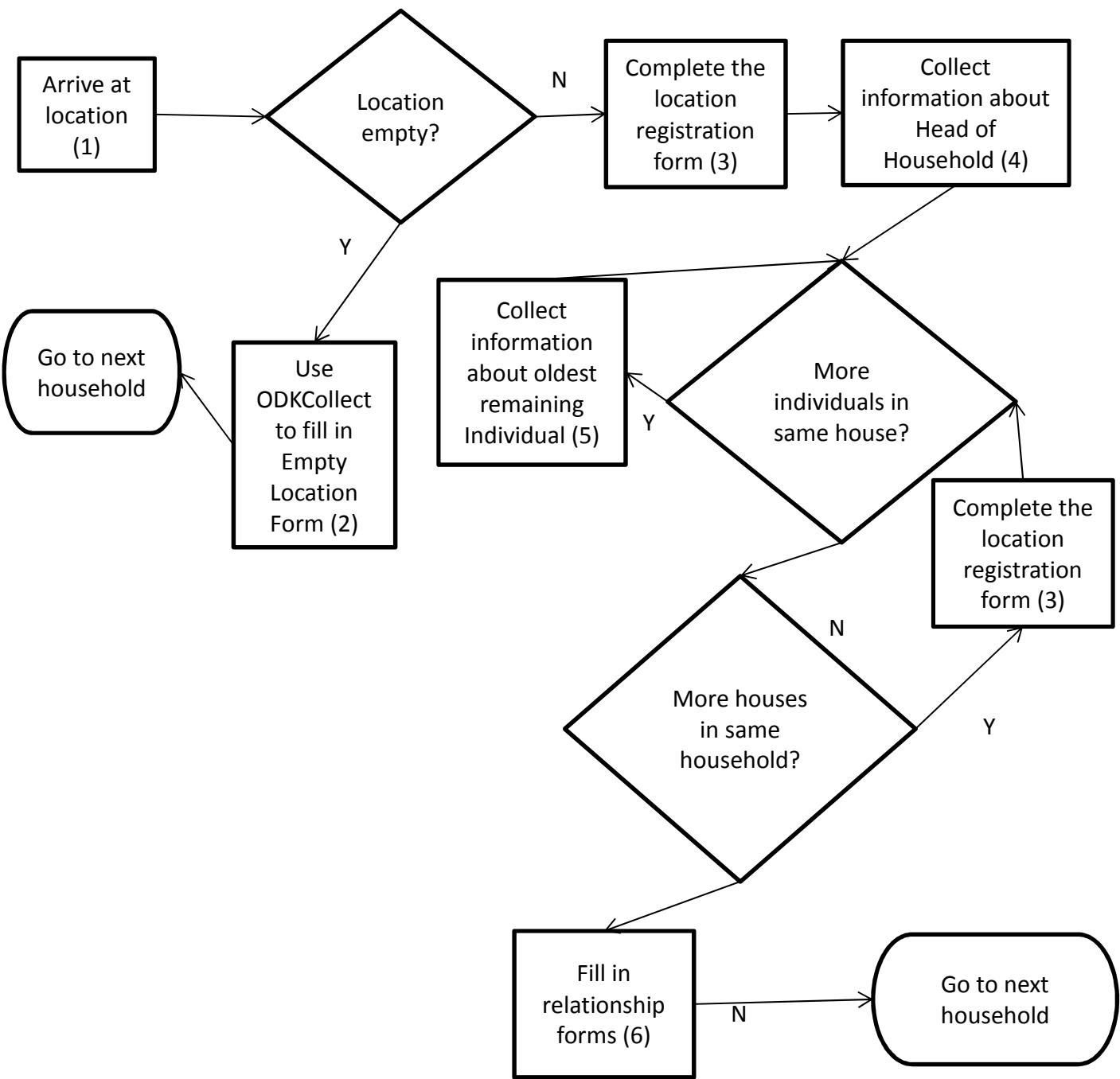
Figure 40

Tablet workflows

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

Baseline Visit

Following flowchart shows the basic workflow of a baseline visit:



(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

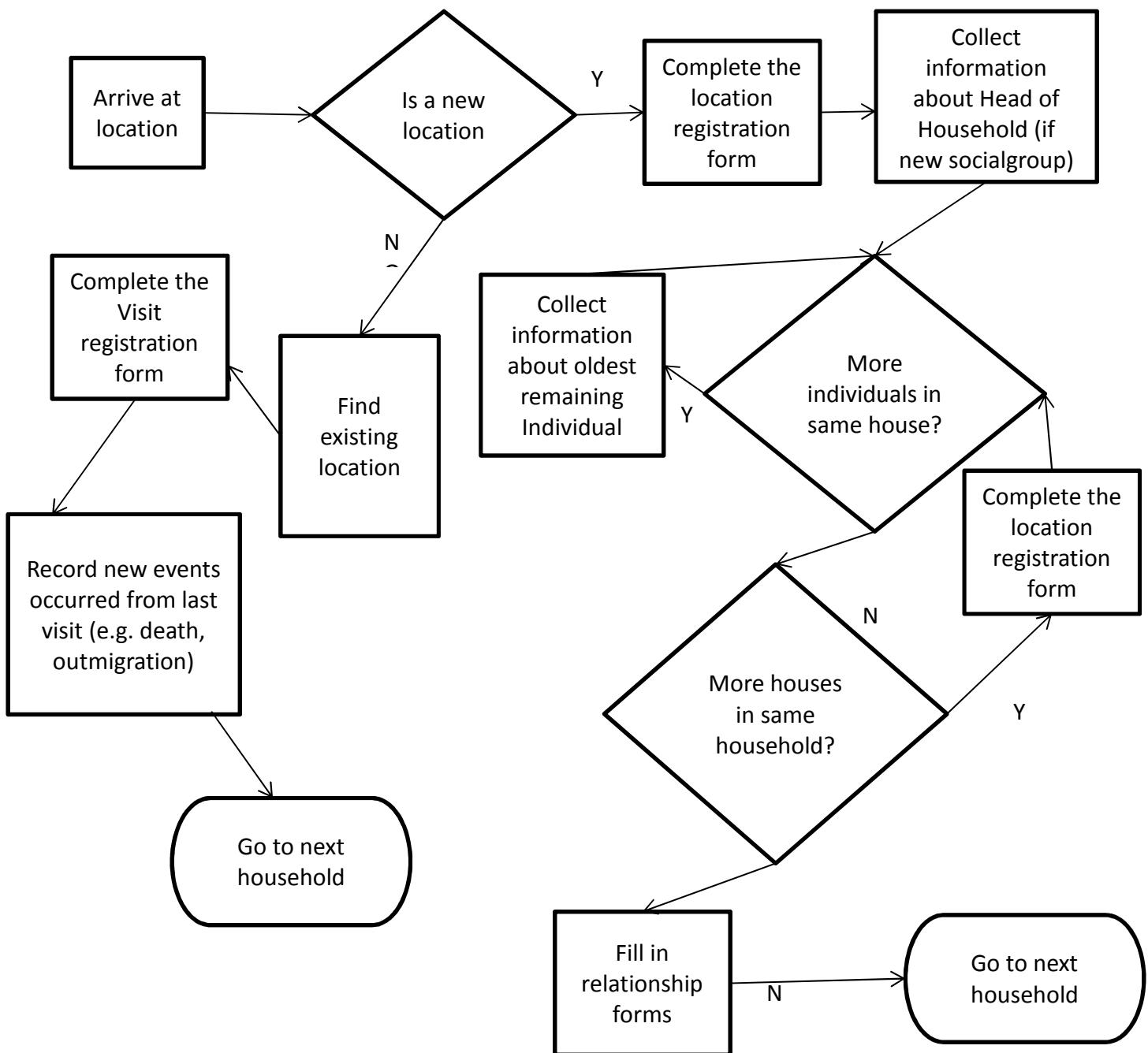
or create a household. Fill in the household information, use the last name of the head of household as household name. Finish by selecting “Clear individual”.

(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

Following flowchart shows the basic workflow of an update 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,

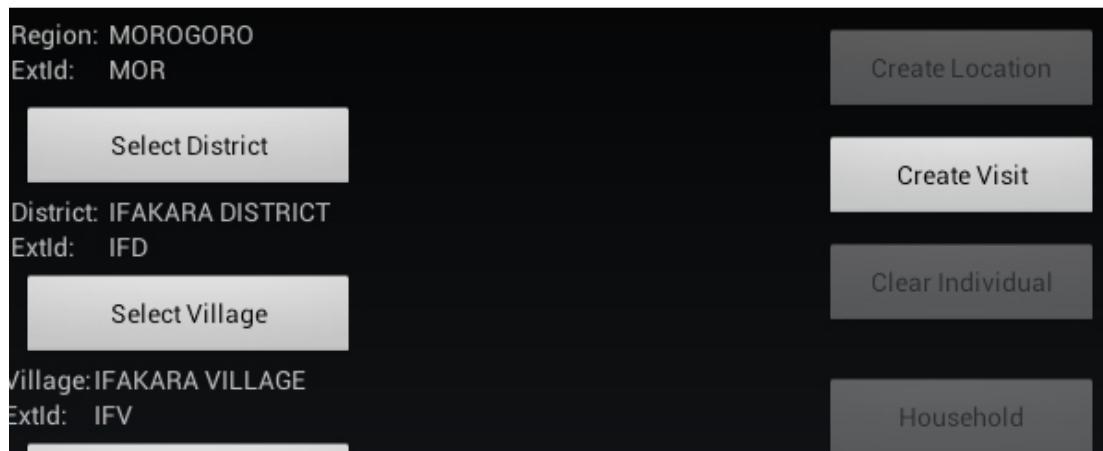


Figure 42

Filter Existing Location

Sometimes a Field Worker may 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 43 below shows the search location page for user to search the actual location they need to visit.

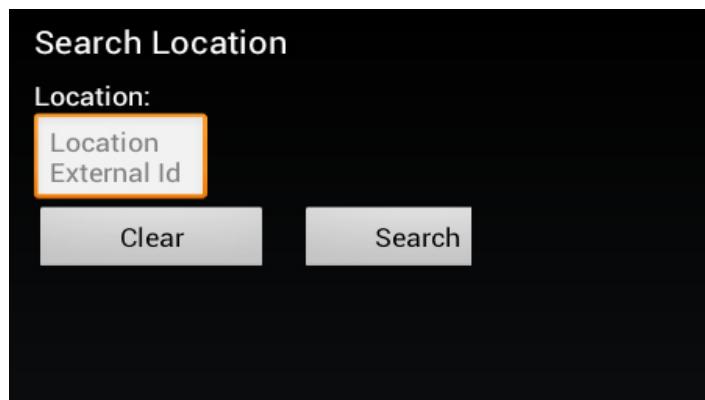


Figure 43

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 44 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.

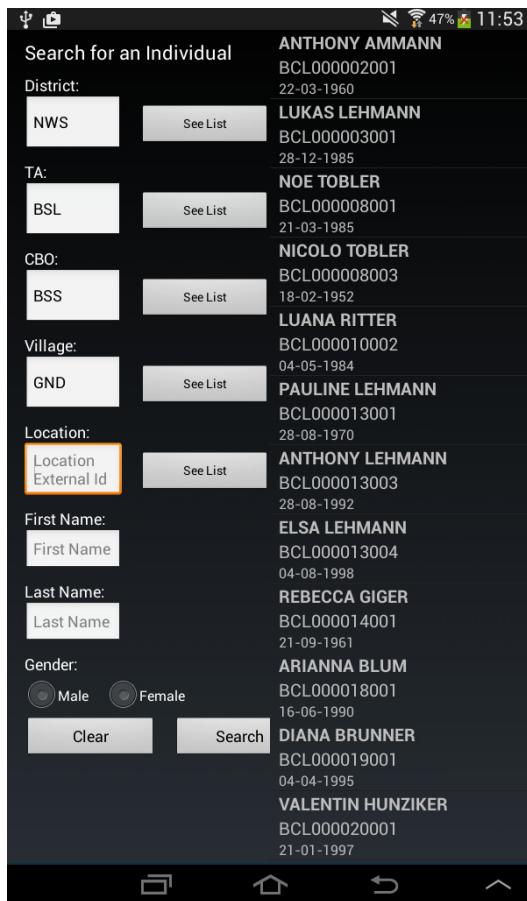


Figure 44

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 45 below;

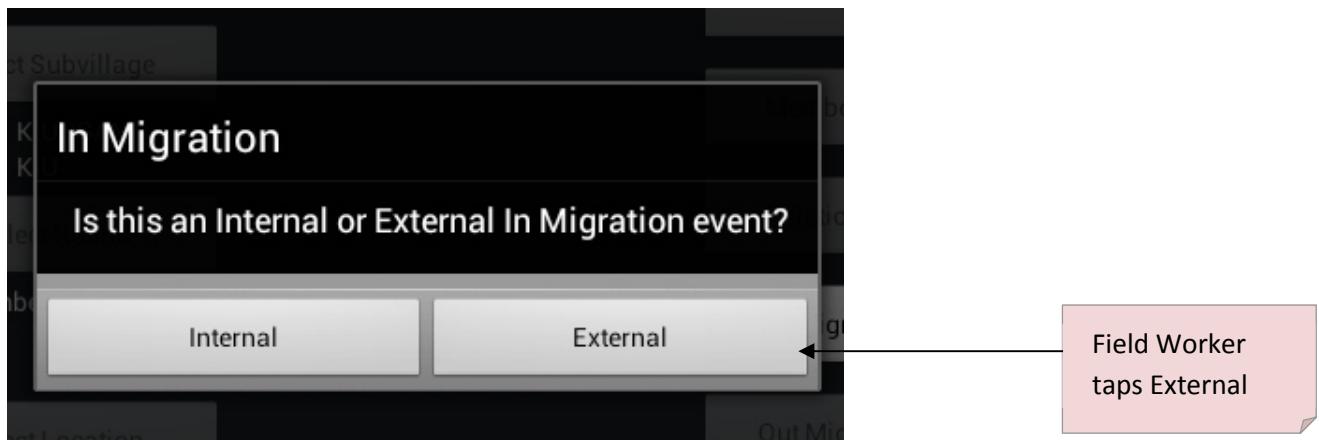


Figure 45

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 Figure 46 and Figure 47 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.

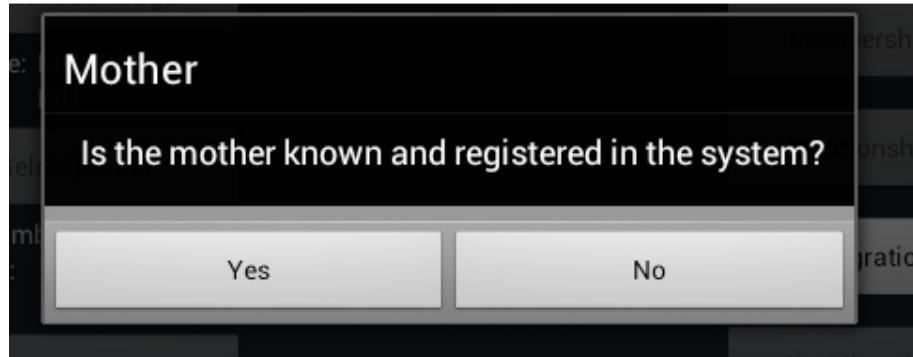


Figure 46

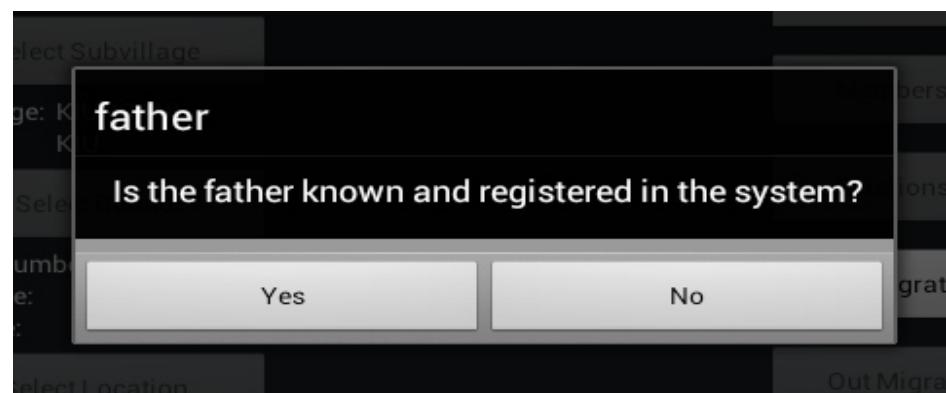


Figure 47

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 48 below:

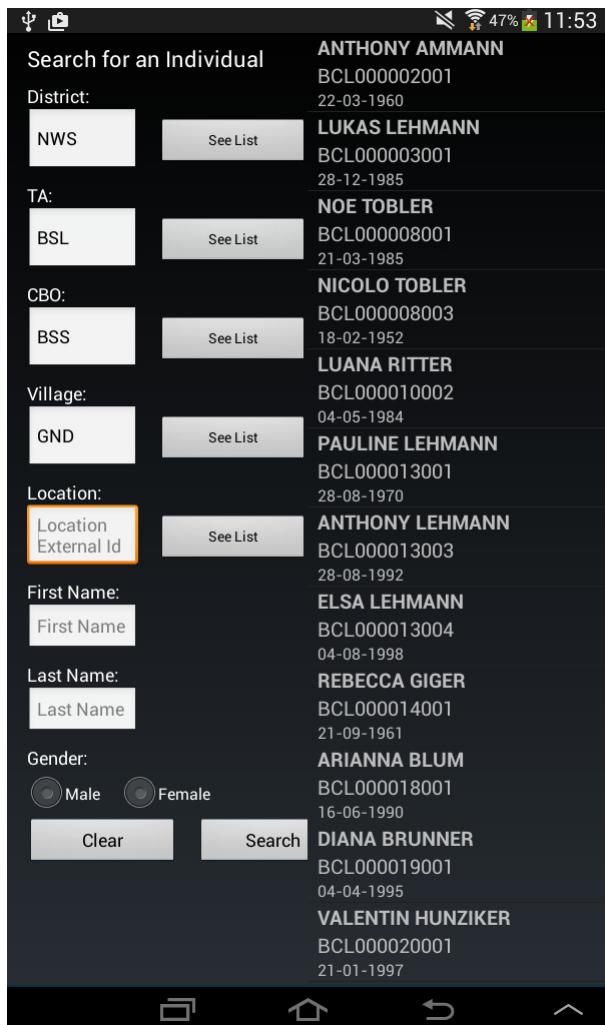


Figure 48

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 49 below;

ODK Collect > In Migration

Location Id	IFA000002
Visit Id	IFA000002018
Field Worker Id	FWAD1
Individual Id	IFA000002003
Mother Id	IFA000002002
Father Id	UNK
First Name	
Middle Name	
Last Name	
Gender	
Date of Birth	
Partial Date	
Date of In Migration	
Origin	
Reason	

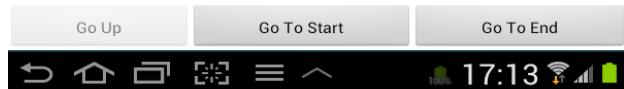


Figure 49

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 49 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 50 is the window asking Field worker to create membership. Also please follow the procedures for adding individual membership on the **membership** section;

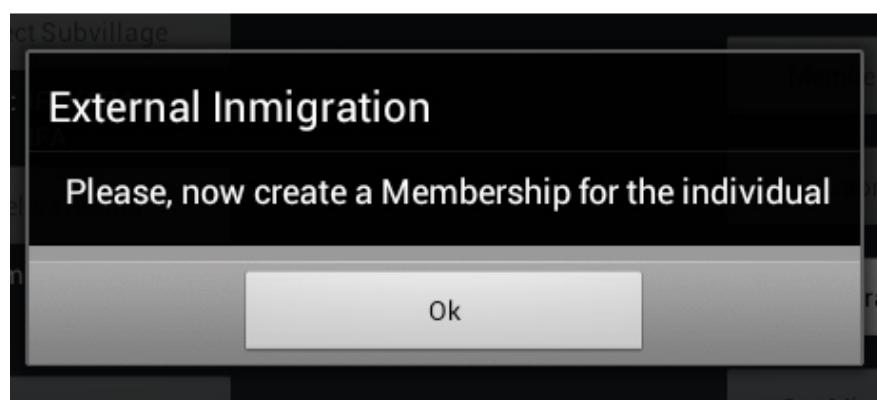


Figure 50

If we are migrating a social group in a new household first the field worker should migrate the head of the household.

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.

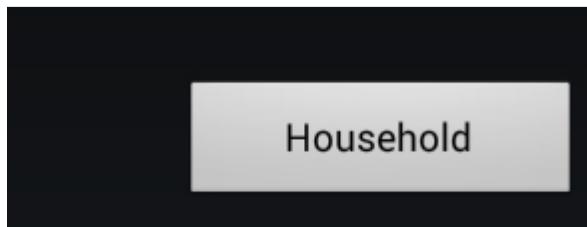


Figure 51

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 52 below;

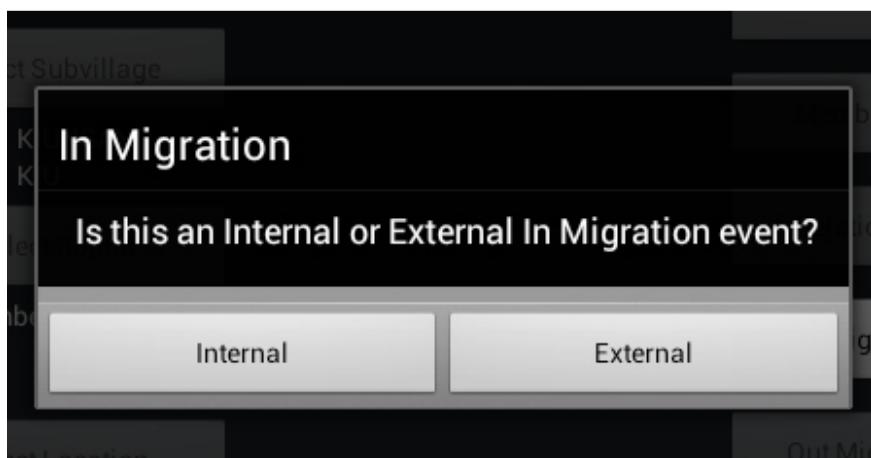
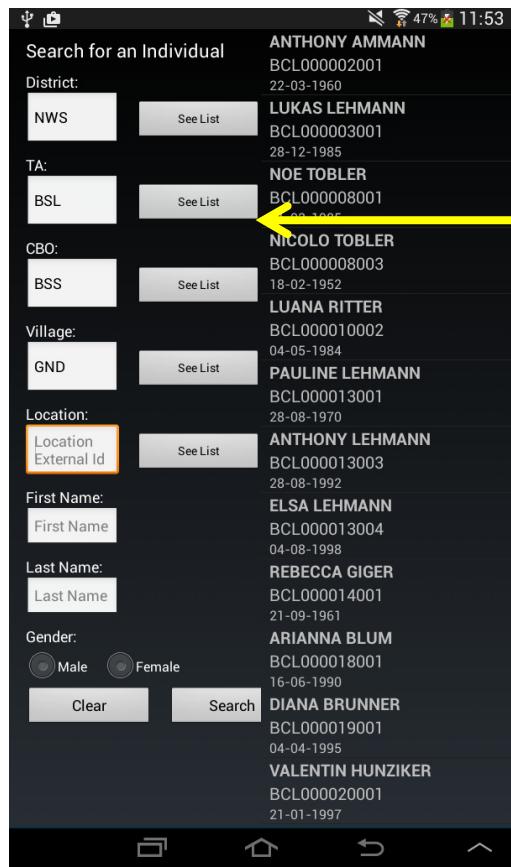


Figure 52

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 53 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 53

ODK Collect > In Migration

Location Id	IFA000668
Visit Id	IFA000668017
Field Worker Id	FWAD1
Individual Id	IFA000668006
Date of In Migration	
Origin	IFA000668
Reason	

Go Up Go To Start Go To End

08:52 3G

Figure 54

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 54 above. Field worker should fill in and save the form and exit.

Out Migration

The screenshot shows a mobile application interface for 'Out Migration Registration'. At the top, a dark bar displays 'ODK Collect > Out Migration Registration'. Below this, there are several input fields with pre-filled data:

- Individual Id:** KAT0002001
- Field Worker Id:** FWAD1
- Visit Id:** VKAC01491
- Date of Migration:** (empty field)
- Name of Destination:** (empty field)
- Reason for Out Migration:** (empty field)

At the bottom of the screen, there is a navigation bar with three buttons: 'Go Up', 'Go To Start', and 'Go To End'. Below the navigation bar, a black status bar shows icons for signal strength, battery level, and time (09:27).

Figure 55

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 56 below depicting



Figure 56

Field worker should start filling the blank fields as depicted in Figure 57 below, once finished should save the form and exit at the end of the form;

Figure 57

Change Head of Household

To change the Head of Household, you must first select a location and then fill out the visit form. After you did this, a button “**Change Hoh**” will appear (see

Figure 58 Change Hoh).

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

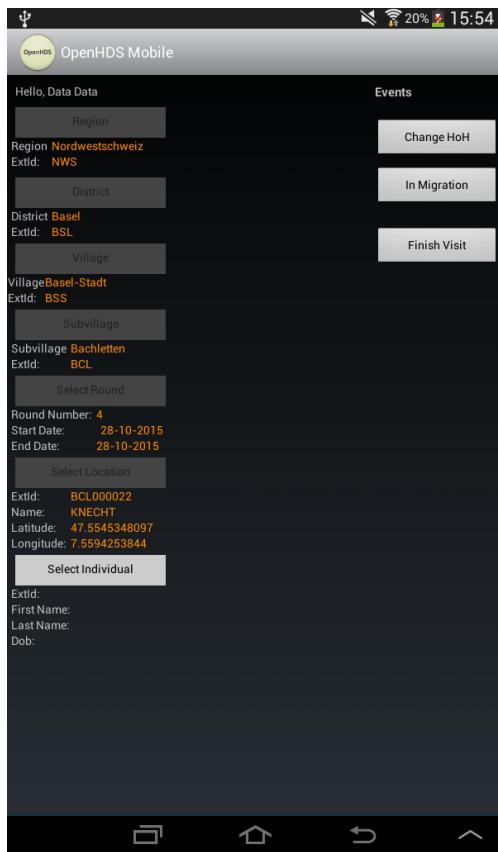


Figure 58 Change HoH

By clicking on the button, a list of all the eligible individuals which are associated with the same location/household is displayed (refer to Figure 59). When you select an entry from the list, a confirmation dialog will pop up, asking you if you want to continue with the selected individual. If you continue, the Change Head of Household form will load for you to fill out.

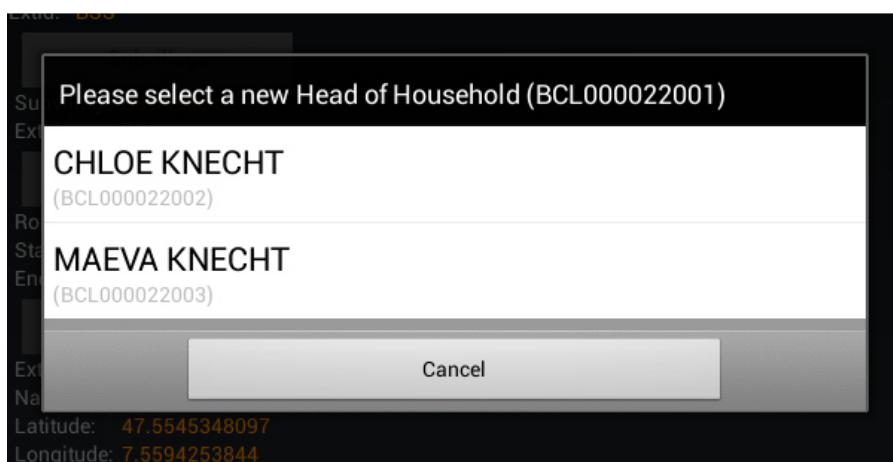


Figure 59

If there are no eligible successors found or no current Head of household defined for this Socialgroup/Location, a message will be displayed.

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 60 below:

ODK Collect > Relationship

Individual A
IFAO000668002

Individual B
IFAO000668003

Field Worker Id
FWAD1

Relationship Type

Start Date

Go Up Go To Start Go To End

09:29 3G

Figure 60

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 61 below;

The screenshot shows a mobile application titled "ODK Collect > Pregnancy observation registration". The screen displays a series of questions and fields:

- Number of months pregnant
- Does the expectant mother have:
 - Have you been to an ANC clinic?
 - Have you attended any other Health Facility?
 - Have you received any medicine for the pregnancy?
 - Have you received TT injection?
- Specify if other medicine
- Total number of pregnancies to date
- PERM Id
- Field Worker Id
- Visit Id
- expectedDeliveryDate
- Recorded Date

At the bottom of the screen are several navigation buttons:

- Go Up
- Go To Start
- Go To End
- Back arrow
- Home icon
- Search icon
- More options menu

Figure 61

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 62 below.

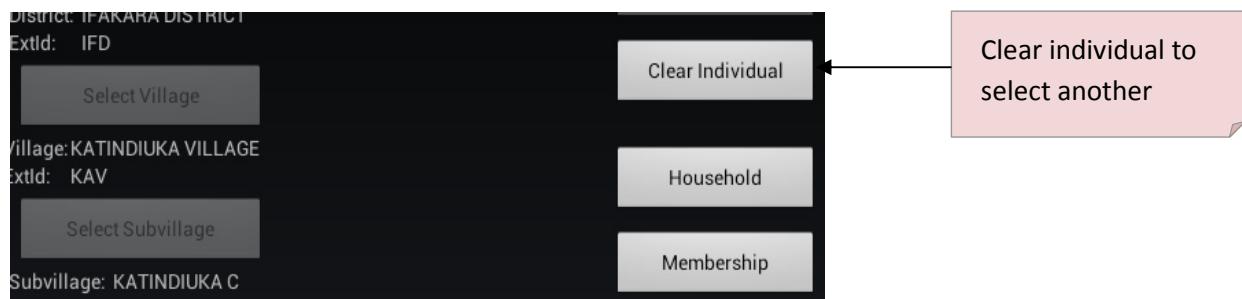


Figure 62

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 Figure 63 below:



Figure 63

Tap the father in the **Choose father**, window to link the new born with their father as depicted in Figure 64 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):

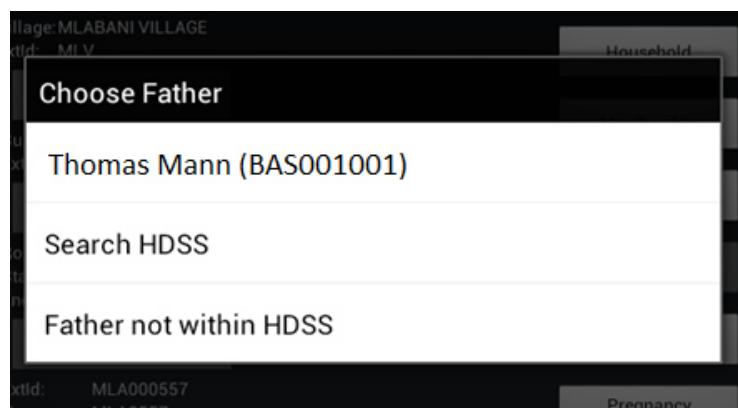


Figure 64

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 65.

ODK Collect > Death Registration

Individual Id
MLA000557002

Field Worker Id
FWAD1

Visit Id
MLA000557017

Date of Death

ODK Date : Application Date

Place of Death

Cause of Death

Go Up **Go To Start** **Go To End**

Figure 65

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.

The screenshot shows the ODK Collect application running on a mobile device. The top status bar indicates signal strength, battery level, and the time 11:14. The title bar reads "ODK Collect > deathToHOHform". The main screen displays a series of input fields and buttons. The fields include:

- Interview Date
- Household ID
- Field Worker
- Visit ID
- Deceased Head of House ID
- Number of Resident Household Members
2
- New Head of House ID
- First Name of New Head of House
- Last name of New Head of House
- Date of Birth of the New Head of House
- Date of Death
16/04/14
- Death Cause diagnosed by health authority?
- Place of Death
- remaining
1
- (▼) members

At the bottom of the screen are three buttons: "Go Up", "Go To Start", and "Go To End". Below these buttons is a black bar containing four icons: a left arrow, a home icon, a square icon, and a three-dot menu icon.

Figure 66

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 67 below

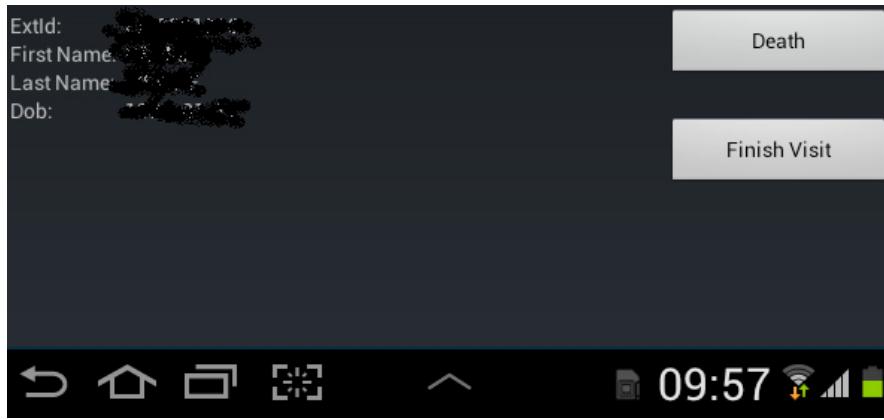


Figure 67

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 68 below displaying: Tap **Yes** to finish the visit, otherwise tap **cancel** to proceed with the visit

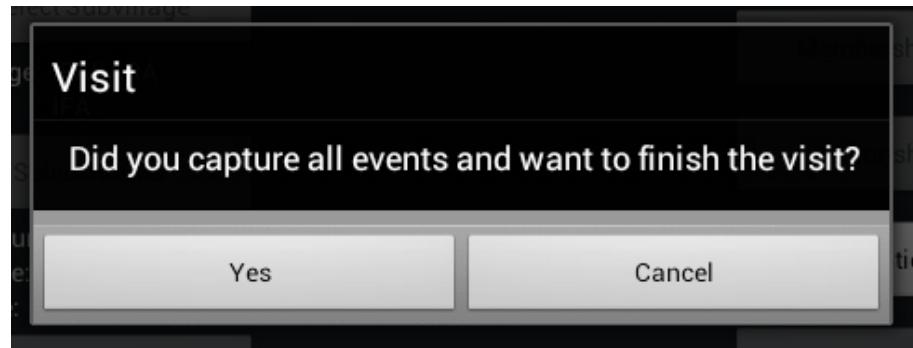


Figure 68

VERY IMPORTANT:

For HDSS purpose the hypothesis is that every location is visited once per round. If for any other reason a house is visited more than once (e.g. for special study or extra forms) no HDSS events should be recorded on these visits. HDSS events should be recorded the day of the first visit only.

Data manager manual

Fieldworkers creation

On the OpenHDS server the data manager can create/edit fieldworkers through the Utility Routines
→ Fieldworker.

The fields needed are

First Name:	<input type="text"/>
Last Name:	<input type="text"/>
Password:	<input type="password"/>
Confirm Password:	<input type="password"/>

If e.g. the fieldworker name and last name are John Smith then the fieldworker created in the system will be FWJS1 (FW + first letter of the first name + first letter of the last name + a number that goes from 1 to 9 if there are other FWs with same initials).

We **strongly** suggest that the data manager keep note of username and the passwords on a document (electronic or paper).

The username created and the password entered will be the ones to be provided to the fieldworker to login to the tablet once the sync of the tablet has been done.

The reason to keep a document tracing the fieldworkers' credentials is not only to provide them those to login but in case a fieldworker forget the password to login into the tablet.

The data manager could always of course reset (edit the password) but this would need a re-sync of the tablet to make the changes take effect on the device.

Collection of non-core variables in (“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 formId 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: filter that allows specifying if the form is just for Males or Females or applying to all.

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

When a Field worker starts the survey, once he gets to 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.

An extra form (a new questionnaire) can be designed with Excel easily following determined syntax and rules and then should be transformed in XForm format (format supported by ODK) to be uploaded on the Aggregate server. The tool we suggest to use to do that is XLSform (formerly XLS2Xform).

On the URL <http://xlsform.org/> you can find all the information about XLSform, including syntax, rules (basically all the info needed to create your questionnaire). Follow the link <https://opendatakit.org/wp-content/uploads/2013/06/sample.xlsform.xls> to download an example questionnaire Excel file.

Extra-form data integration

Data from Extra Forms can be transferred directly from ODKAggregate to OpenHDS. By doing so, data integrity is enforced, since the form entries are linked with the entities in OpenHDS. Hence modification or removal of linked extra form entries in OpenHDS is restricted to keep data consistent, and records properly linked. In addition, by keeping the submissions from the extra-forms tightly coupled with the OpenHDS data, in case the extra forms get removed from the ODK Aggregate server instance, the data is still available.

Currently, the extra form integration and transfer functionality excludes certain features of the questionnaire design as supported by ODK. Specifically, loops and groups are not supported at this stage (due the ODK internal way of splitting data into several tables when groups or tables are used). Further development of this extra-form module is under way and will amend some shortcomings.

This extra-form functionality is currently still experimental.

The following section describes the Extra-form functionality more in detail:

XLSForm

Electronic questionnaire forms that make use of the OpenHDS Extra-form feature will need to follow a special XLSForm structure before they are uploaded to the ODK Aggregate server. The extra form will have to include an additional **required** field with the name “**processedByMirth**” of data type *integer*. It is compulsory to set the value in the column “read_only” for this entry to *true* and default equal to zero to prevent any false user input.

In addition, if you want to make use of the extra data integrity constraints, the extra form questionnaire should include at least one of the following entries:

A group with name “**openhds**” and one or more of the following fields: *locationId*, *visitId*, *fieldWorkerId*, *roundNumber*, *individualId*, *householdId* The values of the appropriate fields are filled in automatically by OpenHDS Mobile app if available. To prevent any wrong user input, the *read_only* flag for these entries must be set to *true* in the XLSForm.

Additional supported fields or naming conventions may be supported in the future

Extra Form execution order

The extra-form functionality is separated into two steps. First, the database table is created for the extra-form data. Then, all the available form submissions are transferred over to this new database. Both of these activities are implemented by using Mirth Connect Channels and a dedicated OpenHDS ExtraFormController web-service.

1. Step: Create Table

Every extra-form will have its own table created to store the submitted data in. This will happen in five steps:

1. Query for new extra forms
2. Fetch form Id
3. Find core table for form Id
4. Read core table structure
5. Send table info to OpenHDS web-service

The Mirth *ExtraFormCreationChannel* will poll the OpenHDS database and wait for a new extra-form that was defined on the OpenHDS website in the ODK forms section. A new extra form is defined as an entry in the form table with an empty CORE_TABLE value and a status of 0. The entry must also be set to active and the flag for deleted must not be set to 1. Upon finding a new form, the Channel will get the formName for this entry and connect to the ODKAggregate database to find the responding CORE TABLE name for this form name. The structure of this table is extracted and then sent to the OpenHDS extra-channel web-service as an XML-Payload through which the new table will then be created. After a successful creation, the CORE_TABLE name in the form table for this form is set to the newly created table and the status is set to 2. (See also figure 70 for an overview).

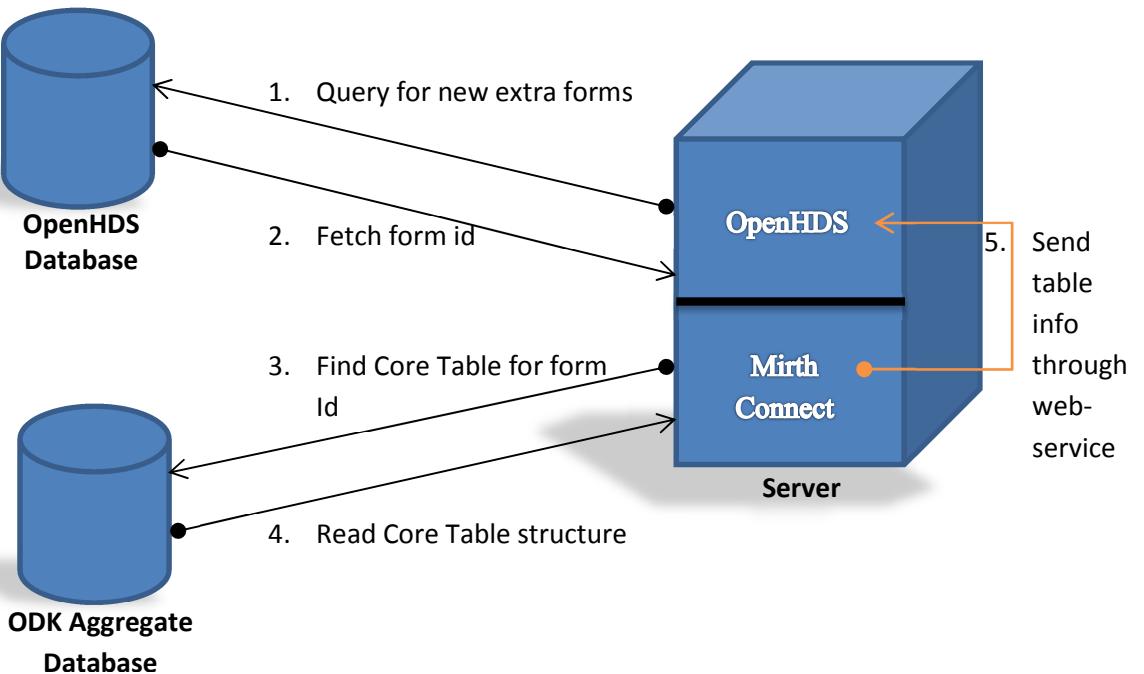


Figure 69

2. Step: Save Submissions

After the creation of the extra form table, new extra-form submissions can be transferred over. This happens in these 5 steps:

1. Query for ready form entries
2. Fetch CORE_TABLE
3. Search for CORE_TABLE data
4. Read unsent CORE_TABLE data
5. Send submissions

The Mirth *ExtraFormSendDataChannel* will poll for entries in the OpenHDS *form* table that are ready for receiving submitted data. An entry that is ready is defined hereby as a row in the table *form* that is set to active, not deleted, has a value for the CORE_TABLE and its status is set to 2. When such an entry is found, the CORE_TABLE is selected in the ODK Aggregate database, looking for rows where the processedByMirth flag is set to 0. This data is then transformed into an XML, before this payload information is sent to the OpenHDS extraForm web-service. From there, the information is verified, foreign key references are added and the data then inserted into the CORE_TABLE for this form. Upon successful insertion, i.e. the web-service returned a http value of success, the Mirth Channel will set the processedByMirth status in the ODK Aggregate database for this entry to 1, in case of errors to 2.

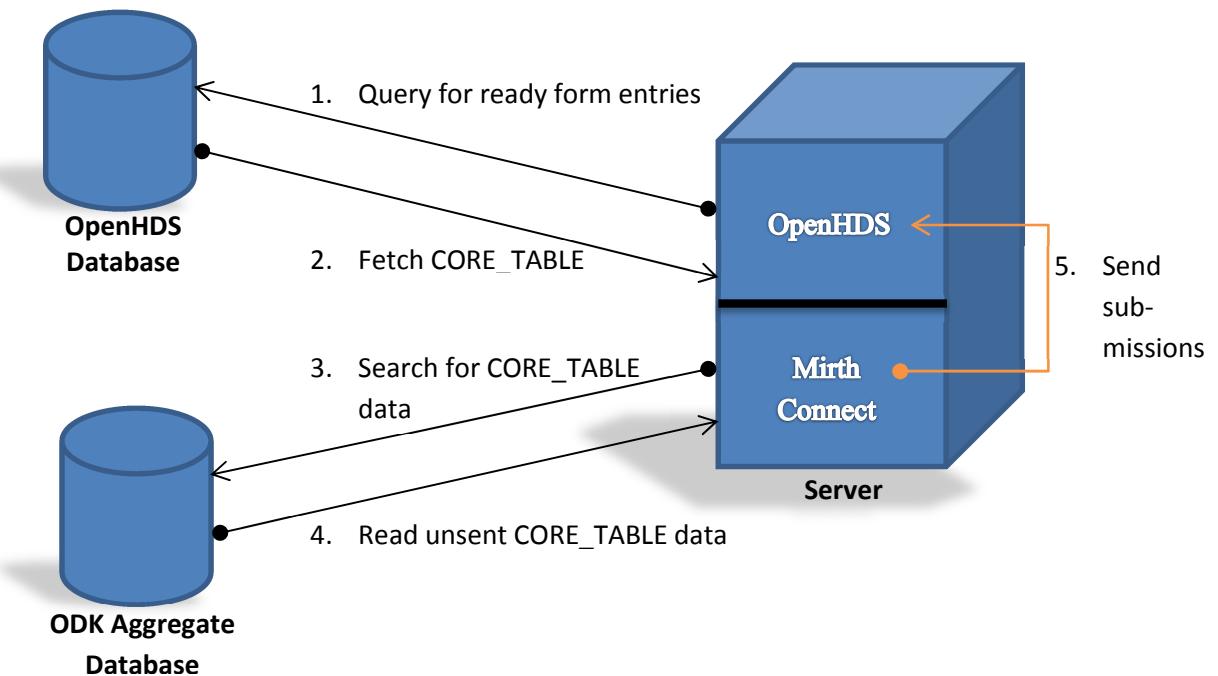


Figure 70

Export Extra Form submissions

If a new submission is sent to the ODKAggregate server, and this submission belongs to a form that was defined in OpenHDS as an *Extra Form*, this entry gets transferred over to the OpenHDS database. From now on it is decoupled from the ODKAggregate database and resides there even after the form it belongs to is removed from the ODKAggregate application.

You can export these extra form submissions from within OpenHDS by going to “Utility Routines” and then the “ODK Forms” in the sub-menu. From there, the “Listing” link in the top menu will bring you to an overview of all the Extra Forms that were defined.

The screenshot shows the OpenHDS interface with the following details:

- Left Sidebar:** Includes links for Select Language (Welcome admin), HOME, GETTING STARTED, BASELINE, UPDATE, AMENDMENTS, REPORTS, and UTILITY ROUTINES (Tasks, Data Extensions, Individual History, Modify Group Head, Death for Group Head, Field Workers, ODK Forms).
- Top Bar:** Shows the title "OpenHDS" and a "Listing" tab.
- Table Headers:** Form Name, Active, Apply to gender.
- Data Rows:**

Form Name	Active	Apply to gender
baseline	Yes	All
extra_form_v1	Yes	All
PHMRC_Full_Instrument_10_11_13	Yes	M
- Right Side:** A column of icons for each row, including a blue info icon, a pencil icon, a red trash bin icon, and a blue folder icon.

Figure 71: Extra Form listing with submission icon on the far right

From there a click on the *Submissions* icon will open the submissions view. From there you see the number of available submissions and also the currently generated exports.

Generating new export

If there are no available exported files, but submissions available for exports, a button will be shown (see Figure 72: Empty Submissions view). Clicking on the button will start the generation of the export file with all available submissions. It is formatted as a CSV-file with one entry for each submission. The button will also show up if the generated date of the latest export is older than 1 day.

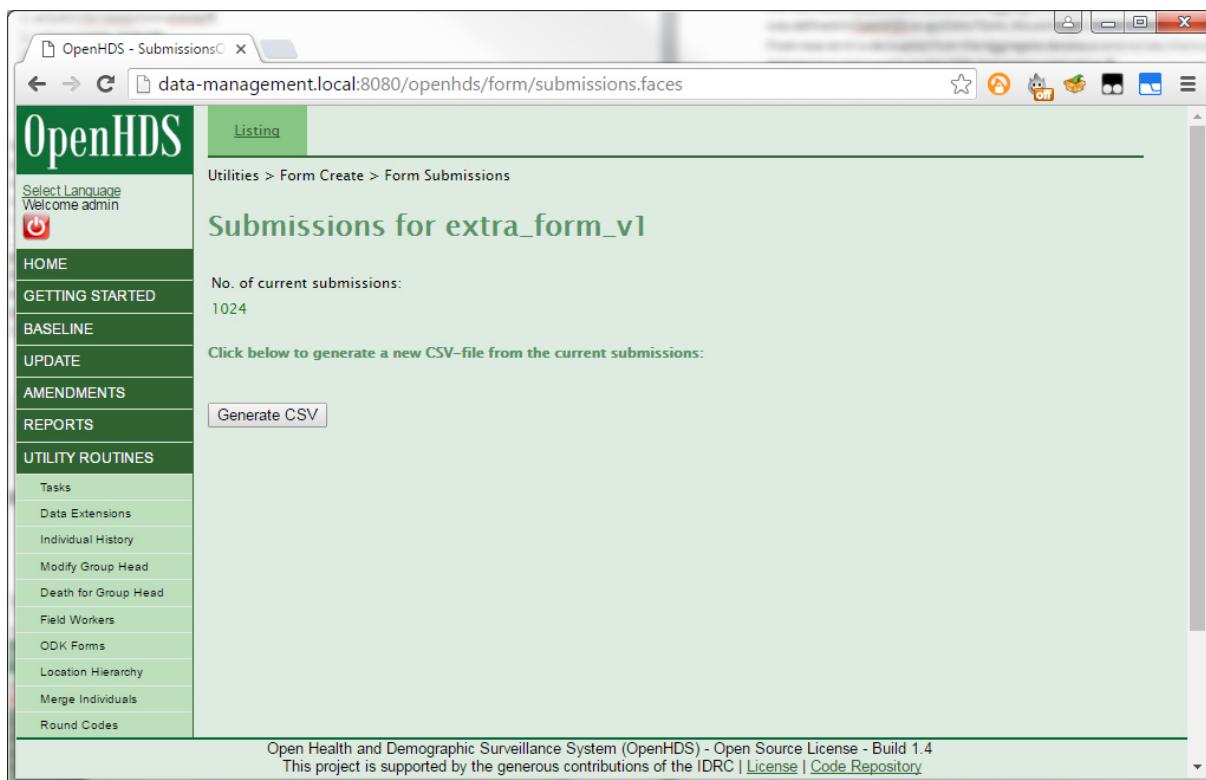


Figure 72: Empty Submissions view

Download of extra form exports

If exports are available, a table will show all the files you can download with the date of creation, file size and the number of lines. The row with the entry for the most recent file is highlighted in yellow. You can download an export an entry by clicking on the Download icon in the last column. If you wish to delete a specific entry from the server, click the Delete icon. A pop-up message will show, asking you to confirm if you really want to remove the selected file (see Figure 73).

The screenshot shows the OpenHDS Submissions interface. The left sidebar has a dark green header "OpenHDS" and a white body with links like "Select Language", "Welcome admin", "HOME", "GETTING STARTED", "BASELINE", "UPDATE", "AMENDMENTS", "REPORTS", and "UTILITY ROUTINES". The "UTILITY ROUTINES" section is expanded, showing "Tasks", "Data Extensions", "Individual History", "Modify Group Head", "Death for Group Head", "Field Workers", "ODK Forms", "Location Hierarchy", "Merge Individuals", and "Round Codes". The main content area has a green header "Listing" and a breadcrumb "Utilities > Form Create > Form Submissions". Below it is a title "Submissions for extra_form_v1". A message says "No. of current submissions: 1024". Under "Available Exports" is a table:

Filename	Last Modified Date	Size	Line Count	
extra_form_v1_submissions_2016-04-04_214700.csv	04-04-2016 21:47:00	460.4 kB	1025	LATEST!

A message below the table says "Found existing recent export." At the bottom, there's a footer with "Open Health and Demographic Surveillance System (OpenHDS) - Open Source License - Build 1.4" and "This project is supported by the generous contributions of the IDRC | [License](#) | [Code Repository](#)".

Figure 73

Error Handling

Error handling and data cleaning should be done on a continuous basis. Monitoring of the data submission first happens at the stage of transferring data from ODKAggregate to OpenHDS through MirthConnect. By default, the Mirth channel will poll the ODKAggregate database every night. Mirthconnect reports error messages as created by the OpenHDS web services, and allows the data manager to browse and filter errors (Figure 74 Error reports in MirthConnect).

Faulty records are also visible in the dedicated views using the role of datamanager in an MySQL client.

The data manager after log in into mysql can browse the views available. He is required to fix the records which failed successful transfer to OpenHDS (i.e. are flagged processed_by_mirth=2), if necessary in consultation with the field team.

The data manager will get an email from mirth with an attachment (csv Error file) if error occur during the transfer.

In the file he gets he has the following information:

- The channel that failed (e.g. inmigration)
- The data contained in the record failed (e.g.
<inmigration><collectedBy>FWAD1</collectedBy><visit>BAS000001000</visit><individual>BAS00000003<firstName>ARTHUR</firstName><lastName>NELSON</lastName><gender>M</gender><dob>2006-10-24 00:00:00.0</dob><dobAspect>1</dobAspect><mother>BAS00001002</mother><father>UNK</father></individual><origin>ITALY</origin><reason>WORK</reason><recordedDate>2014-10-10 00:00:00.0</recordedDate><migType>EXTERNAL_INMIGRATION</migType></inmigration>
- The error message: <errors>Invalid mother id </errors>

After error resolution of the problem (e.g. correct the mother id for the above example), the processed_by_mirth flag should be reset to zero to make the record eligible for reprocessing by Mirth. Records which are fixed and reset in this way will be picked up the following night when the Mirth channel polls the ODKAggregate database again.

Ideally the data managers should go through all the records in the file once he receives it as soon as possible. It is always possible for the data manager to access the error table, where he gets the cumulative list of errors that come day by day. Data manager can access the error table, check the error record and also add a comment on a particular record. It is also possible to export the table itself to an Excel file. Ideally the error table is truncated at the end of the round to prevent it from getting too big.

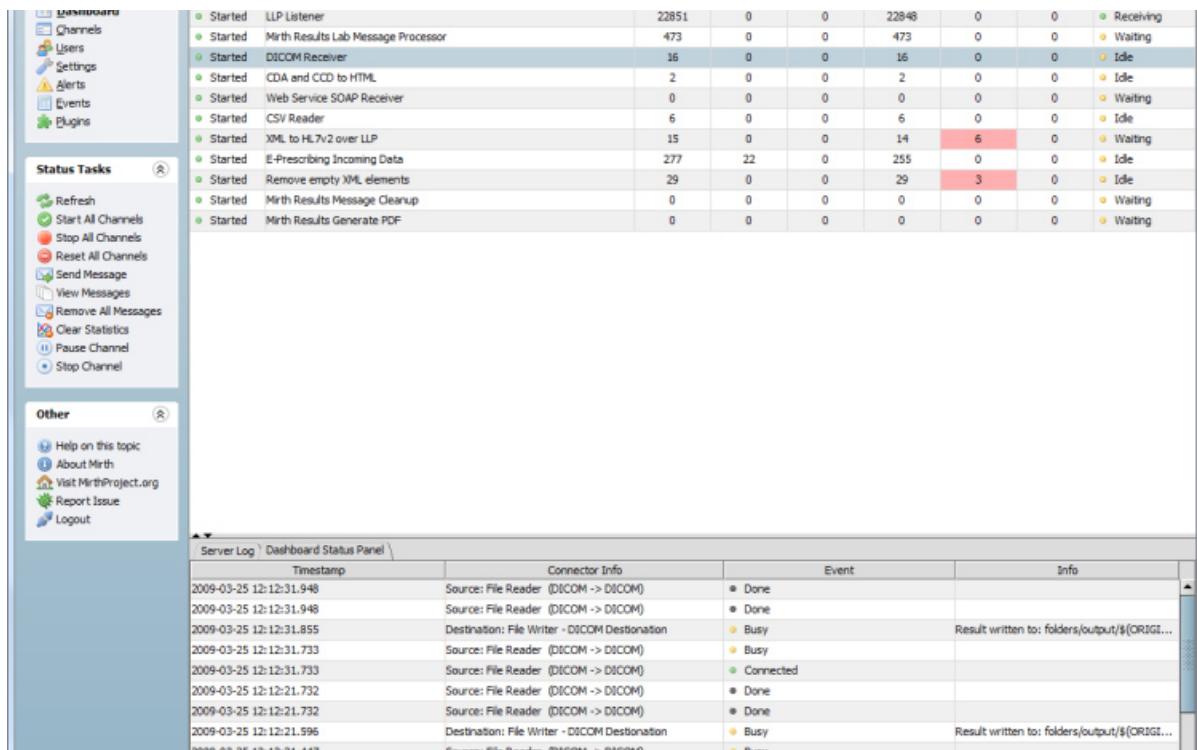


Figure 74 Error reports in MirthConnect

Data flows in OpenHDS

The data flow from the tablet to the OpenHDS server is illustrated in Figure 75 Data flows and monitoring. On submission to ODKAggregate, the records are flagged by a column in the corresponding database table as unprocessed by Mirth: each table has a column processed_by_mirth, which is set to zero at this stage. MirthConnect attempts to transfer these record to OpenHDS, and set the flag processed_by_mirth=1 on success. A record which fails to pass because of a violation of one of the imposed constraints has the flag processed_by_mirth set to two. These records have to be addressed by a data manager as detailed in the following section.

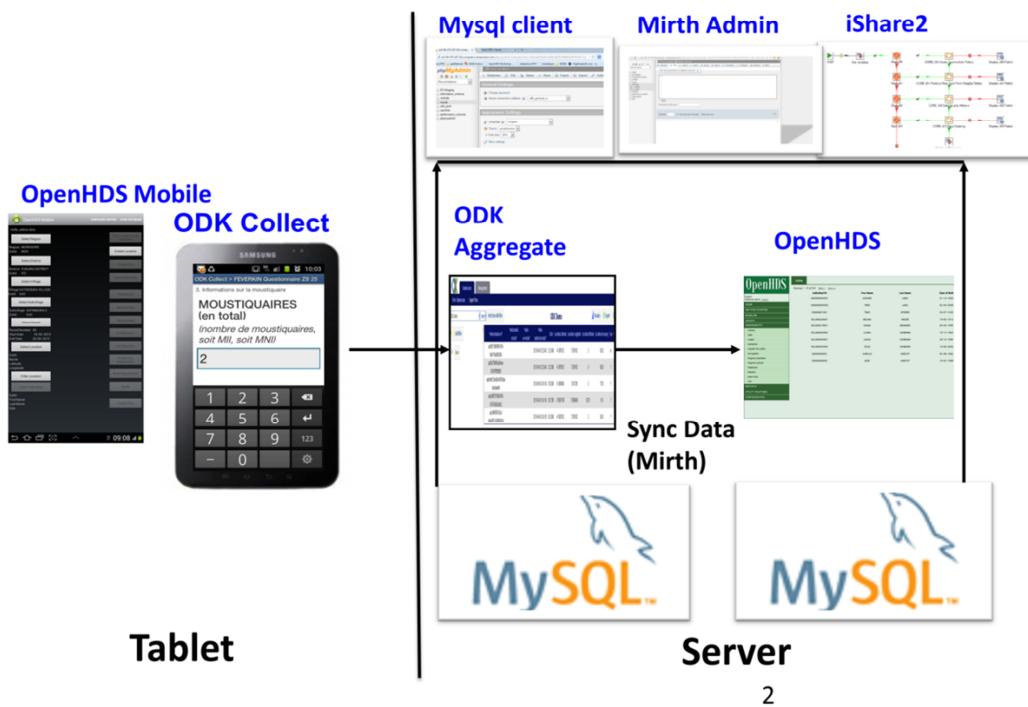


Figure 75 Data flows and monitoring

Migration of legacy data to OpenHDS

This section explains the necessary steps to migrate from the HRS2 platform to its successor OpenHDS. (Figure 76: Migration of legacy data from HRS2 to OpenHDS) Figure 74 shows the principle of using a MirthConnect channel for moving data from HRS2 via a staging database to OpenHDS. This approach allows re-use of much of the OpenHDS error checking functionality that was developed for Baseline and Update rounds.

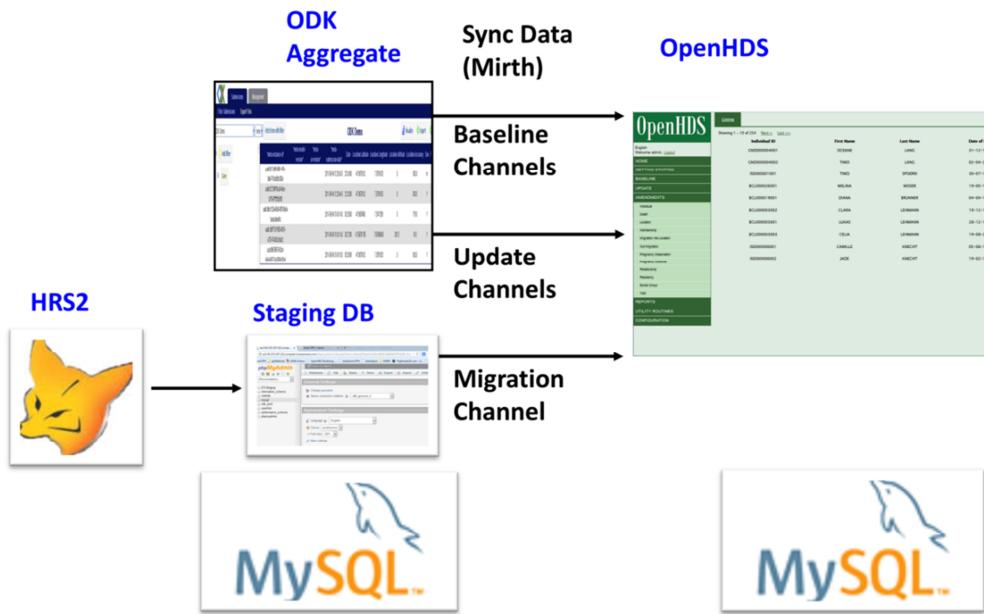


Figure 76: Migration of legacy data from HRS2 to OpenHDS

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.

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.

We recommend to dry-run this process first by following the tutorial using the scripts and synthetic data found here: <https://github.com/SwissTPH/openhds-from-hrs2>. The scripts starting with numbers referred to below (e.g. 01_create_hrs_mysql_db.sql) refer to files which are part of the data migration tutorial. We assume that you use the same directory layout as in the github repository. Adjust your paths in the scripts if this is not the case.

Requirements

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

- HRS2 Database Dump (DBF-Files)
- OpenHDS (version 1.2 or later) with MySQL as database system

File Conversion and staging HDSS database

In a first step all dbf files corresponding to the tables listed as required below need to be imported into a database staging database, we here use hrs_mysql_db to name this. There are various ways of

achieving this (e.g. via export to csv files and import to mysql), here we demonstrate how to use R for this purpose. For this:

All DBF-Files should be in one directory.

Create a database named hrs_mysql_db (01_create_hrs_mysql_db.sql). This database is only needed temporary and will be used to clean the data.

Run script 02_a_dbf_mysql.R to convert the dbf to mysql, making sure 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. Then run 02_b_alter_fields.py to change all database columns to datatype varchar (the default of importing from R is datatype text, which has drawbacks).

Required tables for migration

After the creation of the staging HDSS 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.

Common steps for all tables

These steps occur in steps 03-17 in the tutorial.

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
```

(03_alter_rounds.sql)

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))  
(04_alter_location.sql)
```

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);
(05_alter_observation.sql)
```

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_individual (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;
```

(06_alter_individual.sql)

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;  
(07_alter_socialgroup.sql)
```

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 EVENTTYPE 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';
```

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;
```

(08_alter_relationship.sql)

Membership

Run common SQL-queries.

create indexes:

```
ALTER TABLE membership
```

```
ADD INDEX IDX_MEMIND (INDIVIDID);
```

```
ALTER TABLE membership
```

```
ADD INDEX IDX_RELSPG (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='IMG' 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, EVENTTYPE, str_to_date(EDATE,'%Y-%m-%d') EDATE FROM membership
WHERE processed_by_mirth =0 order by SDATE asc;
```

(09_alter_membership.sql)

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
```

(10_alter_indvstatus.sql)

Pregnancy outcome (pregoutcome)

Run common SQL-queries.

create indexes:

```
ALTER TABLE pregoutcome
```

```
ADD INDEX idx_proind (INDIVIDID);
```

```
ALTER TABLE pregoutcome
```

```
ADD INDEX idx_prov (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
```

(11_alter_pregoutcome.sql)

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
```

(12_alter_birth.sql)

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;
```

(13_alter_residency.sql)

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;
```

(14_alter_death.sql)

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 OUTMGRREASO 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;
```

(15_alter_outmigration.sql)

Immigration

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 immigration where  
processed_by_mirth=0;
```

(16_alter_inmigration.sql)

Error table

The Mirth error reporting channel will write all problems to an error database. We can create table errors on the HRS database, for the tutorial we will create it in the odk_prod database so we don't need to reconfigure the channel (17_create_error_table.sql).

```
CREATE TABLE `errors` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `CHANNEL` varchar(30) DEFAULT NULL,  
  `DATA` varchar(1000) DEFAULT NULL,  
  `ERROR` varchar(280) DEFAULT NULL,  
  `exported` int(1) DEFAULT '0',  
  `inserted_timestamp` timestamp DEFAULT CURRENT_TIMESTAMP,  
  `COMMENT` varchar(500) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
```

Data import into OpenHDS

A few prerequisites for the OpenHDS database before transferring the data from the staging database (18_prepare_openhds_db.sql):

Create Fieldworker FWAD1 (Admin Data) in openhds database;

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

A location hierarchy corresponding to the entities in the staging database should be defined and present on OpenHDS (18_prepare_openhds_db.sql takes care of this).

Transformation using MirthConnect

Open Mirth Connect Administrator.

Status	Data Type	Name	Id	Description	Rev Δ	Last Deployed
Enabled	XML	Baseline	814bcc47-89b9-47af-882c-265ed71afddb	This channel will a...	0	2014-09-04 09:37
Enabled	XML	Baseline1	95bbf706-26dc-4af7-84ff-aed09a2607e8	This channel will a...	0	2014-09-04 09:37

Connected to: https://data-management.local:8443 | CEST (UTC +2)

Figure 77

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).

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 run on the HRS DB:**

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

Mirth-Channels

You can find the Mirth Channels and additional configuration files that are used for handling the Extra-Form functionality in the official SwissTPH OpenHDS *Mirth Channels* Repository. The Channel that is responsible for polling for new extra forms is called *ExtraFormCreationChannel*. The Channel that is used for sending the extra-form submissions is named *ExtraFormSendDataChannel*.

Configuration

To follow these steps, please first open the “Mirth Connect Administrator” application.

Channels

To import the Channels, first switch to the Channels View. Now select “Import Channel”. In the Open File Dialog, chose the *ExtraFormCreationChannel.xml*, then save and go back to the Channels View. Repeat the same procedure for the *ExtraFormSendDataChannel.xml*.

Global scripts

The following three variables are expected to be in the global map for the extra-form functionality to work. They are defined in the Global script and are used to establish the connection to the OpenHDS database.

- **OPENHDS_DATABASE_URL**: Defines the database URL and schema name that is used for establishing a connection to the OpenHDS database. The default schema name is *openhds*.
- **OPENHDS_DATABASE_USER**: The database username for connecting to the OpenHDS database
- **OPENHDS_DATABASE_PASSWORD**: The password for the database user specified above

These variables are already added and defined in the latest “Global Scripts.xml” file. You can import this file by going to the *Channels* view in the Mirth Connect Administrator, then click on “Edit Global Scripts”. From there chose “Import Scripts” to import the file.

Alternatively, if you just want to upgrade your existing installation, copy and paste the three lines found in the table below and paste them into your existing “Global Scripts”. Then make changes to your variables, so they fit your database settings.

Global scripts

```
globalMap.put("OPENHDS_DATABASE_URL","jdbc:mysql://localhost/openhds");
globalMap.put("OPENHDS_DATABASE_USER","data");
globalMap.put("OPENHDS_DATABASE_PASSWORD","data");
```

Mirth Code Templates

Following Javascript functions were created in the Mirth Code Templates to support extra-form functionality:

- **createTableDummy**: Creates a XML from JS Array containing Information about odk extraform table structure
- **getCoreTableFromFormName**: Extracts the Core Table name for a given formId in the ODK database
- **getCoreTableStructure**: Extracts the Table Structure from a given ExtraForm Core Table and returns a JS Array with the info about each of its columns
- **getExtraFormData**: Extracts ExtraForm Data from a given ExtraForm Table and returns a JS-Array containing objects (JS-Array) for each data entry, describing its column entries. This function expects an integer as second parameter, defining how many entries should be processed at once. The default value is 10, to prevent Out of Memory exceptions. It is recommended not to change this value.
- **getPrimaryKeyList**: Gets the Primary Keys for a given Extra Form Core Table as a JavaScript String Array
- **processForm**: Function to extract the Core Table Name from a given form URI in the ODK database

These functions are already stored in the latest “CodeTemplates.xml”. To import the file, in **Mirth Connect Administrator** switch to the *Channels* view. From there, select “Edit Code Templates”. In the following view, select the “Import Code Templates” option on the left and import the xml. Save and the functions are now available.

Set up an alert for errors on Extra-Forms in Mirth:

After importing the two ExtraForm Channels, please follow these steps:

1. Go to section "Alerts"
2. Double click on "Events Connection Alert"
3. On the right, under Channels, click on *ExtraFormCreationChannel* and Enable, do the same for *ExtraFormSendDataChannel*.
4. Click on "Save Alert" on the left. Ignore the info message.

You can now switch back to the Dashboard.

Migrating from older version of openhds SQL ALTER script

If you are upgrading from an already existing installation of OpenHDS and want to use the extra-form feature, you'll also need to modify the form table (2 new fields were added). Please find required ALTER SQL statement in the table below.

SQL ALTER TABLE

```
ALTER TABLE `form` ADD `CORE_TABLE` varchar(80) DEFAULT NULL, ADD `status` tinyint(4) NOT NULL DEFAULT '0';
```

OpenHDS evaluation platform

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, as well as launching a virtual server in the Amazon EC2 infrastructure. The latter has the advantage that an internet-visible network interface is readily available, which makes testing to the tablet application straightforward.

OpenHDS in VirtualBox

Download and install Virtualbox (<https://www.virtualbox.org/wiki/Downloads>). Download the openhds_server.ova (<https://github.com/SwissTPH/openhds-eval.com/SwissTPH/openhds-eval/>) and import this into your local Virtualbox host. This is an Ubuntu 16.04 LTS 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 192.168.56.101). 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"

192.168.56.101 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 /](https://data-management.local:8443/)

From there you can start the java webstart app. It can 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.