MOBILE APPLICATION TO SECURE TENURE

Technical Software Documentation

Version 2.1

Abstract

This document provides details on the technical design, deployment and configuration of different components of the Mobile Application to Secure Tenure (MAST) application.

Contents

1 INTRODUCTION AND BACKGROUND

1.1 Document Overview

The objective of this document is to provide the details of technical design, deployment, and configuration of different components of the Mobile Application to Secure Tenure (MAST) application. It also provides configuration details of external third-party tools that are utilized in the MAST application framework.

1.2 Context for MAST

The technology framework outlined in this document was developed to support the collection of land rights information using mobile technology, and to effectively store information to create an inventory of land information that creates a greater degree of tenure security.

The Mobile Technology and Crowdsourcing to Strengthen Land Tenure Security Pilot (hereafter referred to as Mobile Application to Secure Tenure (MAST) is testing the viability of an innovative participatory, or “crowdsourced”, approach to capturing land rights information, including information about customary holdings, using mobile technologies.

The Cloudburst Group is implementing this project for the United States Agency for International Development (USAID). The MAST pilot is issued under the Evaluation, Research and Communication (ERC) Task Order of the Strengthening Tenure and Resource Rights (STARR) IQCs. This contract is managed by USAID’s Washington-based E3/Land Office. The MAST pilot program fits into USAID’s strategic reform agenda pertaining to the use of science and technology to resolve development problems.

The development of the information technology platforms for MAST has been implemented in Tanzania. The objectives of the Pilot are congruent with the needs of the Government of Tanzania related to helping demarcate and secure land rights. Tanzanian development priorities are focused on lowering the cost of land titling programs and improving land governance in order to stimulate economic development, particularly by promoting large-scale investment in agriculture.

All phases of this pilot were implemented in relatively small villages in Iringa Rural District. Iringa falls in the important Southern Agricultural Growth Corridor of Tanzania (SAGCOT), a zone of interest for both the Government of Tanzania and for USAID. Ilalasimba was the first pilot village. Itagutwa and Kitayawa were the second and third pilot villages. Most of the economic activities within these villages are focused on agriculture. Maize is the predominant crop, and several secondary cash crops are grown by inhabitants (tomatoes, sunflower and tobacco). On average, each household occupies 2 or 3 parcels. Parcel sizes range from 5 to 10 acres and larger holdings are farmed in peripheral areas of the village.

1.3 Application Overview

The MAST application provides a suite of applications to support collection and management of land rights information. The Mobile Application captures land rights information in the field and a back-end land rights data management infrastructure application contains tools to manage an inventory of land information.

The initial pilot of the MAST application has been implemented in Ilalasimba, Iringa, Tanzania. An update to the application was completed before implementation in the second (Itagutwa) and third (Kitayawa) pilot villages. The key components of the MAST framework are:

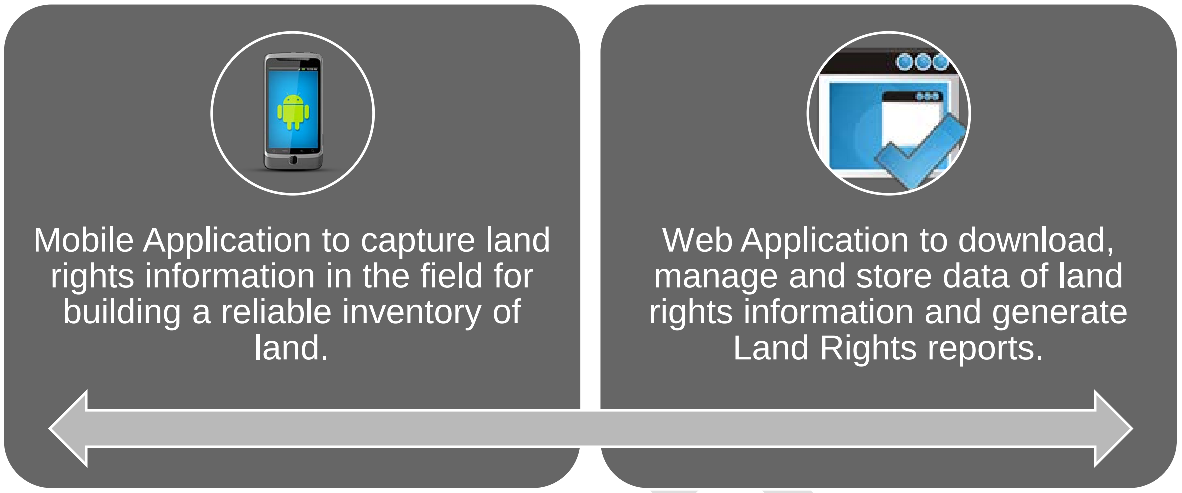
- **Mobile Data Capture Application** – On the front-end, the MAST framework is an Android-based Mobile Application that is focused on the capture of land rights information (spatial, alphanumeric, and multimedia). The Mobile Application allows for the collection of data without being connected to a central cloud-based server in offline mode. Data is collected and stored on users' handheld device, and once the user is within range of an internet connection, data can be synced to the server.

- **Land Rights Data Management (Web) Application** – On the back-end, the MAST framework has a Web Application which provides the ability to configure the Mobile Application, manage data collection projects, and manage land rights information data that has been collected in the field. Key modules of the Web Application are:

    - Configuration Tool – The web-based configuration tool is used to configure the front-end of the land rights data collection Mobile Application. In the configuration tool, adjustments can be made to the attribute fields of the data collection form. This widens the adaptability of the land rights data collection form to multiple environments by configuring the mobile applcation to meet the specific needs of the area, for a specific project.

    - Administration Tool – The Administration tool provides the facility to manage users and roles; to import and configure data layers; configure layer groups; configure survey projects; and configure master attributes that can be used in projects. This module facilitates creation and configuration of survey projects, and the association of layer groups and users.

    - Data Management Infrastructure – The data collected on mobile devices is transferred to a cloud-based data management application, which provides tools to download, manage, and store data of land rights information. It also provides a mapping tool as well as reporting components so that required Land Rights reports can be generated.


Mobile Application to capture land rights information in the field for building a reliable inventory of land.
Web Application to download, manage and store data of land rights information and generate Land Rights reports.

2 OVERVIEW OF MAST APPLICATION SUITE

The MAST system consists of the following sub-systems:

1. Mobile Application to Secure Tenure
2. Land Rights Data Infrastructure Web Application
    - Administration Tool
    - Mobile Configuration Tool
    - Land Rights Data Management Tool

This section lists the functional overview of various components and tools of the Mobile and Web Application, which describe the general functions of the MAST software system.

2.1 Workflow of the Mobile Data Capture Application

The Mobile Application of the MAST suite provides the facility to capture spatial, attribute and multimedia data in the field. Figure 1 depicts the high level workflow of the Mobile Application of the MAST system, along with the user interaction with the different components of the system.

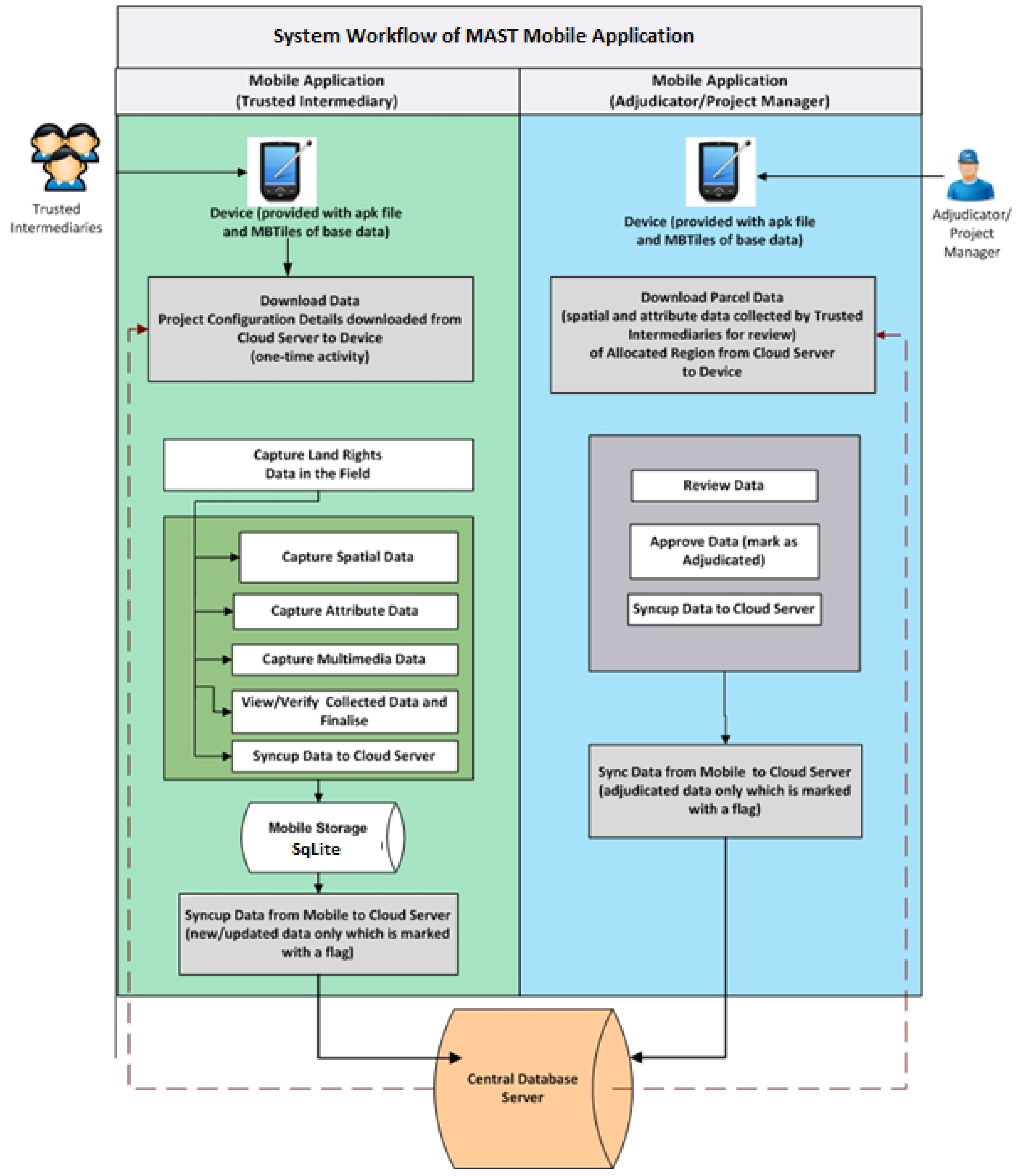

System Workflow of MAST Mobile Application
Mobile Application (Trusted Intermediary)
Mobile Application (Adjudicator/Project Manager)
Trusted Intermediaries
Device (provided with apk file and MBTiles of base data)
Download Data Project Configuration Details downloaded from Cloud Server to Device (one-time activity)
Capture Land Rights Data in the Field
Capture Spatial Data
Capture Attribute Data
Capture Multimedia Data
View/Verify Collected Data and Finalise
Syncup Data to Cloud Server
Mobile Storage SqLite
Syncup Data from Mobile to Cloud Server (new/updated data only which is marked with a flag)
Adjudicator/ Project Manager
Device (provided with apk file and MBTiles of base data)
Download Parcel Data (spatial and attribute data collected by Trusted Intermediaries for review) of Allocated Region from Cloud Server to Device
Review Data
Approve Data (mark as Adjudicated)
Syncup Data to Cloud Server
Sync Data from Mobile to Cloud Server (adjudicated data only which is marked with a flag)
Central Database Server

Figure 1: Workflow of the Mobile Data Capture Application

Trusted Intermediaries or para-surveyors (individuals that are trained specifically to capture land rights information with the MAST application) will utilize the mobile data capture application to capture land rights information (consisting of spatial, attribute and multimedia data) of spatial units in field.

- Users will download the data that has been configured for the project, before the initiation of data capture.

- Users will download the project configuration information (as a one-time activity) in the device before initiation of data capture process.
- User will also download the base data for the allocated project as a onetime activity on mobile device.

The download of the data provides the basis for data capture. Once the data capture is completed, data will be synced to the back-end cloud server (automatically and also via a user initiated event of data syncup).

At this time, an adjudicator (authorized user with authority) will download the collected data on their device for review and adjudication. Using the mobile data capture application, he/she will be able to review and approve the data of spatial units collected by trusted intermediaries.

A Project Manager and other users can also utilize the same application for data review.

2.2 Workflow of Land Rights Data Infrastructure Web Application

The Web Application of the MAST system provides functionalities to manage the users and data layers, configure a data collection project, and administer data that has been collected in the field. The following figure depicts a workflow of the Web Application starting from user creation to final generation of reports. It is configured so that different users will have access privileges to different components of the Web Application as depicted in the workflow. A designated System Administrator will have access to all the components/features of the Web Application.

Different functionalities provided in the Web Application are categorized into the following modules which can be accessed by authorized users for management of survey project and data collected in the field and then generation of reports/certificates on approved data:

- Administration Tool
- Mobile Configuration Tool
- Data Management Tool
- Reporting Tool

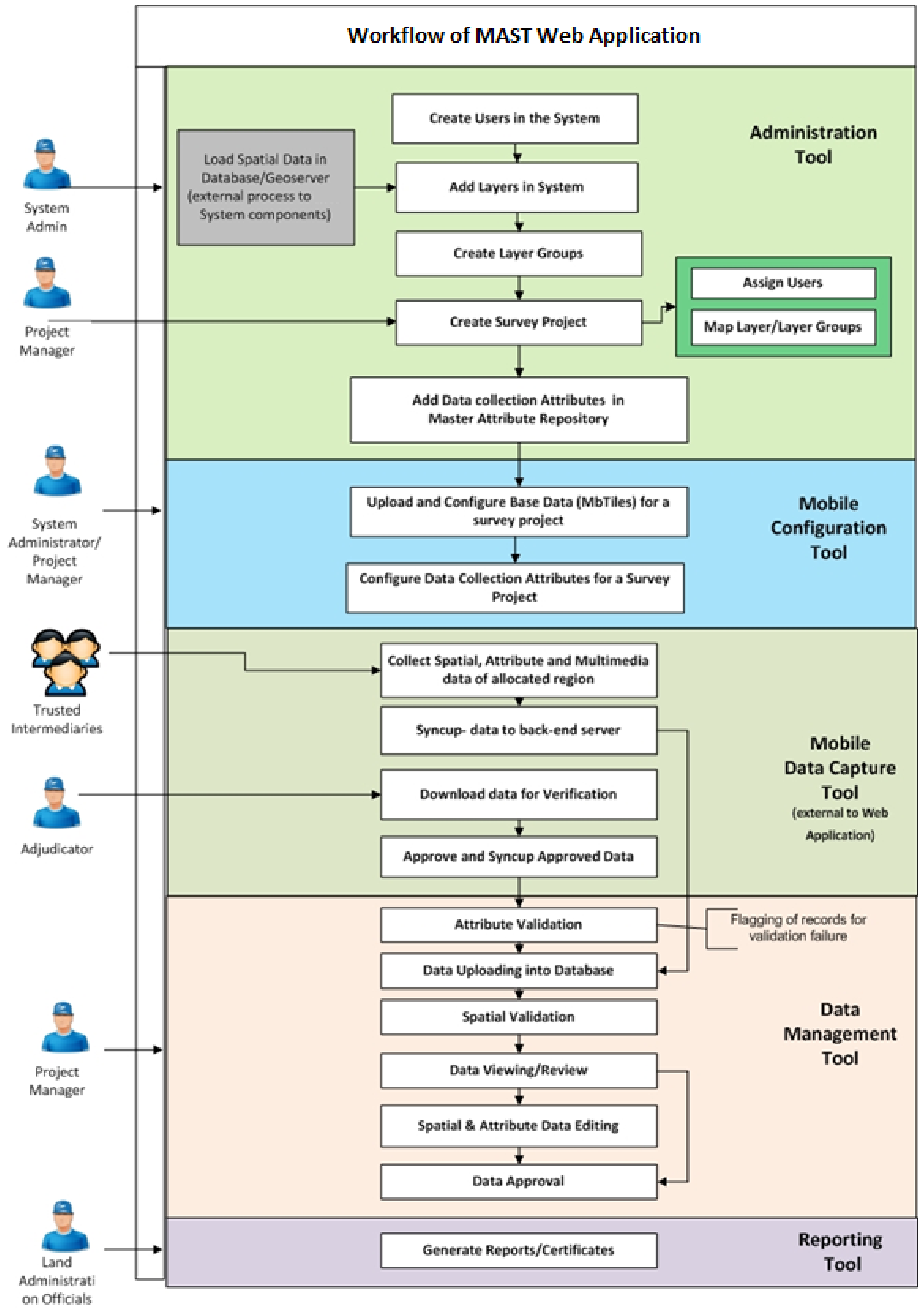

Workflow of MAST Web Application
System Admin
Project Manager
System Administrator/ Project Manager
Trusted Intermediaries
Adjudicator
Project Manager
Land Administration Officials
Load Spatial Data in Database/Geoserver (external process to System components)
Create Users in the System
Add Layers in System
Create Layer Groups
Create Survey Project
Assign Users
Map Layer/Layer Groups
Add Data collection Attributes in Master Attribute Repository
Administration Tool
Upload and Configure Base Data (MbTiles) for a survey project
Configure Data Collection Attributes for a Survey Project
Mobile Configuration Tool
Collect Spatial, Attribute and Multimedia data of allocated region
Syncup- data to back-end server
Download data for Verification
Approve and Syncup Approved Data
Mobile Data Capture Tool (external to Web Application)
Attribute Validation
Flagging of records for validation failure
Data Uploading into Database
Spatial Validation
Data Viewing/Review
Spatial & Attribute Data Editing
Data Approval
Data Management Tool
Generate Reports/Certificates
Reporting Tool

Figure 2: Workflow of the Web based Data Infrastructure Application

- **Administration Tool** – This tool will be accessed by the System Administrator to create and manage users who will be accessing the system. It will also provide the facility to add/configure spatial data layers in the application, which have been loaded and configured in Geoserver. It will also facilitate the creation of layer groups.

  Whenever a new data collection project is initiated, a new survey project is created in the system which will then be mapped to the users who will have permission to access and work on the survey project. Layers will also be configured in the survey project which will be accessible in the Data Management tool (which has a Map Viewer component) of the Web Application when a specific project is accessed in it.

- **Master Attribute Repository** – The administration user will be able to manage the master data and configure data collection attributes. The user will have the ability to define custom attributes. Attributes which will be then mapped to different survey projects for data collection using the mobile configuration tool.

- **Mobile Configuration Tool** - Once the survey project is created in the system using the Administration Tool, an authorized user (Project Manager of the designed survey project) can configure the data that will be transferred to the Mobile Application. The MAST Configuration Tool will provide the facility to:
  - Upload and configure base data layers (in the form of MBTiles) that will be downloaded by the Mobile Application users.
  - Configure the data collection attributes which will be collected in the field in a specific survey project. The desired attributes for a project will need to be specified from the master list of attributes.

- **Data Management Tool** – The MAST Web Application provides a Data Management Tool which will be utilized once the data is collected in the field and data is synced back to the cloud-based server. Once uploaded from the mobile device, all data will be stored in the centralized cloud-based server. Using the Data Management Tool, authorized users will be able to validate, view, edit, and approve the land rights data collected in the field.
  - Web Mapping Tools –a set of tools that perform basic to advanced functions based on user rights. These tools include:
    - Layer Manager – Enable/disable layer view, set transparency
    - Navigation Tools
    - Query builder
    - Spatial Editing
    - Attribute Editing
    - Export
    - Print
  - Reporting Tool – used to generate reports, and ultimately land rights certificates.

2.3 Functionality Role Matrix

This section provides an overview of the different functional roles in the system.

User Role	Group Name	Detail Description
System Administrator	Manager	The System Administrator will have full access permissions through the whole application including all data and will be in charge of system maintenance. They will be responsible for master data management, managing projects, managing users, and allocating roles and responsibilities to the users. Will also be able to configure data collection forms for specific survey projects.
Trusted Intermediary	Data Collector	Trusted Intermediaries will collect land rights data in the field using mobile devices. They will capture spatial, attribute, and multimedia information of spatial units in the field. They will also verify the data before transferring it to the server of the Land Rights Data Management Application.
Adjudicator	Data Collector	Adjudicator will utilize the Mobile Application to review, approve, and validate the collected data. They will download the data collected by Trusted Intermediaries from the server for verification and validation. After validation, they will re-upload the data to the server.
Land Administration Official	Designated User	Land Administration Officials are government officials in state agencies. They will access the Land Rights Data Management Application to view and print spatial data (maps) and produce reports that are required for the formalization of land rights. They will have full access of data for a designated survey project.
Project Manager	Manager	The Project Manager will have full access to data into the MAST database using Land Rights Data Management Application. They will be able to review and edit data, will have full permissions to add, delete, or modify data. The primary function will be to validate data that has been uploaded to the MAST database.

3 ARCHITECTURE OF MAST

The MAST system consists of the following sub-systems:

- Mobile Application to Secure Tenure
- Land Rights Data Infrastructure Web Application

This section provides details of high level design and architecture of the MAST Mobile and Web Application.

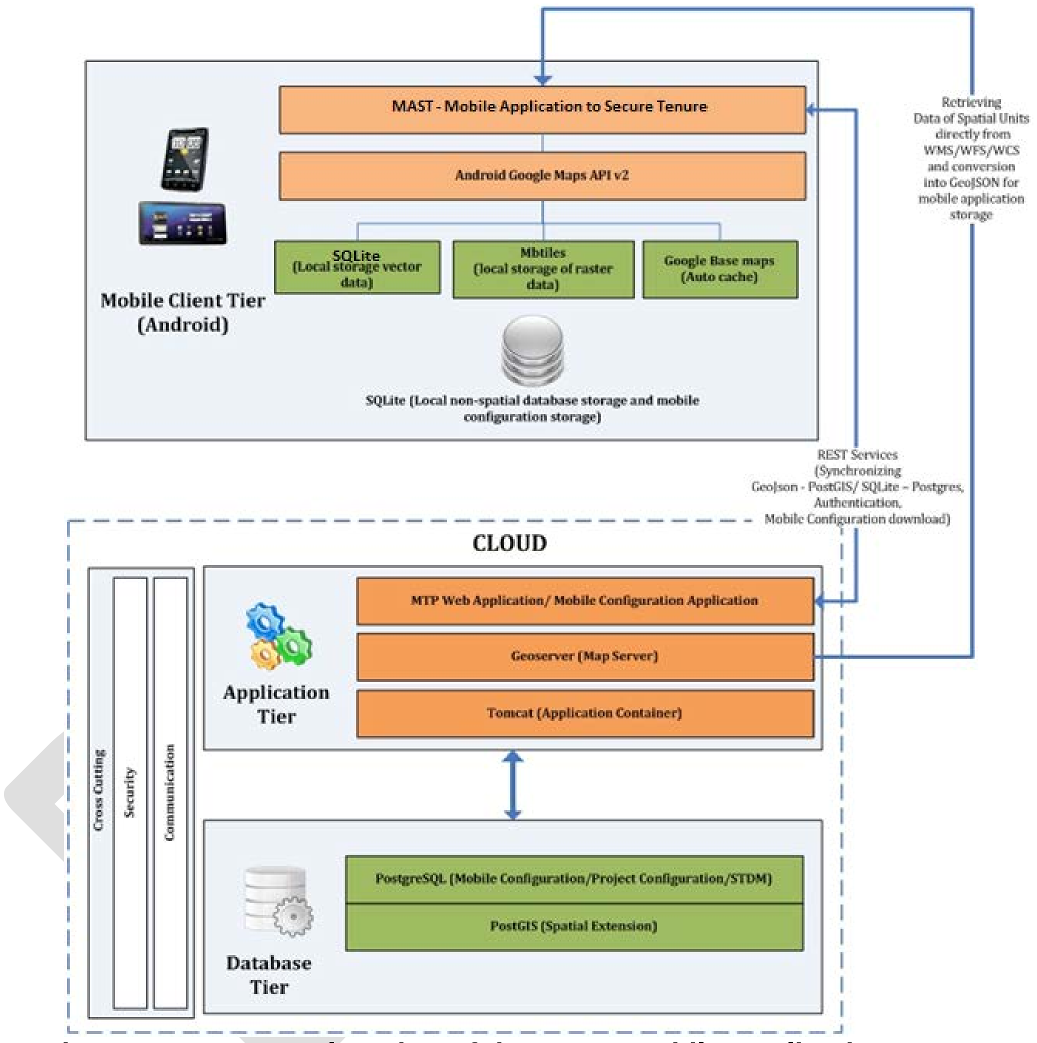

MAST - Mobile Application to Secure Tenure
Android Google Maps API v2
SQLite (Local storage vector data)
Mbtiles (local storage of raster data)
Google Base maps (Auto cache)
Mobile Client Tier (Android)
SQLite (Local non-spatial database storage and mobile configuration storage)
Retrieving Data of Spatial Units directly from WMS/WFS/WCS and conversion into GeoJSON for mobile application storage
REST Services (Synchronizing GeoJson - PostGIS/ SQLite – Postgres, Authentication, Mobile Configuration download)
CLOUD
Cross Cutting
Security
Communication
Application Tier
MTP Web Application/ Mobile Configuration Application
Geoserver (Map Server)
Tomcat (Application Container)
Database Tier
PostgreSQL (Mobile Configuration/Project Configuration/STDM)
PostGIS (Spatial Extension)

Figure 3: Conceptual Design of the MAST Mobile Application Framework

3.1 Architecture of the Mobile Data Capture Application

The MAST Mobile Data Capture framework consists of tools for offline spatial and textual data capturing and editing and connects to custom rest based services to download the mobile configuration data on mobile devices and allows for syncing of the collected data to the server. Spatial Editing and info functionalities are integrated into the using open source JTS library. Layer Manager is built to give user accessibility to toggle between layers. The MAST Mobile Data Capture framework also provides support for viewing application in multiple languages, customizing capture data features and changing colors of captured features according to the user preferences. It also provides accessibility to authenticate user using credentials created on the server for a specific project. This prevents any non-user using the application.

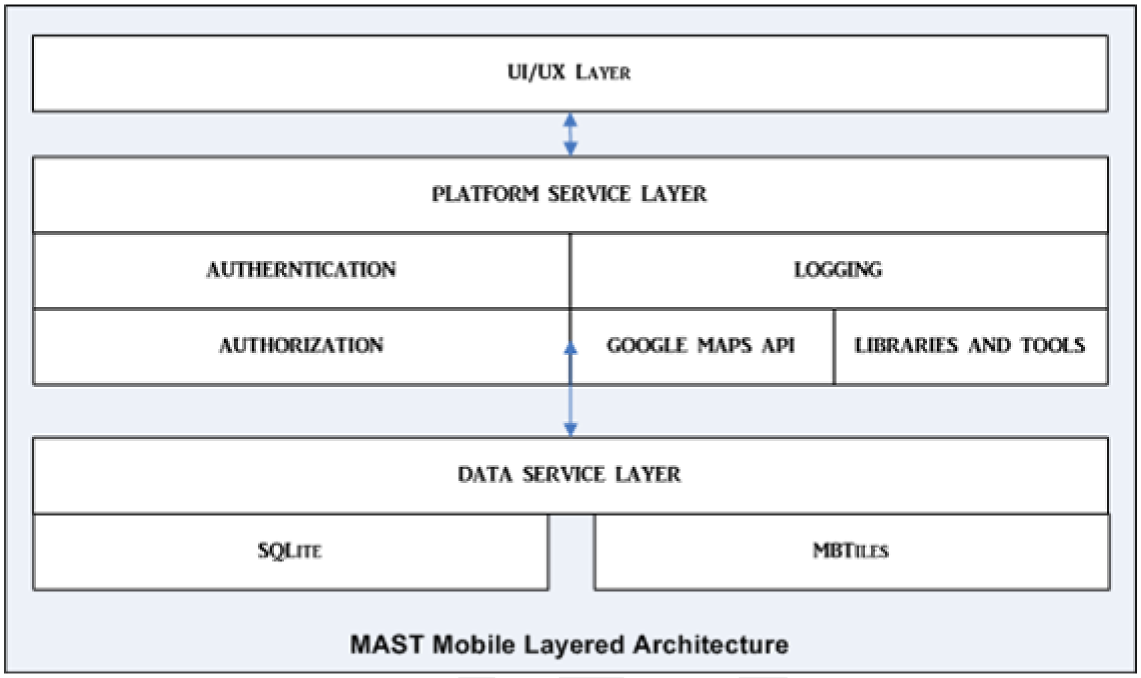


Figure 4: N-Tier architecture of the MAST Mobile application

As shown in the figure above MAST Mobile Application can be broadly divided into three layers:

<u>**UI/UX Layer**</u>

UI Layer, which interacts with the user and is responsible for showing results and taking user inputs. A typical user interface of the application consists of an action bar and the application content area. A UI is defined in an xml file. During compilation, each element in the XML is compiled into equivalent Android GUI class with attributes represented by methods.

<u>**Platform Service Layer**</u>

A Service Layer is responsible for all the operations in the application and connects all three layers. Service layers are responsible for making calls to the rest services for authentication and authorization when online and saving the user details in the data service layer.

A Platform Service Layer also uses the libraries and tools to perform business operations and display maps using a Google Map API for spatial layer display and manipulation in the device.

Google Maps API is also used to show Google satellite maps and offline maps using tile overlays (in .mbtiles format) to provide ease in the data capture in remote locations with very poor internet connectivity. Geometry classes of Marker, Circle, Polyline and Polygon from the Google Maps API are used to show features on the map.

<u>Data Service Layer</u>

Data Service Layer deals with interacting with the underlying database and processing data requests originating from the service layer. Data service layer consists of read-only MBTiles file and SQLite database in WKT format for easy read and write capabilities of geometries along with their attributes. All the form data is also stored in SQLite database for faster fetching and editing of data. The offline data is stored on the device in folders configured for the application.

All the above sub sections combined makes the MAST Mobile Data Capture Application which is wrapped in an .apk file using Android Software development kit (SDK) and Android development toolkit (ADT).

3.2 Architecture of Land Rights Data Infrastructure Web Application

Implementation of the web-based application is based on open standards such as WMS, WFS, WCS from Open Geospatial Consortium implemented using GeoServer, PostgreSQL/PostGIS database for spatial database. It is flexible, reliable and scalable as the underlying architecture is based on stateless protocol using rest based services.

The solution utilizes web based n-tier architecture on Java2 Enterprise Edition (J2EE) platform and PostgreSQL/PostGIS Relational Database. The solution proposed consists of a centralized database server.

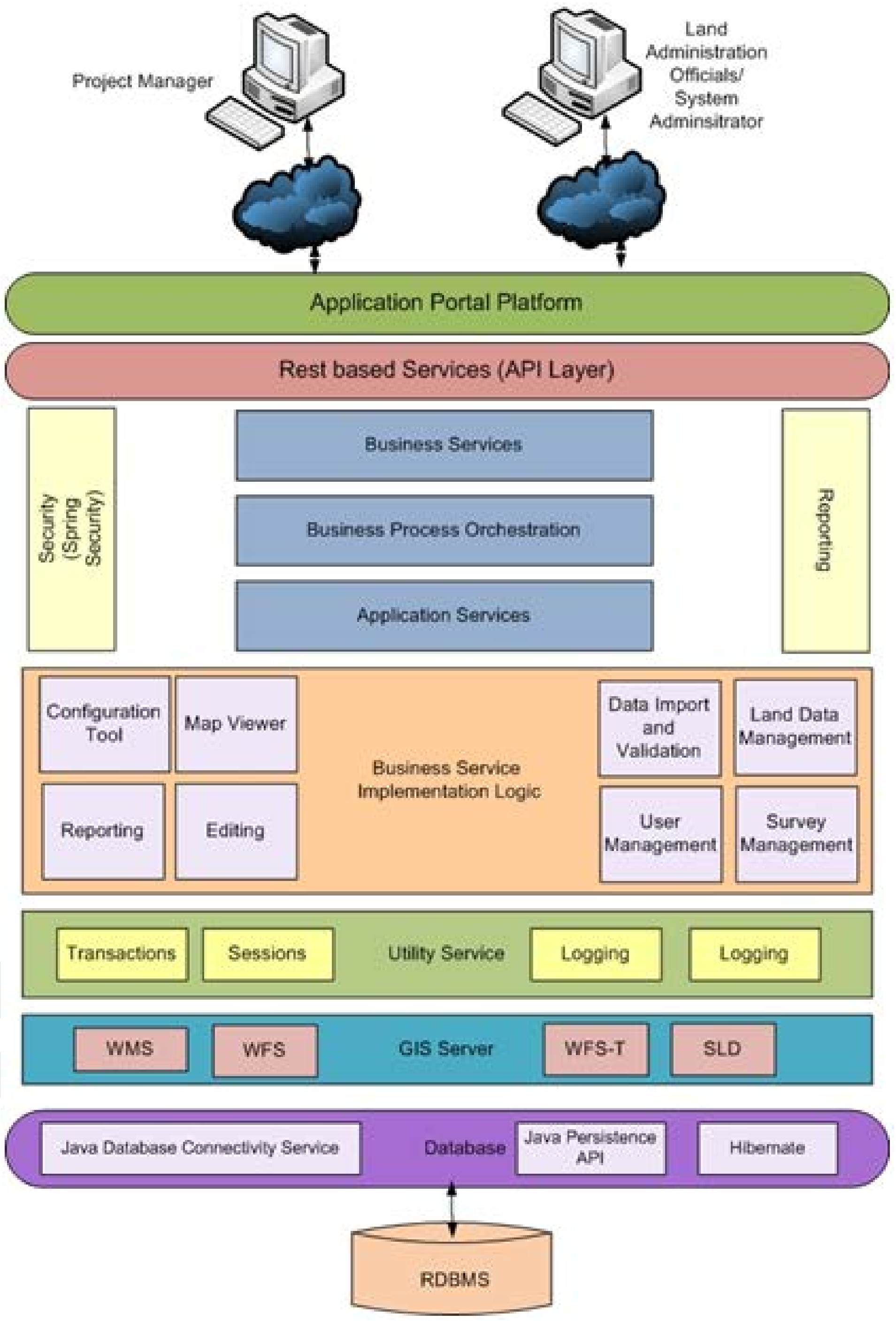


Figure 5: Multi-tier solution of the WebGIS Application

The MAST Web Application is developed as a Web GIS platform to manage, visualize and analyze spatial data through a standard web browser. The MAST Web Application is purely based on OGC standards such as WMS, WFS and WPS and can use any OGC Compliant commercial or Open Source web mapping engines such as GeoServer.

Presentation/ Web Tier: It is the user interface which is responsible for gathering inputs from the user and passing the same to the business layer for processing. The presentation layer ensures that the communications passing through are in the appropriate form for the recipient business objects in the business tier. In the proposed system, the user interface constitutes this tier.

- This tier is based on XHTML/ CSS and interact with the business tier via a set of Java Server Pages and JavaScript Libraries.
- This interaction will leverage on the AJAX (Asynchronous JavaScript and XML) technology to give a seamless experience to the user. This will also enhance the response time as only the relevant dataset will be fetched from the server at any given time.
- This implementation is scalable, platform-independent and complies with the web standards.

Service Layer: The Service Layer implements service level requirements through which the solution achieves scalability, manageability, maintainability and extensibility. In MAST, the service layer allows following flexibility because it:

- Enables the developer to make a clear distinction between web type activity best suited to be done in controller or a generic business logic that is not web-related;
- Enables testing of service enabled business logic separately from controller logic; and
- Enables transactional behavior, and encompasses multiple data objects that should be part of same transaction.

Business Tier: The Business Logic Tier implements the business rules for the system, but it treats data as objects and is not concerned with how the data is stored or displayed. When the client tier requests some information, the Business Logic Tier manages that request, deciding what information to retrieve and whether the client is entitled to that piece of information. When the client performs some action on a piece of information, such as updating a field, adding a new object, or deleting an object, again it is the responsibility of the Business Logic Tier to decide whether the client is entitled to make the change, and making the change if so, in a manner that is consistent with the requirements of the system.

In MAST, two engines – a mapping engine and a data computation engine – constitute this tier:

- The Mapping Engine will interact with the Web Tier and the Database Tier through OGC-Compliant Web Mapping Service (WMS) and Web Feature Service (WFS). This ensures faster performance, interoperability between different systems and thus, unlimited scalability
- The Data Exchange Engine process the AJAX requests from the Web Tier and return the response from the Integration Tier back to the Web Tier. This engine will be custom built based on the requirements of the project.

Data Access Layer: The Data Access Tier lives between the Business Logic Tier and the Data Tier and handles the translation from data in its native format, such as SQL tables or XML elements, to a form more suitable for the business logic tier, such as objects. The integration tier is often referred to as a data abstraction layer because it allows the Business Logic Tier to treat the data in an abstract manner; that is, it can treat the data as objects without being concerned with the details of how the data is actually stored.

Database Tier: This consists of the environment that allows persistence of the user information – both lookup and computed date. Physical implementation of this layer could be via files on the system or databases.

This enables easy change of the data tier to a completely different database server application or from a database server to a set of XML files. If the data tier is changed, the only modification required will be in the integration tier and the rest of the application will work just as before without any modification.

The Database Tier consists of an object relational mapping framework that maps the underlying PostgresSQL relational database. The Database Tier maps Java classes with underlying relational database tables with high level object handling functions.

The MAST infrastructure uses PostgreSQL an open source RDMS as backend database. Spatial data is stored using PostGIS a spatial extension of PostgreSQL.

3.3 Overall Architectural Considerations

3.3.1 Data Collection

The MAST Mobile Application is used to collect data in offline mode. The spatial data is stored in the SQLite database in WKT format. The form data associated with a parcel is also stored in the SQLite database. This storage method helps the device to work seamlessly in offline mode given that fetching and editing the data is much faster as compared to parsing xml or GeoJson-based storage techniques.

To ease in the data collection, offline spatial data configured through web is stored in local storage and is fetched as tiles to provide an uninterrupted base map even with no internet connectivity.

3.3.2 Data Management

3.3.2.1 Data Loading

MAST web infrastructure

A variety of spatial data can be loaded to MAST web infrastructure by publishing them in GeoServer.

- GeoServer supports a variety of vector and raster data formats (refer to data formats section).
- In GeoServer data can be published from either flat files like shape files or spatial tables from RDBMS like PostGIS, Oracle spatial or Microsoft sq server spatial.
- Because the MAST web infrastructure is built over PostgreSQL/PostGIS we would refer to the methodology adopted to load data to spatial tables in PostGIS.
- PGADMIN module of PostgreSQL has a plugin (PostGIS shapefile and DBF loader) to load shape files and to load data to PostGIS table.
- Shape files can be directly stored in GeoServer Data folder and accessed for publishing.

<u>MAST Mobile Application</u>

The MAST Mobile Application required MBTiles created from raster files to act as a base layer for offline mode. MBTiles can be generated from raster images using the Tilemill application from Mapbox. Once MBTiles are created they can copied to the device using a usb connection at a defined location.

3.3.2.2 Data Formats

<u>Data Management Infrastructure</u>

The MAST Web Application supports multiple data formats and follows OGC standards. Data collected from the mobile device will be auto-synced to the back-end server which can then be viewed, edited and approved on the Web Application. The Web Application supports the following data formats:

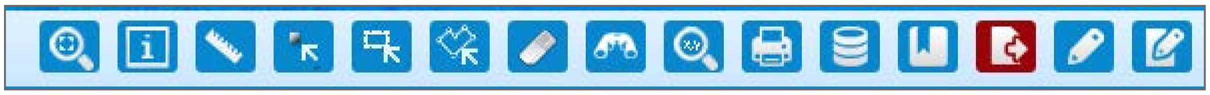
- **Attribute Data** - PostgreSQL – Attribute data is stored in Postgres rdbms tables.

- **Vector Data Sources**
  - PostGIS Database - Spatial data can be loaded to PostgreSQL database with PostGIS extension in tables with geometry datatype columns. The data coordinate reference system should be defined for the specific table for proper usage.
  - PostGIS tables having geometry support that can be published using GeoServer and can be served as the base layer in the MAST web infrastructure.
  - Shapefile - ESRI(tm) Shapefiles (*.shp) - ESRI shape files can be loaded in the GeoServer data folder and then be published in GeoServer to serve as the base layer in MAST web infrastructure.
  - Web Feature Server (NG) -Web feature server extension for GeoServer can be used to publish data in local GeoServer that has already been published in 3rd party servers.

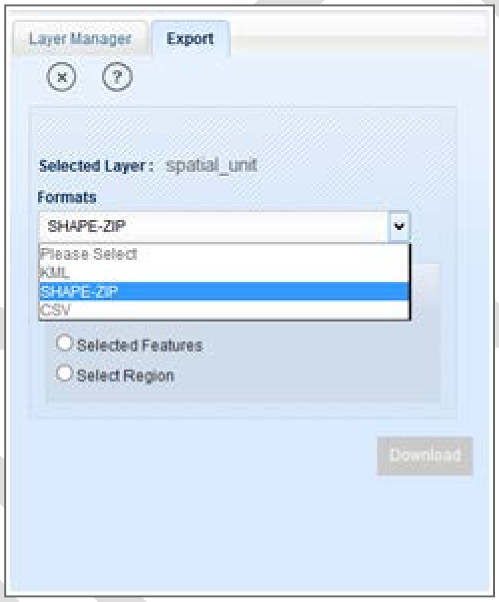
- **Raster Data Sources**
  - ArcGrid - Arc Grid Coverage Format
  - GeoTIFF - Tagged Image File Format with Geographic information
  - MBTiles – MBTiles file format
  - WorldImage - A raster file accompanied by a spatial data file
  - Images with tfw, jgw can be published and used as base layer in MAST web infrastructure.
  - Image Pyramids - Image pyramids can be created from a local desktop/Laptop using GDal library and can be copied to geoserver data folder in the server. These tiles can then be published in geoserver which in turn can be used as base raster tiles in MAST web infrastructure.

3.3.2.3 Data Export

The MAST Web infrastructure provides the functionality to export spatial data.



- A user can access the export or download available in toolbar.
- By clicking the toolbar, a user has access to download data into 3 different formats.
- KML format and shape-zip format for GIS data and csv format for attribute data.


Selected Layer : spatial_unit

Formats: SHAPE-ZIP

Users have several sub-options to export either a complete data set, to select a sub set of features, or select features in an area of interest.

3.3.3 Data Processing

3.3.3.1 Data download & Configuration Base Data on Mobile Device

The MAST web infrastructure application will provide the functionality to configure the project attributes that will be collected for a specific survey project using the mobile data capture application in the field.

Whenever a new survey project is created in the system, the Project Manager/Admin user will configure the attributes which will be utilized in the mobile data capture application to collect land rights information in the field for that specific project. The Project Manager/Admin will also configure offline data to be used in the project in the mobile devices. Data Collection attributes and the associated information to the offline spatial data configured for a project are stored in the server database. While the offline spatial data in form of MBTiles is stored in the file system of the server.

The following figure depicts the workflow of the configuration process of a project with specific attributes that are selected and downloaded from the master repository (in cloud server).

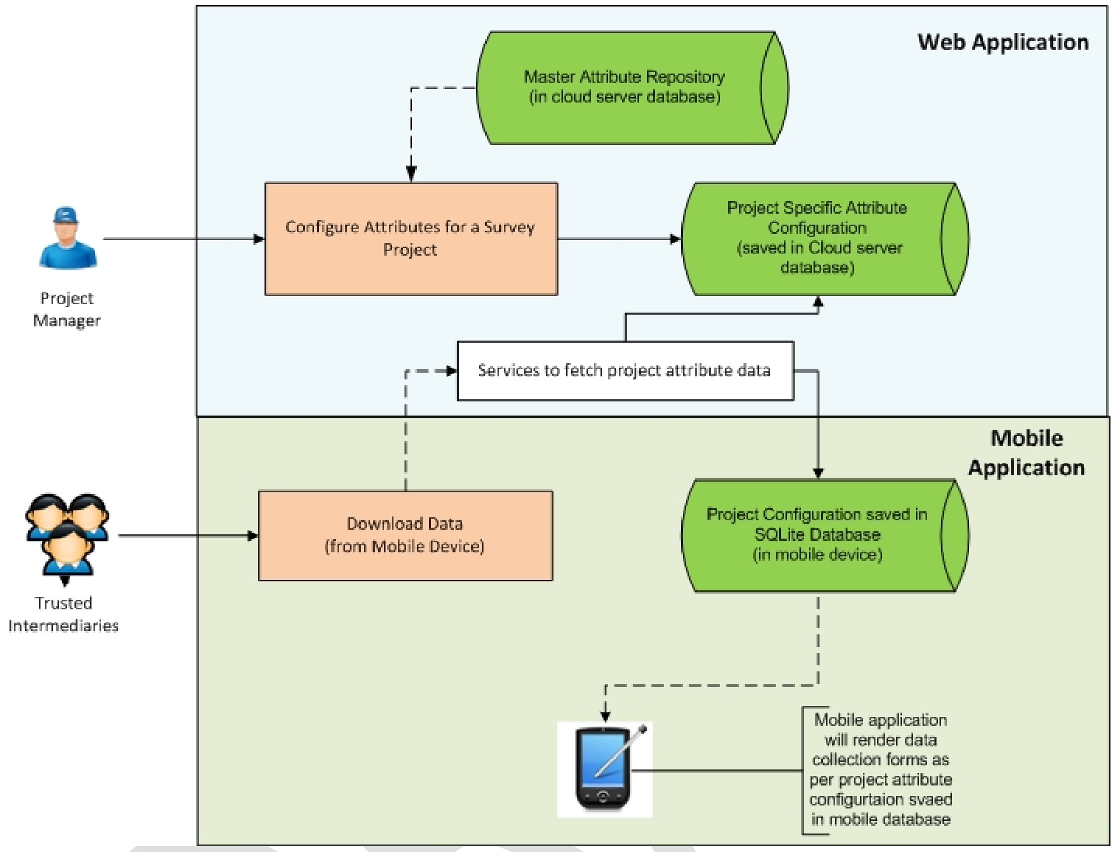

Web Application
Master Attribute Repository (in cloud server database)
Project Manager
Configure Attributes for a Survey Project
Project Specific Attribute Configuration (saved in Cloud server database)
Services to fetch project attribute data
Mobile Application
Trusted Intermediaries
Download Data (from Mobile Device)
Project Configuration saved in SQLite Database (in mobile device)
Mobile application will render data collection forms as per project attribute configurtaion svaed in mobile database

Figure 6: Project Specific Attribute configuration and Data Transfer

When the Mobile Application user initiates data collection work, s/he will need to download the project configuration details by clicking on the download option in Mobile Application. The configured attributes will be fetched from the database server using REST services in JSON format and will be saved to SQLite database in Mobile Application. The configured offline data is downloaded using 'multipart/form-data' encoding type REST services.

Mobile Application attribute pages are rendered according to the configuration saved in the mobile device for specific project. The concept of widgets is used to render dynamic forms in mobile devices according to the type of attributes (Boolean, numeric, text, dropdowns) and the offline data is shown in the layer manager.

Figure 7 below depicts the workflow process of base data download on the mobile device.

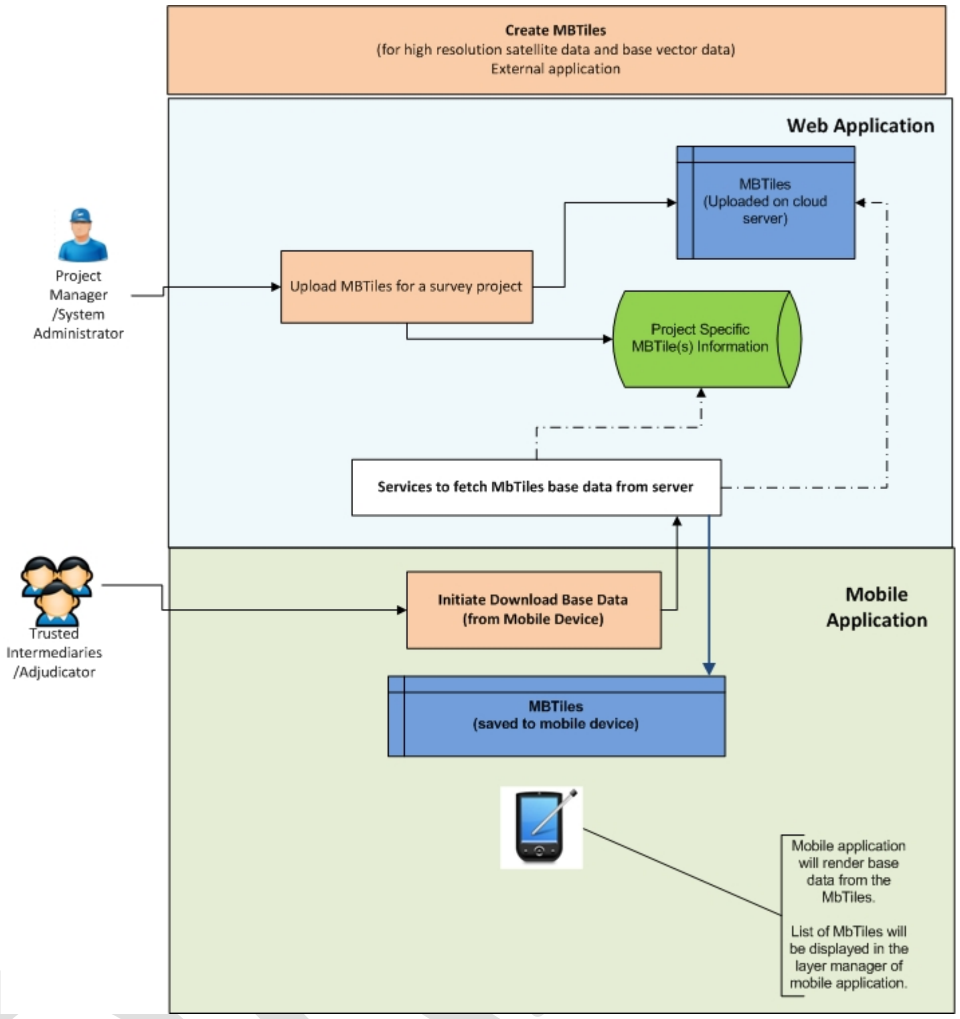

Create MBTiles
(for high resolution satellite data and base vector data)
External application
Web Application
MBTiles
(Uploaded on cloud server)
Project Manager /System Administrator
Upload MBTiles for a survey project
Project Specific MBTile(s) Information
Services to fetch MbTiles base data from server
Trusted Intermediaries /Adjudicator
Initiate Download Base Data (from Mobile Device)
Mobile Application
MBTiles
(saved to mobile device)
Mobile application will render base data from the MbTiles.
List of MbTiles will be displayed in the layer manager of mobile application.

Figure 7: Base data download on mobile device

Other than the base data download, Google satellite data is also shown on mobile device which is auto-cached for the region displayed on the map first time when the internet connection is available and hence can be used subsequent times without internet connection.

3.3.3.2 MBTile Processing

This section provides the key steps involved in creation of MBTiles which need to be pre-created for base data rendering on mobile device.

MapBox is used for the creation of MBTiles, which is a file format for storing map tiles in a single file. MBTiles allows us to render high resolution satellite imagery and vector data in a format that can be rendered in the Mobile Application irrespective of the size of the data as at a time only few tiles are fetched which are to be displayed on the map, instead of the whole data as in conventional formats of spatial data storage.

The following flowchart depicts the process of creating MBTiles, which is a one-time process done for each survey project.

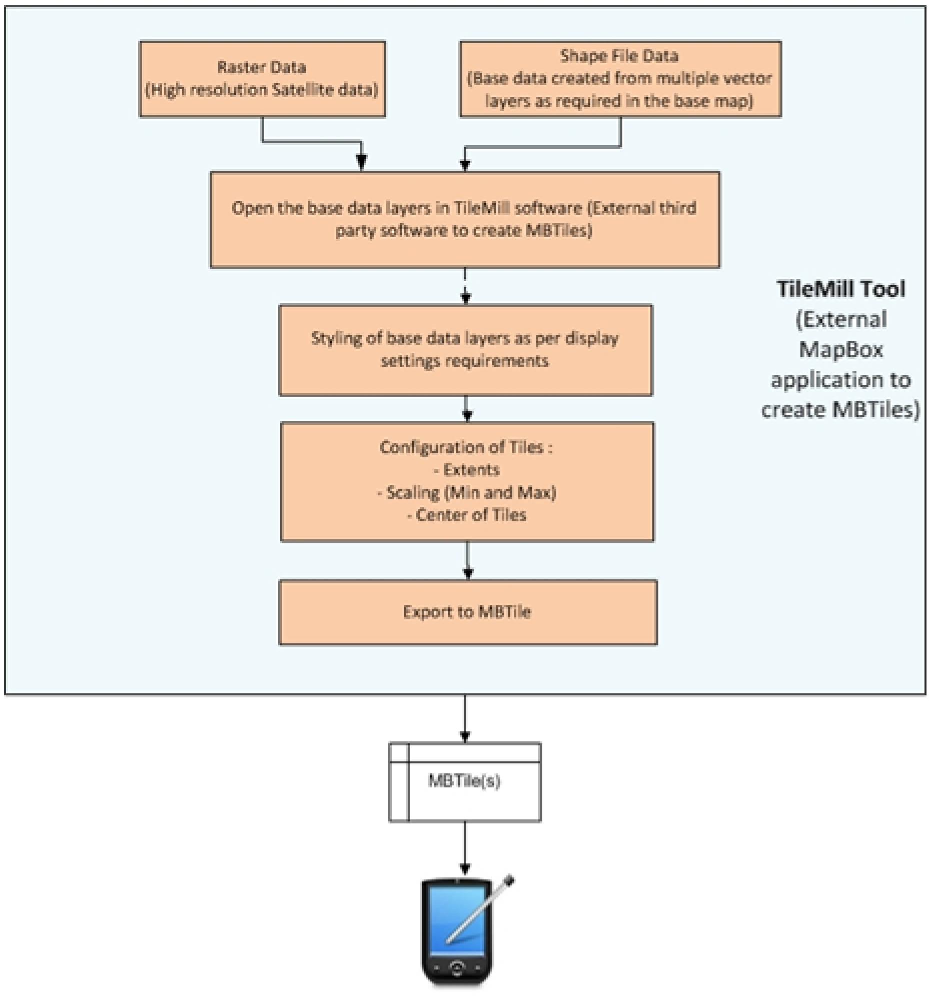

Raster Data
(High resolution Satellite data)
Shape File Data
(Base data created from multiple vector layers as required in the base map)
Open the base data layers in TileMill software (External third party software to create MBTiles)
Styling of base data layers as per display settings requirements
Configuration of Tiles :
- Extents
- Scaling (Min and Max)
- Center of Tiles
Export to MBTile
TileMill Tool
(External MapBox application to create MBTiles)
MBTile(s)

Figure 8: High level process workflow to create MBTiles (back ground layers on map device)

3.3.3.3 GIS mapping

The mapping function for MAST web infrastructure is implemented using a combination of technologies and protocols. The MAST web infrastructure is compliant with Open Geospatial Consortium (OGC). The client side features are developed using OpenLayers which makes it easy to put a dynamic map in any web page.

OpenLayers is a pure JavaScript library used for displaying map data in most modern web browsers, with no server-side dependencies. OpenLayers implements a JavaScript API for building rich web-based geographic applications. It interacts with Geoserver to show the GIS layers on the browser.

Navigation, spatial queries, and editing have been developed using a combination of features and functionalities available in OpenLayers client scripting and Geoserver rest services (i.e. WMS and WFS).

3.3.4 Reporting

The MAST web infrastructure reporting tool generates two types of reports: Adjudication Forms and CCRO reports.

Both these reports are currently hard coded into the system. These reports are generated by fetching data from backend database and rendered on a pre-formatted HTML page.

Map components for these reports are created from getmap call to GeoServer which returns image for the supplied extents. These images are then bound to the page to show parcel geometry in the report html img tag.

3.3.5 Security

3.3.5.1 MAST Mobile Application

The MAST Mobile Application works on the concept of one time authentication. This means that the user has to only authenticate the first time s/he starts using the application. The first authentication is done from the server using authentication web service created on the server. Subsequently, the credentials are stored in the local database to ease logging into the application.

3.3.5.2 MAST Web Infrastructure

Application Security (Authentication and Authorization) – The security subsystem uses a spring security framework to provide out-of-the-box authentication and authorization against credentials stored in the database which is configurable without changing the application code. Based on authorization details received (role-based), the subsystem implicitly controls access to the parts of the system by the user.

3.3.6 Platform

3.3.6.1 Mobile Application

For the MAST application, the minimum configuration recommended for Android devices are:

1. Android Version : Android 4.1 (Jelly Bean)
2. RAM : 1GB
3. Internal Storage : 32 GB
4. Camera : 2MP
5. Location capabilities : GPS+GLONASS
6. Battery : 2000 mAh

3.3.6.2 Web Application

Application Server

1. Linux based OS
2. RAM: 8GB
3. Processor : 8 core, 64 bit architecture
4. HDD : 500GB
5. Network speed : Moderate

Database Server

1. Linux based OS
2. RAM: 4GB
3. Processor : 4 core, 64 bit architecture
4. HDD : 100GB
5. Network speed : Moderate

3.3.7 Data Storage

3.3.7.1 Data Storage/Rendering on Mobile Device

MAST uses a base data on the mobile device. The base data will be provided in following forms to facilitate data collection on offline mode:

- Google base data (Auto-cached for specific region when the mobile is connected to the internet)
- High resolution satellite data (MBTiles needs to be created for high resolution base data)
- Vector base data (for roads and other layers) – MBTiles can be created for vector data layers which needs to be rendered in the Mobile Application as base data layers

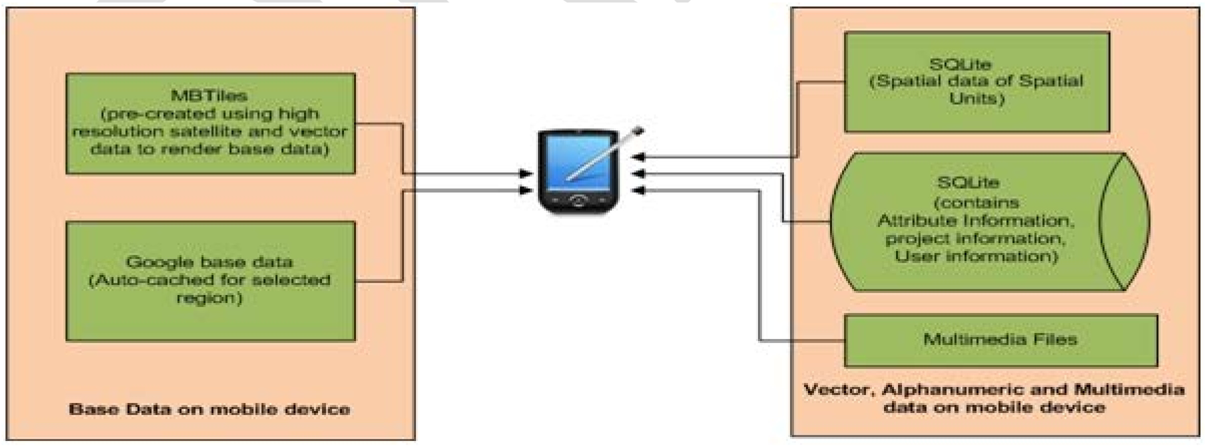

MBTiles
(pre-created using high resolution satellite and vector data to render base data)
Google base data
(Auto-cached for selected region)
Base Data on mobile device
SQLite
(Spatial data of Spatial Units)
SQLite
(contains Attribute Information, project information, User information)
Multimedia Files
Vector, Alphanumeric and Multimedia data on mobile device

Figure 9: Data storage in mobile device (for data collection in offline mode)

Other than the base data that is loaded in configuration, the MAST mobile data capture application will manage spatial, alphanumeric and multimedia data of spatial units that are collected in the field; final data that is downloaded from the back-end server for data viewing/reviewing.

These data entities will be stored in the device in the following manner:

- Spatial data – Stored in SQLite database in WKT (Well known text) format.
- Alphanumeric data of spatial units – Stored in SQLite database in tabular structure.
- Multimedia Files – Stored in pre specified application folders in the mobile devices.

In mobile devices the MAST application configures storage of related data in a defined folder structure:

Parent folder: **MAST**

Sub-folders:

1. **Database** : containes sqlite database files
2. **Spatialdata** : all the offline .mbtiles files are stored in this folder
3. **Multimedia** : containes all the video/images captured using the MAST application.

3.3.8 System Performance

To get the best performance from the application the following have been kept in mind during the implementation of the system:

Pure browser based HTML application

The system has been designed as a pure browser based HTML application that doesn't require a plug-in download on browser.

TMS and WMS-C cached map service

Tile Map Service and WMS cache service have been used for quick generation of maps on browser.

Optimizing data access

The MAST web infrastructure uses Java persistence API/hibernate ORM combination. Optimization of data access from the database server is done by using Hibernate caching.

4 MAST SOLUTION INTEGRATION

4.1 MAST Technology Stack

The tables below provide a listing of development tools that have been used to development MAST.

4.1.1 Mobile Data Capture Application

Overall implementation of the Mobile Application is based on following software tools listed below:

Software	Purpose	License/ Freeware	Details
Android Google Maps API v2	Android Application programming interface to develop map based application.	Open source	
Eclipse IDE (with Android Development Toolkit plugin)	Development Environment	Freeware	
MBTiles	Specification for storing tiled map data in SQLite databases	Open File format	MBTiles will be used to display Tiled Raster data on mobile device. These files will be pre-generated using an open source tools for high resolution Satellite images and other base data layers.
SQlite	SQL database Engine for local data storage	Open Source	Embedded SQL database engine in Mobile device for storage of attribute data of spatial units.
Android SDK API level 16 – Level21	Develop android native application	Open Source	

4.1.2 Land Rights Data Management Web Application

Overall implementation of the Web Application will be based on following software's as mentioned below:

Software	Purpose	License
Java/JEE	Programming Language/Development Platform	OpenSource
Hibernate Framework 3.x	Java Framework (ORM) for Mapping object-oriented domain model to the database solution	OpenSource
Spring Framework 3.x	Java Framework (MVC framework and Security)	OpenSource
Javascript/HTML/ JQuery	Client side coding	OpenSource
Open Layers	Client side JavaScript library for building rich web-based GIS applications.	OpenSource
GeoServer	Share Geospatial Data	OpenSource
PostgreSQL/PostGIS	RDBMS	OpenSource
Apache Tomcat 7.x	Application Server for hosting the Web Application.	Open Source

4.2 MAST Solution Integration

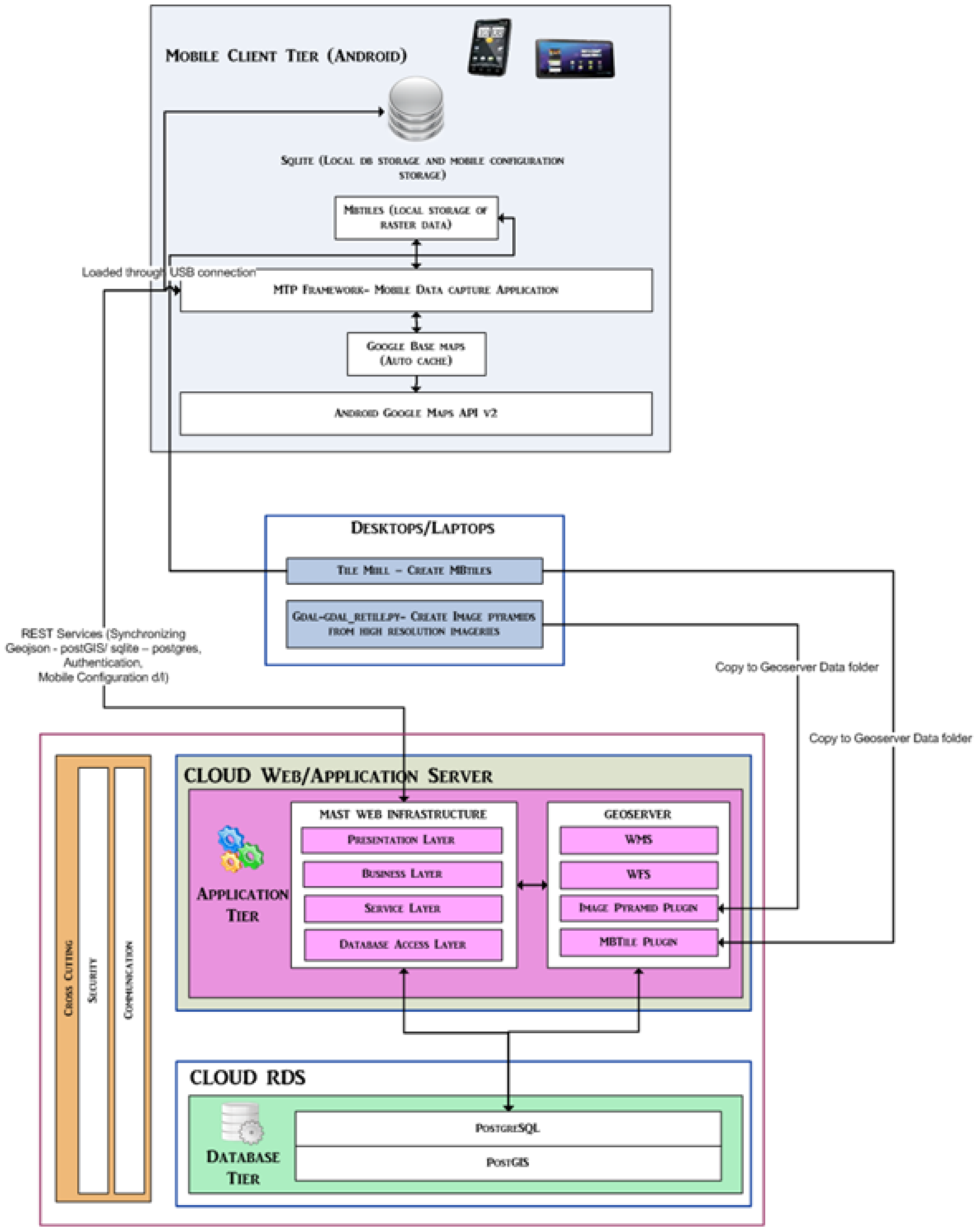

MOBILE CLIENT TIER (ANDROID)
SQLITE (LOCAL DB STORAGE AND MOBILE CONFIGURATION STORAGE)
MBTILES (LOCAL STORAGE OF RASTER DATA)
Loaded through USB connection
MTP FRAMEWORK- MOBILE DATA CAPTURE APPLICATION
GOOGLE BASE MAPS (AUTO CACHE)
ANDROID GOOGLE MAPS API V2
DESKTOPS/LAPTOPS
TILE MILL – CREATE MBTILES
GDAL-GDAL_RETILE.PY- CREATE IMAGE PYRAMIDS FROM HIGH RESOLUTION IMAGERIES
REST Services (Synchronizing Geojson - postGIS/ sqlite – postgres, Authentication, Mobile Configuration d/l)
Copy to Geoserver Data folder
Copy to Geoserver Data folder
CLOUD WEB/APPLICATION SERVER
APPLICATION TIER
MAST WEB INFRASTRUCTURE
PRESENTATION LAYER
BUSINESS LAYER
SERVICE LAYER
DATABASE ACCESS LAYER
GEOSERVER
WMS
WFS
IMAGE PYRAMID PLUGIN
MBTILE PLUGIN
CROSS CUTTING
SECURITY
COMMUNICATION
CLOUD RDS
DATABASE TIER
POSTGRESQL
POSTGIS

Figure 10: Overall Integration of the **MAST web infrastructure and Mobile Application**

The MAST web infrastructure and Mobile Application consists of a suite of the Web and Mobile Applications and third part tools utilized for preprocessing of data.

The diagram in figure 10 depicts high level of integration between these components.

4.2.1 Cloud Web Application Server

The cloud Web Application server and cloud Relational Database service are hosted in the Amazon cloud environment.

- The application server contains Apache Tomcat 7 web container installed in Ubuntu 14.04LTS OS.
- MAST web infrastructure and GeoServer web map server are hosted in this server inside Tomcat servlet container.
- Database configuration and connection pooling is managed by Tomcat and the configurations are available in server.xml.

This server is hosted on public domain and calls are made on this server for the MAST Web Application to work with underlying calls to GeoServer to get spatial layers visible and MAST RDS server to get data from PostgreSQL/PostGIS database.

GeoServer would host the map layers including Vector and Raster layers. GeoServer is deployed in the Amazon Application server along with MAST web infrastructure.

4.2.2 Cloud RDS

- The Cloud RDS server hosts the relational database PostgreSQL/PostGIS.
- Calls are made on this server to get the data from MAST web infrastructure.
- The MAST Android application makes the rest call to the application server which in turn makes the call to the RDS server to get spatial and non-spatial data.

4.2.3 Client Tier (MAST Android Mobile Application)

- The Android MAST application is a native application that synchronizes data by using rest based services available in the MAST web infrastructure.
- The native application stores the data in SqlLite for the local device.
- MBTiles can be downloaded from the MAST web infrastructure as configured with the survey project or it can be copied to the device using a USB connection.

4.2.4 External Data Preprocessing

- MBTiles are created for high resolution satellite images as well as the vectors layer to be displayed as base layers in MAST Android application. Mapbox tool Tilemill is used to create MBTile. MBTiles are created to be used in the MAST Mobile Application as the base layer for offline data capturing.

- An Image Pyramid is created using GDALutility retile.py which is a python script that is used to create image tiles. This tool splits the larger images to smaller chunks that can be easily displayed on a Web Application with better performance. Once pyramids are created they should be copied into the GeoServer data folder to be published and used in the MAST web infrastructure.

4.3 Configuration and Deployment

This section briefly describes the operational hardware requirement, network requirements and data transfer requirements for optimum performance of the MAST web infrastructure application. It also covers lists of prerequisites and their installation, steps to deploy MAST web infrastructure WAR (Web Application aRchive) and server.xml configuration in Apache Tomcat for connecting to the RDS server with connection pooling parameters.

4.3.1 Operational Requirement

Application Instance

Instance Type	:	m3 large
Processor Architecture	:	64Bit
vCPU	:	2
ECU	:	6.5
Memory GiB (RAM)	:	7.5
Instance Storage	:	1 x 32 SSD*6
Network Performance	:	Moderate

RDS Instance

Instance Type	:	m3 medium
Processor Architecture	:	64Bit
vCPU	:	1
ECU	:	3
Memory GiB (RAM)	:	3.75
Instance Storage	:	1 x 4 SSD*6
Network Performance	:	Moderate

Amazon EBS Volume

No of EBS Volumes	:	1
Volume Type	:	Standard
Storage	:	250GB

Data Transfer In/out

100GiB/100GiB

4.3.2 Deployment of Prerequisites & MAST

This section will give a brief introduction on the installation, deployment and configuration of the MAST web infrastructure and GeoServer onto a Linux-based (Ubuntu) server. (Please refer to installation guide **MAST2_development_environment_installation_setup_guide_phase2** for directions on loading prerequisites)

4.3.2.1 MAST Deployment Prerequisites

Core tools/applications used to deploy MAST web infrastructure and GeoServer on to Ubuntu 14.04 LTS are:

- JavaJDK 7 (64-bit)– Java Developer Kit
- PostgreSQL 9.3 (64-bit)
- PostGIS 2.1
- Tomcat 7 (64-bit)

Installing JavaJDK 7

Java JDK 7 can be either be manually installed or can be installed from the Ubuntu repository using this command on Ubuntu Terminal:

```
$sudo add-apt-repository ppa:webupd8team/java
```

Installing PostgreSQL

Ubuntu's default repositories contain PostgreSQL packages. We can then get the Postgres package and a "contrib" package that adds some additional utilities and functionality by using the following command on Ubuntu Terminal:

```
$sudo apt-get update
$sudo apt-get install postgresql postgresql-contrib
```

Installing PostGIS

As mentioned above PostGIS is available in Ubuntu’s default repository. We can get install PostGIS by using the following command on Ubuntu Terminal:

```
$sudo apt-get install postgresql-9.3-postgis
```

Install Tomcat 7

Download Apache Tomcat 7 Server for Linux from http://tomcat.apache.org/download-70.cgi
Use below command in terminal to extract the tar file to tmp folder:

```
$tar xvzf apache-tomcat*.tar.gz -C /tmp/
```

Move the extracted folder to usr folder after providing proper rights:

```
$sudo su -c "chmod -R root:root /tmp/apache-tomcat*
$mv /tmp/apache-tomcat-7 /usr/lib
```

Commands to Start and Stop the Tomcat Server:

Start
/usr/lib/apache-tomcat-7/bin/startup.sh
Stop
/usr/lib/apache-tomcat-7/bin/shutdown.sh

Pre-requisites

You will require administration privileges on your computer to complete this exercise.

4.3.2.2 MAST Web infrastructure/Geoserver Deployment

After all above installations are done, we are ready to deploy the MAST web infrastructure and GeoServer in Tomcat.

Stop the Tomcat Server.

Copy MAST.war and Geoserver.war file to folder /usr/lib/apache-tomcat-7/webapps

Go to folder /usr/lib/apache-tomcat-7/conf folder

Open file server.xml file in edit mode amd modify resource type tag to add rds server details and credentials.

```
<Resource type="javax.sql.DataSource"
        name="jdbc/Cloudburst"
        factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
        driverClassName="org.postgresql.Driver"
                url="jdbc:postgresql:hostname:port/database"
        username="username"
        password="password"

                 testWhileIdle="true"
         testOnBorrow="true"
         testOnReturn="false"
         validationQuery="SELECT 1"
         validationInterval="30000"
```

```
                timeBetweenEvictionRunsMillis="5000"
                maxActive="100"
                minIdle="10"
                maxWait="10000"
                initialSize="10"
                removeAbandonedTimeout="60"
                removeAbandoned="true"
                logAbandoned="true"
                minEvictableIdleTimeMillis="30000"
                jmxEnabled="true"
                jdbcInterceptors="org.apache.tomcat.jdbc.pool.interceptor.ConnectionState;
                    org.apache.tomcat.jdbc.pool.interceptor.StatementFinalizer;
                    org.apache.tomcat.jdbc.pool.interceptor.SlowQueryReportJmx(threshold=10000)"
/>
```

Save server.xml file.

Start the Tomcat Server.

Check deployment

The user needs to check whether Tomcat, the MAST web infrastructure and GeoServer are deployed properly. For this, the user has to open any web browser and type the urls listed below to see whether the **home pages are loaded properly or not.**

Tomcat: http://appserverIP:8080/

Web MAST Infrastructure: http://appserverIP:8080/mast/

GeoServer: http://appserverIP:8080/geoserver

4.3.3 Deployment on Amazon Architecture

The following details pertain to deployment done in Amazon Cloud AWS and RDS.

- Operating system used for pilot deployment is Ubuntu 14.04LTS.
- Amazon Web Services (AWS) is used for hosting MAST web infrastructure. Amazon provides widely available and highly scalable platforms on Cloud.

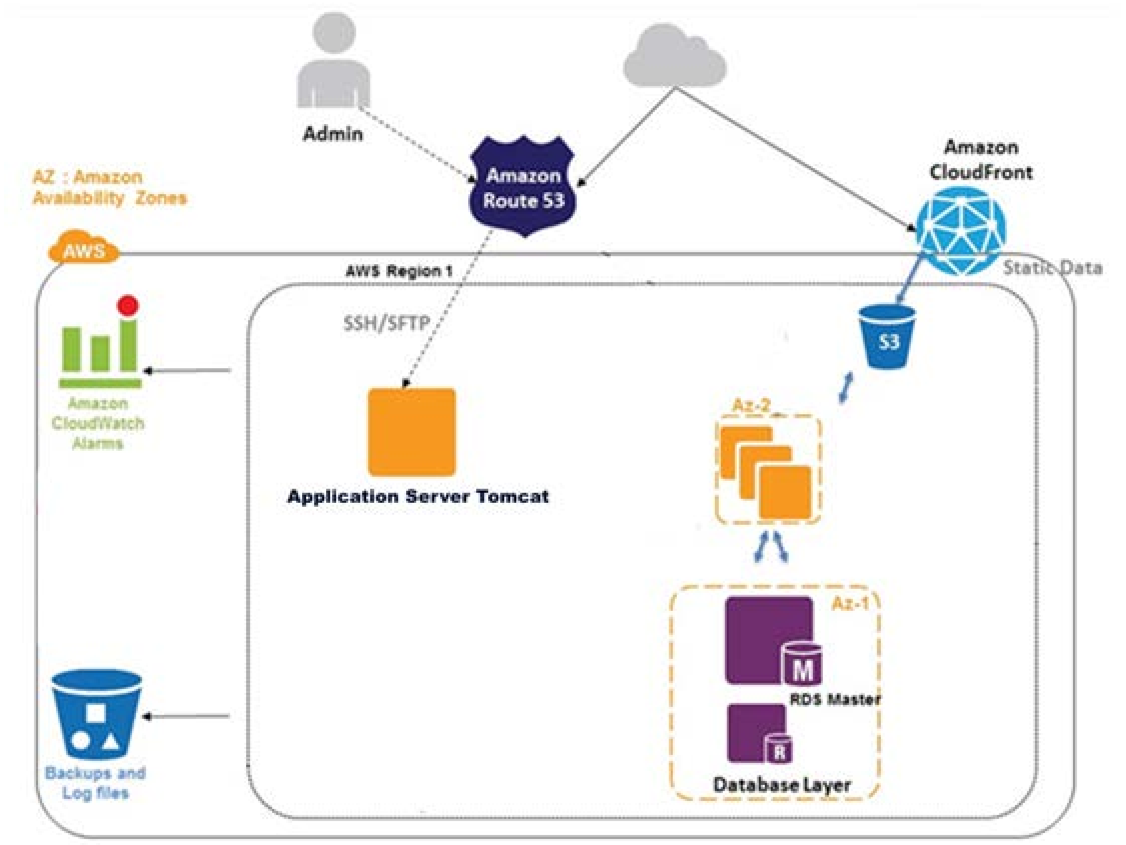


Figure 11: MAST web infrastructure Hosting

In the above diagram the Application Server is used to deploy the MAST web infrastructure and GeoServer in the Apache Tomcat application container. The RDS master server stores the PostgreSQL database with PostGIS extension for storing spatial and textual data.

Amazon Web Services provides the reliable, scalable, secure, and high performance infrastructure required for Web Applications while enabling an elastic, scale-out and scale-down infrastructure to match IT costs in real time as customer traffic fluctuates.

5 DATABASE DESIGN

5.1 MAST Framework (PostgreSQL/PostGIS)

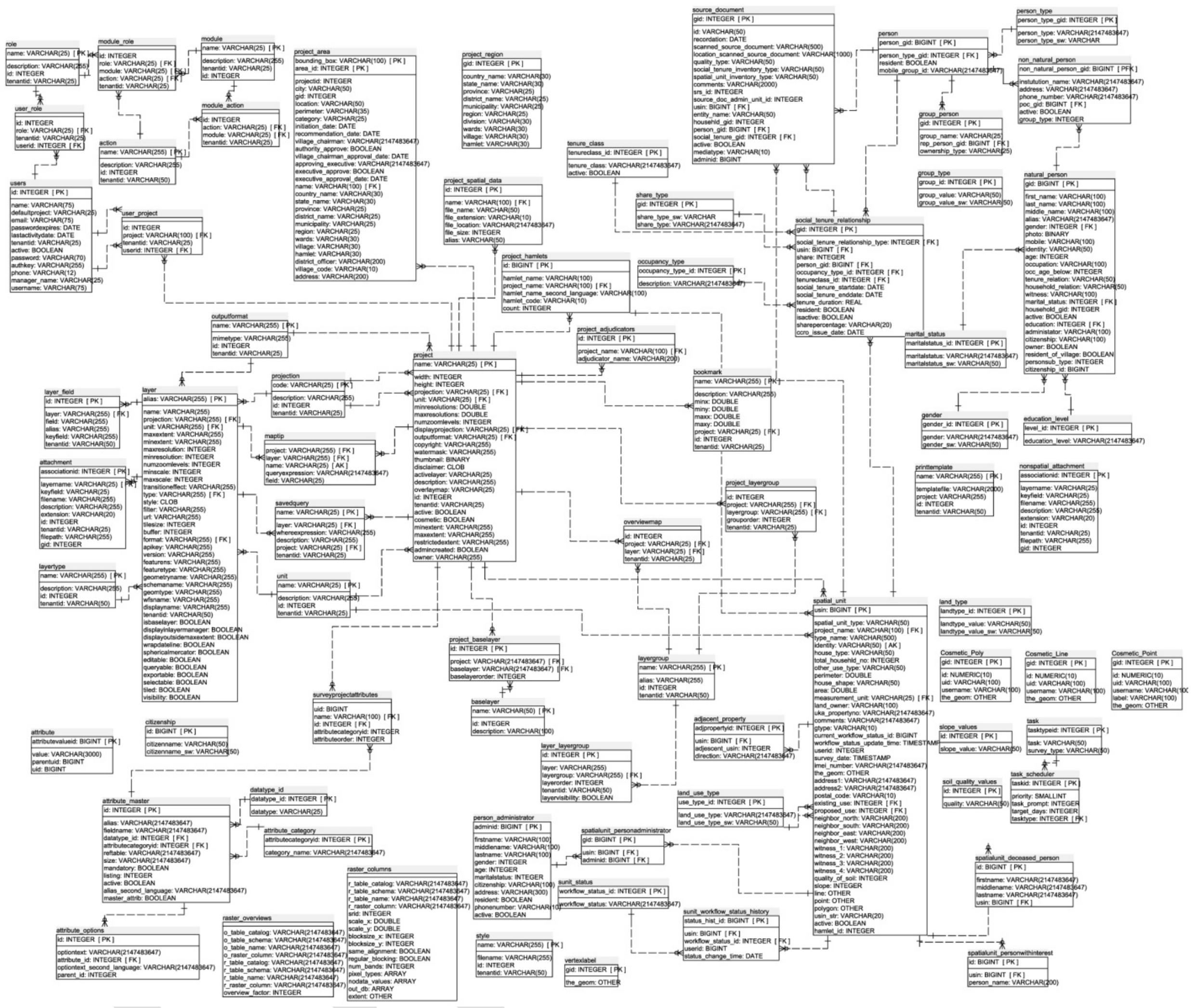


Figure 12: ER Diagram- MAST Web Infrastructure

5.1.1 Table Details

5.1.1.1 action

(Physical Name: action)

Logical Column Name	**Physical Column Name**	**Type**
name (PK)	name	VARCHAR(255)
description	description	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(50)

Referenced By

- module_action referencing (name)
- module_role referencing (name)

5.1.1.2 the adjacent_property

(Physical Name: adjacent_property)

Logical Column Name	**Physical Column Name**	**Type**
adjpropertyid (PK)	Adjpropertyid	INTEGER
usin (FK)	Usin	BIGINT
adjescent_usin	adjescent_usin	INTEGER
direction	direction	VARCHAR(2147483647)

References

- spatial_unit through (usin)

5.1.1.3 attachment

(Physical Name: attachment)

Logical Column Name	**Physical Column Name**	**Type**
associationid (PK)	associationid	INTEGER
layername (FK)	layername	VARCHAR(25)
keyfield	keyfield	VARCHAR(25)
filename	filename	VARCHAR(255)
description	description	VARCHAR(255)
extension	extension	VARCHAR(20)

Logical Column Name	Physical Column Name	Type
id	id	INTEGER
tenantid	tenantid	VARCHAR(25)
filepath	filepath	VARCHAR(255)
gid	gid	INTEGER

References

- layer through (layername)

5.1.1.4 attribute

(Physical Name: attribute)

Logical Column Name	Physical Column Name	Type
attributevalueid (PK)	attributevalueid	BIGINT
value	value	VARCHAR(3000)
parentuid	parentuid	BIGINT
uid	uid	BIGINT

5.1.1.5 attribute_category

(Physical Name: attribute_category)

1. General
2. Person
3. Tenure
4 Multimedia
etc

Logical Column Name	Physical Column Name	Type
attributecategoryid (PK)	attributecategoryid	INTEGER
category_name	category_name	VARCHAR(2147483647)

Referenced By

- attribute_master referencing (attributecategoryid)

5.1.1.6 *attribute_master*
(Physical Name: attribute_master)

Stores all the attributes for lookup purpose

Logical Column Name	**Physical Column Name**	**Type**
id (PK)	id	INTEGER
alias	alias	VARCHAR(2147483647)
fieldname	fieldname	VARCHAR(2147483647)
datatype_id (FK)	datatype_id	INTEGER
attributecategoryid (FK)	attributecategoryid	INTEGER
reftable	reftable	VARCHAR(2147483647)
Size	size	VARCHAR(2147483647)
mandatory	mandatory	BOOLEAN
listing	listing	INTEGER
active	active	BOOLEAN
alias_second_language	alias_second_language	VARCHAR(2147483647)
master_attrib	master_attrib	BOOLEAN

References

- attribute_category through (attributecategoryid)
- datatype_id through (datatype_id)

Referenced By

- attribute_options referencing (id)
- surveyprojectattributes referencing (id)

5.1.1.7 *attribute_options*
(Physical Name: attribute_options)

Logical Column Name	**Physical Column Name**	**Type**
id (PK)	id	INTEGER
optiontext	optiontext	VARCHAR(2147483647)
attribute_id (FK)	attribute_id	INTEGER
optiontext_second_language	optiontext_second_language	VARCHAR(2147483647)
parent_id	parent_id	INTEGER

References

- attribute_master through (attribute_id)

5.1.1.8 baselayer

(Physical Name: baselayer)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(50)
Id	id	INTEGER
description	description	VARCHAR(100)

Referenced By

- project_baselayer referencing (name)

5.1.1.9 bookmark

(Physical Name: bookmark)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(255)
description	description	VARCHAR(255)
minx	minx	DOUBLE
miny	miny	DOUBLE
maxx	maxx	DOUBLE
maxy	maxy	DOUBLE
project (FK)	project	VARCHAR(25)
Id	id	INTEGER
tenantid	tenantid	VARCHAR(25)

References

- project through (project)

5.1.1.10 citizenship
(Physical Name: citizenship)

Logical Column Name	Physical Column Name	Type
id (PK)	id	BIGINT
citizenname	citizenname	VARCHAR(50)
citizenname_sw	citizenname_sw	VARCHAR(50)

5.1.1.11 Cosmetic_Line
(Physical Name: Cosmetic_Line)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	INTEGER
Id	id	NUMERIC(10,0)
Uid	uid	VARCHAR(100)
username	username	VARCHAR(100)
the_geom	the_geom	geometry

5.1.1.12 Cosmetic_Point
(Physical Name: Cosmetic_Point)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	INTEGER
Id	id	NUMERIC(10,0)
Uid	uid	VARCHAR(100)
username	username	VARCHAR(100)
label	label	VARCHAR(100)
the_geom	the_geom	geometry

5.1.1.13 Cosmetic_Poly
(Physical Name: Cosmetic_Poly)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	INTEGER
Id	id	NUMERIC(10,0)
Uid	uid	VARCHAR(100)
username	username	VARCHAR(100)
the_geom	the_geom	geometry

5.1.1.14 datatype_id
(Physical Name: datatype_id)

Logical Column Name	Physical Column Name	Type
datatype_id (PK)	datatype_id	INTEGER
datatype	datatype	VARCHAR(25)

Referenced By

- attribute_master referencing (datatype_id)

5.1.1.15 education_level
(Physical Name: education_level)

Logical Column Name	Physical Column Name	Type
level_id (PK)	level_id	INTEGER
education_level	education_level	VARCHAR(2147483647)

Referenced By

- natural_person referencing (level_id)

5.1.1.16 gender
(Physical Name: gender)

Logical Column Name	Physical Column Name	Type
gender_id (PK)	gender_id	INTEGER
gender	gender	VARCHAR(2147483647)
gender_sw	gender_sw	VARCHAR(50)

Referenced By

- natural_person referencing (gender_id)

5.1.1.17 group_person
(Physical Name: group_person)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	INTEGER
group_name	group_name	VARCHAR(25)
rep_person_gid (FK)	rep_person_gid	BIGINT
ownership_type	ownership_type	VARCHAR(25)

References

- person through (rep_person_gid)

5.1.1.18 group_type
(Physical Name: group_type)

Logical Column Name	Physical Column Name	Type
group_id (PK)	group_id	INTEGER
group_value	group_value	VARCHAR(50)
group_value_sw	group_value_sw	VARCHAR(50)

5.1.1.19 land_type
(Physical Name: land_type)

Logical Column Name	Physical Column Name	Type
landtype_id (PK)	landtype_id	INTEGER
landtype_value	landtype_value	VARCHAR(50)
landtype_value_sw	landtype_value_sw	VARCHAR(50)

5.1.1.20 land_use_type
(Physical Name: land_use_type)

Logical Column Name	Physical Column Name	Type
use_type_id (PK)	use_type_id	INTEGER
land_use_type	land_use_type	VARCHAR(2147483647)
land_use_type_sw	land_use_type_sw	VARCHAR(50)

Referenced By

- spatial_unit referencing (use_type_id)
- spatial_unit referencing (use_type_id)

5.1.1.21 layer

(Physical Name: layer)

Logical Column Name	Physical Column Name	Type
alias (PK)	alias	VARCHAR(255)
name	name	VARCHAR(255)
projection (FK)	projection	VARCHAR(255)
unit (FK)	unit	VARCHAR(255)
maxextent	maxextent	VARCHAR(255)
minextent	minextent	VARCHAR(255)
maxresolution	maxresolution	INTEGER
minresolution	minresolution	INTEGER
numzoomlevels	numzoomlevels	INTEGER
minscale	minscale	INTEGER
maxscale	maxscale	INTEGER
transitioneffect	transitioneffect	VARCHAR(255)
type (FK)	type	VARCHAR(255)
style	style	CLOB
filter	filter	VARCHAR(255)
url	url	VARCHAR(255)
tilesize	tilesize	INTEGER
buffer	buffer	INTEGER
format (FK)	format	VARCHAR(255)
apikey	apikey	VARCHAR(255)
version	version	VARCHAR(255)
featurens	featurens	VARCHAR(255)
featuretype	featuretype	VARCHAR(255)
geometryname	geometryname	VARCHAR(255)
schemaname	schemaname	VARCHAR(255)
geomtype	geomtype	VARCHAR(255)
wfsname	wfsname	VARCHAR(255)
displayname	displayname	VARCHAR(255)
tenantid	tenantid	VARCHAR(50)
isbaselayer	isbaselayer	BOOLEAN
displayinlayermanager	displayinlayermanager	BOOLEAN
displayoutsidemaxextent	displayoutsidemaxextent	BOOLEAN

Logical Column Name	Physical Column Name	Type
wrapdateline	wrapdateline	BOOLEAN
sphericalmercator	sphericalmercator	BOOLEAN
editable	editable	BOOLEAN
queryable	queryable	BOOLEAN
exportable	exportable	BOOLEAN
selectable	selectable	BOOLEAN
tiled	tiled	BOOLEAN
visibility	visibility	BOOLEAN

References

- layertype through (type)
- outputformat through (format)
- projection through (projection)
- unit through (unit)

Referenced By

- attachment referencing (alias)
- layer_field referencing (alias)
- maptip referencing (alias)
- savedquery referencing (alias)

5.1.1.22 layer_field

(Physical Name: layer_field)

Logical Column Name	Physical Column Name	Type
id (PK)	id	INTEGER
layer (FK)	layer	VARCHAR(255)
field	field	VARCHAR(255)
alias	alias	VARCHAR(255)
keyfield	keyfield	VARCHAR(255)
tenantid	tenantid	VARCHAR(50)

References

- layer through (layer)

5.1.1.23 layer_layergroup
(Physical Name: layer_layergroup)

Logical Column Name	Physical Column Name	Type
id (PK)	id	INTEGER
layer	layer	VARCHAR(255)
layergroup (FK)	layergroup	VARCHAR(255)
layerorder	layerorder	INTEGER
tenantid	tenantid	VARCHAR(50)
layervisibility	layervisibility	BOOLEAN

References

- layergroup through (layergroup)

5.1.1.24 layergroup
(Physical Name: layergroup)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(255)
alias	alias	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(50)

Referenced By

- layer_layergroup referencing (name)
- overviewmap referencing (name)
- project_layergroup referencing (name)

5.1.1.25 layertype
(Physical Name: layertype)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(255)
description	description	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(50)

Referenced By

- layer referencing (name)

5.1.1.26 maptip

(Physical Name: maptip)

Logical Column Name	Physical Column Name	Type
project (FK)	project	VARCHAR(255)
layer (FK)	layer	VARCHAR(255)
name	name	VARCHAR(25)
queryexpression	queryexpression	VARCHAR(2147483647)
field	field	VARCHAR(25)

References

- layer through (layer)
- project through (project)

5.1.1.27 marital_status

(Physical Name: marital_status)

Logical Column Name	Physical Column Name	Type
maritalstatus_id (PK)	maritalstatus_id	INTEGER
maritalstatus	maritalstatus	VARCHAR(2147483647)
maritalstatus_sw	maritalstatus_sw	VARCHAR(50)

Referenced By

- natural_person referencing (maritalstatus_id)

5.1.1.28 module

(Physical Name: module)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(25)
description	description	VARCHAR(255)

Logical Column Name	Physical Column Name	Type
tenantid	tenantid	VARCHAR(25)
id	id	INTEGER

Referenced By

- module_action referencing (name)
- module_role referencing (name)

5.1.1.29 module_action

(Physical Name: module_action)

Logical Column Name	Physical Column Name	Type
id	id	INTEGER
action (FK)	action	VARCHAR(25)
module (FK)	module	VARCHAR(25)
tenantid	tenantid	VARCHAR(25)

References

- action through (action)
- module through (module)

5.1.1.30 module_role

(Physical Name: module_role)

Logical Column Name	Physical Column Name	Type
id	id	INTEGER
role (FK)	role	VARCHAR(25)
module (FK)	module	VARCHAR(25)
action (FK)	action	VARCHAR(25)
tenantid	tenantid	VARCHAR(25)

References

- action through (action)
- module through (module)
- role through (role)

5.1.1.31 natural_person
(Physical Name: natural_person)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	BIGINT
first_name	first_name	VARCHAR(100)
last_name	last_name	VARCHAR(100)
middle_name	middle_name	VARCHAR(100)
alias	alias	VARCHAR(2147483647)
gender (FK)	gender	INTEGER
photo	photo	[-2]
mobile	mobile	VARCHAR(100)
identity	identity	VARCHAR(50)
age	age	INTEGER
occupation	occupation	VARCHAR(100)
occ_age_below	occ_age_below	INTEGER
tenure_relation	tenure_relation	VARCHAR(50)
household_relation	household_relation	VARCHAR(50)
witness	witness	VARCHAR(100)
marital_status (FK)	marital_status	INTEGER
household_gid	household_gid	INTEGER
active	active	BOOLEAN
education (FK)	education	INTEGER
administator	administator	VARCHAR(100)
citizenship	citizenship	VARCHAR(100)
owner	owner	BOOLEAN
resident_of_village	resident_of_village	BOOLEAN
personsub_type	personsub_type	INTEGER
citizenship_id	citizenship_id	BIGINT

References

- education_level through (education)
- gender through (gender)
- marital_status through (marital_status)

Referenced By

- non_natural_person referencing (gid)

5.1.1.32 non_natural_person
(Physical Name: non_natural_person)

Logical Column Name	Physical Column Name	Type
non_natural_person_gid (PK) (FK)	non_natural_person_gid	BIGINT
instutution_name	instutution_name	VARCHAR(2147483647)
address	address	VARCHAR(2147483647)
phone_number	phone_number	VARCHAR(2147483647)
poc_gid (FK)	poc_gid	BIGINT
active	active	BOOLEAN
group_type	group_type	INTEGER

References

- natural_person through (poc_gid)
- person through (non_natural_person_gid)

5.1.1.33 nonspatial_attachment
(Physical Name: nonspatial_attachment)

Logical Column Name	Physical Column Name	Type
associationid (PK)	associationid	INTEGER
layername	layername	VARCHAR(25)
keyfield	keyfield	VARCHAR(25)
filename	filename	VARCHAR(255)
description	description	VARCHAR(255)
extension	extension	VARCHAR(20)
id	id	INTEGER
tenantid	tenantid	VARCHAR(25)
filepath	filepath	VARCHAR(255)
gid	gid	INTEGER

5.1.1.34 occupancy_type
(Physical Name: occupancy_type)

Logical Column Name	Physical Column Name	Type
occupancy_type_id (PK)	occupancy_type_id	INTEGER
description	description	VARCHAR(2147483647)

Referenced By

- social_tenure_relationship referencing (occupancy_type_id)

5.1.1.35 outputformat

(Physical Name: outputformat)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(255)
mimetype	mimetype	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(25)

Referenced By

- layer referencing (name)
- project referencing (name)

5.1.1.36 overviewmap

(Physical Name: overviewmap)

Logical Column Name	Physical Column Name	Type
id	id	INTEGER
project (FK)	project	VARCHAR(25)
layer (FK)	layer	VARCHAR(25)
tenantid	tenantid	VARCHAR(25)

References

- layergroup through (layer)
- project through (project)

5.1.1.37 person

(Physical Name: person)

Logical Column Name	Physical Column Name	Type
person_gid (PK)	person_gid	BIGINT

Logical Column Name	Physical Column Name	Type
person_type_gid (FK)	person_type_gid	INTEGER
resident	resident	BOOLEAN
mobile_group_id	mobile_group_id	VARCHAR(2147483647)

References

- person_type through (person_type_gid)

Referenced By

- group_person referencing (person_gid)
- non_natural_person referencing (person_gid)
- social_tenure_relationship referencing (person_gid)
- source_document referencing (person_gid)

5.1.1.38 person_administrator

(Physical Name: person_administrator)

Logical Column Name	Physical Column Name	Type
adminid (PK)	adminid	BIGINT
firstname	firstname	VARCHAR(100)
middlename	middlename	VARCHAR(100)
lastname	lastname	VARCHAR(100)
gender	gender	INTEGER
age	age	INTEGER
maritalstatus	maritalstatus	INTEGER
citizenship	citizenship	VARCHAR(100)
address	address	VARCHAR(300)
resident	resident	BOOLEAN
phonenumber	phonenumber	VARCHAR(10)
active	active	BOOLEAN

Referenced By

- spatialunit_personadministrator referencing (adminid)

5.1.1.39 person_type
(Physical Name: person_type)

Logical Column Name	Physical Column Name	Type
person_type_gid (PK)	person_type_gid	INTEGER
person_type	person_type	VARCHAR(2147483647)
person_type_sw	person_type_sw	VARCHAR(0)

Referenced By

- person referencing (person_type_gid)

5.1.1.40 printtemplate
(Physical Name: printtemplate)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(255)
templatefile	templatefile	VARCHAR(2000)
project	project	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(50)

5.1.1.41 project
(Physical Name: project)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(25)
width	width	INTEGER
height	height	INTEGER
projection (FK)	projection	VARCHAR(25)
unit (FK)	unit	VARCHAR(25)
minresolutions	minresolutions	DOUBLE
maxresolutions	maxresolutions	DOUBLE
numzoomlevels	numzoomlevels	INTEGER
displayprojection (FK)	displayprojection	VARCHAR(25)
outputformat (FK)	outputformat	VARCHAR(25)
copyright	copyright	VARCHAR(255)
watermask	watermask	VARCHAR(255)
thumbnail	thumbnail	[-2]

disclaimer	disclaimer	CLOB
activelayer	activelayer	VARCHAR(25)
description	description	VARCHAR(255)
overlaymap	overlaymap	VARCHAR(25)
id	id	INTEGER
tenantid	tenantid	VARCHAR(25)
active	active	BOOLEAN
cosmetic	cosmetic	BOOLEAN
minextent	minextent	VARCHAR(255)
maxextent	maxextent	VARCHAR(255)
restrictedextent	restrictedextent	VARCHAR(255)
admincreated	admincreated	BOOLEAN
owner	owner	VARCHAR(255)

References

- outputformat through (outputformat)
- projection through (projection)
- projection through (displayprojection)
- unit through (unit)

Referenced By

- bookmark referencing (name)
- maptip referencing (name)
- overviewmap referencing (name)
- project_area referencing (name)
- project_baselayer referencing (name)
- project_layergroup referencing (name)
- project_spatial_data referencing (name)
- savedquery referencing (name)
- spatial_unit referencing (name)
- surveyprojectattributes referencing (name)
- user_project referencing (name)
- project_adjudicators referencing (name)
- project_hamlets referencing (name)

5.1.1.42 project_adjudicators

(Physical Name: project_adjudicators)

Logical Column Name	**Physical Column Name**	**Type**
id (PK)	id	INTEGER
project_name (FK)	project_name	VARCHAR(100)
adjudicator_name	adjudicator_name	VARCHAR(200)

References

- project through (project_name)

5.1.1.43 project_area

(Physical Name: project_area)

Logical Column Name	Physical Column Name	Type
bounding_box (PK)	bounding_box	VARCHAR(100)
area_id (PK)	area_id	INTEGER
projectid	projectid	INTEGER
city	city	VARCHAR(50)
gid	gid	INTEGER
location	location	VARCHAR(50)
perimeter	perimeter	VARCHAR(35)
category	category	VARCHAR(25)
initiation_date	initiation_date	DATE
recommendation_date	recommendation_date	DATE
village_chairman	village_chairman	VARCHAR(2147483647)
authority_approve	authority_approve	BOOLEAN
village_chairman_approval_date	village_chairman_approval_date	DATE
approving_executive	approving_executive	VARCHAR(2147483647)
executive_approve	executive_approve	BOOLEAN
executive_approval_date	executive_approval_date	DATE
name (FK)	name	VARCHAR(100)
country_name	country_name	VARCHAR(30)
state_name	state_name	VARCHAR(30)
province	province	VARCHAR(25)
district_name	district_name	VARCHAR(25)
municipality	municipality	VARCHAR(25)
region	region	VARCHAR(25)
wards	wards	VARCHAR(30)
village	village	VARCHAR(30)
hamlet	hamlet	VARCHAR(30)
district_officer	district_officer	VARCHAR(200)
village_code	village_code	VARCHAR(10)
address	address	VARCHAR(200)

References

- project through (name)

5.1.1.44 project_baselayer
(Physical Name: project_baselayer)

Logical Column Name	Physical Column Name	Type
id (PK)	id	INTEGER
project (FK)	project	VARCHAR(2147483647)
baselayer (FK)	baselayer	VARCHAR(2147483647)
baselayerorder	baselayerorder	INTEGER

References

- baselayer through (baselayer)
- project through (project)

5.1.1.45 project_hamlets
(Physical Name: project_hamlets)

Logical Column Name	Physical Column Name	Type
id (PK)	id	BIGINT
hamlet_name	hamlet_name	VARCHAR(100)
project_name (FK)	project_name	VARCHAR(100)
hamlet_name_second_language	hamlet_name_second_language	VARCHAR(100)
hamlet_code	hamlet_code	VARCHAR(10)
count	count	INTEGER

References

- project through (project_name)

Referenced By

- spatial_unit referencing ()

5.1.1.46 project_layergroup
(Physical Name: project_layergroup)

Logical Column Name	Physical Column Name	Type
id	id	INTEGER

Logical Column Name	Physical Column Name	Type
project (FK)	project	VARCHAR(255)
layergroup (FK)	layergroup	VARCHAR(255)
grouporder	grouporder	INTEGER
tenantid	tenantid	VARCHAR(25)

References

- layergroup through (layergroup)
- project through (project)

5.1.1.47 project_region

(Physical Name: project_region)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	INTEGER
country_name	country_name	VARCHAR(30)
state_name	state_name	VARCHAR(30)
province	province	VARCHAR(25)
district_name	district_name	VARCHAR(25)
municipality	municipality	VARCHAR(25)
region	region	VARCHAR(25)
division	division	VARCHAR(30)
wards	wards	VARCHAR(30)
village	village	VARCHAR(30)
hamlet	hamlet	VARCHAR(30)

5.1.1.48 project_spatial_data

(Physical Name: project_spatial_data)

table to store spatial data upload details

Logical Column Name	Physical Column Name	Type
id (PK)	id	INTEGER
name (FK)	name	VARCHAR(100)
file_name	file_name	VARCHAR(50)
file_extension	file_extension	VARCHAR(10)
file_location	file_location	VARCHAR(2147483647)

Logical Column Name	Physical Column Name	Type
file_size	file_size	INTEGER
alias	alias	VARCHAR(50)

References

- project through (name)

5.1.1.49 projection

(Physical Name: projection)

Logical Column Name	Physical Column Name	Type
code (PK)	code	VARCHAR(25)
description	description	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(25)

Referenced By

- layer referencing (code)
- project referencing (code)
- project referencing (code)

5.1.1.50 raster_columns

(Physical Name: raster_columns)

Logical Column Name	Physical Column Name	Type
r_table_catalog	r_table_catalog	VARCHAR(2147483647)
r_table_schema	r_table_schema	VARCHAR(2147483647)
r_table_name	r_table_name	VARCHAR(2147483647)
r_raster_column	r_raster_column	VARCHAR(2147483647)
srid	srid	INTEGER
scale_x	scale_x	DOUBLE
scale_y	scale_y	DOUBLE
blocksize_x	blocksize_x	INTEGER
blocksize_y	blocksize_y	INTEGER
same_alignment	same_alignment	BOOLEAN

Logical Column Name	Physical Column Name	Type
regular_blocking	regular_blocking	BOOLEAN
num_bands	num_bands	INTEGER
pixel_types	pixel_types	[2003]
nodata_values	nodata_values	[2003]
out_db	out_db	[2003]
extent	extent	geometry

5.1.1.51 raster_overviews

(Physical Name: raster_overviews)

Logical Column Name	Physical Column Name	Type
o_table_catalog	o_table_catalog	VARCHAR(2147483647)
o_table_schema	o_table_schema	VARCHAR(2147483647)
o_table_name	o_table_name	VARCHAR(2147483647)
o_raster_column	o_raster_column	VARCHAR(2147483647)
r_table_catalog	r_table_catalog	VARCHAR(2147483647)
r_table_schema	r_table_schema	VARCHAR(2147483647)
r_table_name	r_table_name	VARCHAR(2147483647)
r_raster_column	r_raster_column	VARCHAR(2147483647)
overview_factor	overview_factor	INTEGER

5.1.1.52 role

(Physical Name: role)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(25)
description	description	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(25)

Referenced By

- module_role referencing (name)
- user_role referencing (name)

5.1.1.53 savedquery
(Physical Name: savedquery)

Logical Column Name	**Physical Column Name**	**Type**
name (PK)	name	VARCHAR(25)
layer (FK)	layer	VARCHAR(25)
whereexpression	whereexpression	VARCHAR(255)
description	description	VARCHAR(255)
project (FK)	project	VARCHAR(25)
tenantid	tenantid	VARCHAR(25)

References

- layer through (layer)
- project through (project)

5.1.1.54 share_type
(Physical Name: share_type)

Logical Column Name	**Physical Column Name**	**Type**
gid (PK)	gid	INTEGER
share_type_sw	share_type_sw	VARCHAR(0)
share_type	share_type	VARCHAR(2147483647)

Referenced By

- social_tenure_relationship referencing (gid)

5.1.1.55 slope_values
(Physical Name: slope_values)

Logical Column Name	**Physical Column Name**	**Type**
id (PK)	id	INTEGER
slope_value	slope_value	VARCHAR(50)

5.1.1.56 social_tenure_relationship

(Physical Name: social_tenure_relationship)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	INTEGER
social_tenure_relationship_type (FK)	social_tenure_relationship_type	INTEGER
usin (FK)	usin	BIGINT
share	share	INTEGER
person_gid (FK)	person_gid	BIGINT
occupancy_type_id (FK)	occupancy_type_id	INTEGER
tenureclass_id (FK)	tenureclass_id	INTEGER
social_tenure_startdate	social_tenure_startdate	DATE
social_tenure_enddate	social_tenure_enddate	DATE
tenure_duration	tenure_duration	REAL
resident	resident	BOOLEAN
isactive	isactive	BOOLEAN
sharepercentage	sharepercentage	VARCHAR(20)
ccro_issue_date	ccro_issue_date	DATE

References

- occupancy_type through (occupancy_type_id)
- person through (person_gid)
- share_type through (social_tenure_relationship_type)
- spatial_unit through (usin)
- tenure_class through (tenureclass_id)

Referenced By

- source_document referencing (gid)

5.1.1.57 soil_quality_values

(Physical Name: soil_quality_values)

Logical Column Name	Physical Column Name	Type
id (PK)	id	INTEGER
quality	quality	VARCHAR(50)

5.1.1.58 source_document
(Physical Name: source_document)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	INTEGER
id	id	VARCHAR(50)
recordation	recordation	DATE
scanned_source_document	scanned_source_document	VARCHAR(500)
location_scanned_source_document	location_scanned_source_document	VARCHAR(1000)
quality_type	quality_type	VARCHAR(50)
social_tenure_inventory_type	social_tenure_inventory_type	VARCHAR(50)
spatial_unit_inventory_type	spatial_unit_inventory_type	VARCHAR(50)
comments	comments	VARCHAR(2000)
srs_id	srs_id	INTEGER
source_doc_admin_unit_id	source_doc_admin_unit_id	INTEGER
usin (FK)	usin	BIGINT
entity_name	entity_name	VARCHAR(50)
househld_gid	househld_gid	INTEGER
person_gid (FK)	person_gid	BIGINT
social_tenure_gid (FK)	social_tenure_gid	INTEGER
active	active	BOOLEAN
mediatype	mediatype	VARCHAR(10)
adminid	adminid	BIGINT

References

- person through (person_gid)
- social_tenure_relationship through (social_tenure_gid)
- spatial_unit through (usin)

5.1.1.59 spatial_unit
(Physical Name: spatial_unit)

Logical Column Name	Physical Column Name	Type
usin (PK)	usin	BIGINT
spatial_unit_type	spatial_unit_type	VARCHAR(50)
project_name (FK)	project_name	VARCHAR(100)
type_name	type_name	VARCHAR(500)
identity	identity	VARCHAR(50)
house_type	house_type	VARCHAR(50)

Logical Column Name	Physical Column Name	Type
total_househld_no	total_househld_no	INTEGER
other_use_type	other_use_type	VARCHAR(50)
perimeter	perimeter	DOUBLE
house_shape	house_shape	VARCHAR(50)
area	area	DOUBLE
measurement_unit (FK)	measurement_unit	VARCHAR(25)
land_owner	land_owner	VARCHAR(100)
uka_propertyno	uka_propertyno	VARCHAR(2147483647)
comments	comments	VARCHAR(2147483647)
gtype	gtype	VARCHAR(10)
current_workflow_status_id	current_workflow_status_id	BIGINT
workflow_status_update_time	workflow_status_update_time	TIMESTAMP
userid	userid	INTEGER
survey_date	survey_date	TIMESTAMP
imei_number	imei_number	VARCHAR(2147483647)
the_geom	the_geom	geometry
address1	address1	VARCHAR(2147483647)
address2	address2	VARCHAR(2147483647)
postal_code	postal_code	VARCHAR(10)
existing_use (FK)	existing_use	INTEGER
proposed_use (FK)	proposed_use	INTEGER
neighbor_north	neighbor_north	VARCHAR(200)
neighbor_south	neighbor_south	VARCHAR(200)
neighbor_east	neighbor_east	VARCHAR(200)
neighbor_west	neighbor_west	VARCHAR(200)
witness_1	witness_1	VARCHAR(200)
witness_2	witness_2	VARCHAR(200)
witness_3	witness_3	VARCHAR(200)
witness_4	witness_4	VARCHAR(200)
quality_of_soil	quality_of_soil	INTEGER
slope	slope	INTEGER
line	line	geometry
point	point	geometry
polygon	polygon	geometry
usin_str	usin_str	VARCHAR(20)
active	active	BOOLEAN
hamlet_id	hamlet_id	INTEGER

References

- land_use_type through (existing_use)

- land_use_type through (proposed_use)
- project through (project_name)
- unit through (measurement_unit)
- project_hamlets through ()

Referenced By

- adjacent_property referencing (usin)
- social_tenure_relationship referencing (usin)
- source_document referencing (usin)
- spatialunit_personadministrator referencing (usin)
- sunit_workflow_status_history referencing (usin)
- spatialunit_deceased_person referencing (usin)
- spatialunit_personwithinterest referencing (usin)

5.1.1.60 spatialunit_deceased_person

(Physical Name: spatialunit_deceased_person)

Logical Column Name	Physical Column Name	Type
id (PK)	id	BIGINT
firstname	firstname	VARCHAR(2147483647)
middlename	middlename	VARCHAR(2147483647)
lastname	lastname	VARCHAR(2147483647)
usin (FK)	usin	BIGINT

References

- spatial_unit through (usin)

5.1.1.61 spatialunit_personadministrator

(Physical Name: spatialunit_personadministrator)

Logical Column Name	Physical Column Name	Type
gid (PK)	gid	BIGINT
usin (FK)	usin	BIGINT
adminid (FK)	adminid	BIGINT

References

- person_administrator through (adminid)
- spatial_unit through (usin)

5.1.1.62 spatialunit_personwithinterest
(Physical Name: spatialunit_personwithinterest)

Logical Column Name	Physical Column Name	Type
id (PK)	id	BIGINT
usin (FK)	usin	BIGINT
person_name	person_name	VARCHAR(200)

References

- spatial_unit through (usin)

5.1.1.63 style
(Physical Name: style)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(255)
filename	filename	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(50)

5.1.1.64 sunit_status
(Physical Name: sunit_status)

1. New
2. Adjudicated
3. Spatially Validated
4. Approved
5. Rejected
6. CCRO Generated

Logical Column Name	Physical Column Name	Type
workflow_status_id (PK)	workflow_status_id	INTEGER
workflow_status	workflow_status	VARCHAR(2147483647)

Referenced By

- sunit_workflow_status_history referencing (workflow_status_id)

5.1.1.65 sunit_workflow_status_history
(Physical Name: sunit_workflow_status_history)

Logical Column Name	Physical Column Name	Type
status_hist_id (PK)	status_hist_id	BIGINT
usin (FK)	usin	BIGINT
workflow_status_id (FK)	workflow_status_id	INTEGER
userid	userid	BIGINT
status_change_time	status_change_time	DATE

References

- spatial_unit through (usin)
- sunit_status through (workflow_status_id)

5.1.1.66 surveyprojectattributes
(Physical Name: surveyprojectattributes)

stores the attribute and survey project mapping

Logical Column Name	Physical Column Name	Type
uid	uid	BIGINT
name (FK)	name	VARCHAR(100)
id (FK)	id	INTEGER
attributecategoryid	attributecategoryid	INTEGER
attributeorder	attributeorder	INTEGER

References

- attribute_master through (id)
- project through (name)

5.1.1.67 task
(Physical Name: task)

Logical Column Name	Physical Column Name	Type
tasktypeid (PK)	tasktypeid	INTEGER
task	task	VARCHAR(50)

Logical Column Name	Physical Column Name	Type
survey_type	survey_type	VARCHAR(50)

Referenced By

- task_scheduler referencing (tasktypeid)

5.1.1.68 task_scheduler

(Physical Name: task_scheduler)

Logical Column Name	Physical Column Name	Type
taskid (PK)	taskid	INTEGER
priority	priority	SMALLINT
task_prompt	task_prompt	INTEGER
target_days	target_days	INTEGER
tasktype (FK)	tasktype	INTEGER

References

- task through (tasktype)

5.1.1.69 tenure_class

(Physical Name: tenure_class)

Logical Column Name	Physical Column Name	Type
tenureclass_id (PK)	tenureclass_id	INTEGER
tenure_class	tenure_class	VARCHAR(2147483647)
active	active	BOOLEAN

Referenced By

- social_tenure_relationship referencing (tenureclass_id)

5.1.1.70 unit

(Physical Name: unit)

Logical Column Name	Physical Column Name	Type
name (PK)	name	VARCHAR(25)

Logical Column Name	Physical Column Name	Type
description	description	VARCHAR(255)
id	id	INTEGER
tenantid	tenantid	VARCHAR(25)

Referenced By

- layer referencing (name)
- project referencing (name)
- spatial_unit referencing (name)

5.1.1.71 user_project

(Physical Name: user_project)

Logical Column Name	Physical Column Name	Type
id	id	INTEGER
project (FK)	project	VARCHAR(100)
tenantid	tenantid	VARCHAR(25)
userid (FK) (FK)	userid	INTEGER

References

- project through (project)
- users through (userid)
- users through (userid)

5.1.1.72 user_role

(Physical Name: user_role)

Logical Column Name	Physical Column Name	Type
id	id	INTEGER
role (FK)	role	VARCHAR(25)
tenantid	tenantid	VARCHAR(25)
userid (FK)	userid	INTEGER

References

- role through (role)
- users through (userid)

5.1.1.73 users
(Physical Name: users)

Logical Column Name	**Physical Column Name**	**Type**
id (PK)	id	INTEGER
name	name	VARCHAR(75)
defaultproject	defaultproject	VARCHAR(25)
email	email	VARCHAR(75)
passwordexpires	passwordexpires	DATE
lastactivitydate	lastactivitydate	DATE
tenantid	tenantid	VARCHAR(25)
active	active	BOOLEAN
password	password	VARCHAR(70)
authkey	authkey	VARCHAR(255)
phone	phone	VARCHAR(12)
manager_name	manager_name	VARCHAR(25)
username	username	VARCHAR(75)

Referenced By

- user_project referencing (id)
- user_project referencing (id)
- user_role referencing (id)

5.1.1.74 vertexlabel
(Physical Name: vertexlabel)

Logical Column Name	**Physical Column Name**	**Type**
gid (PK)	gid	INTEGER
the_geom	the_geom	geometry

5.2 Mobile Data Capture Application (SQLLite)

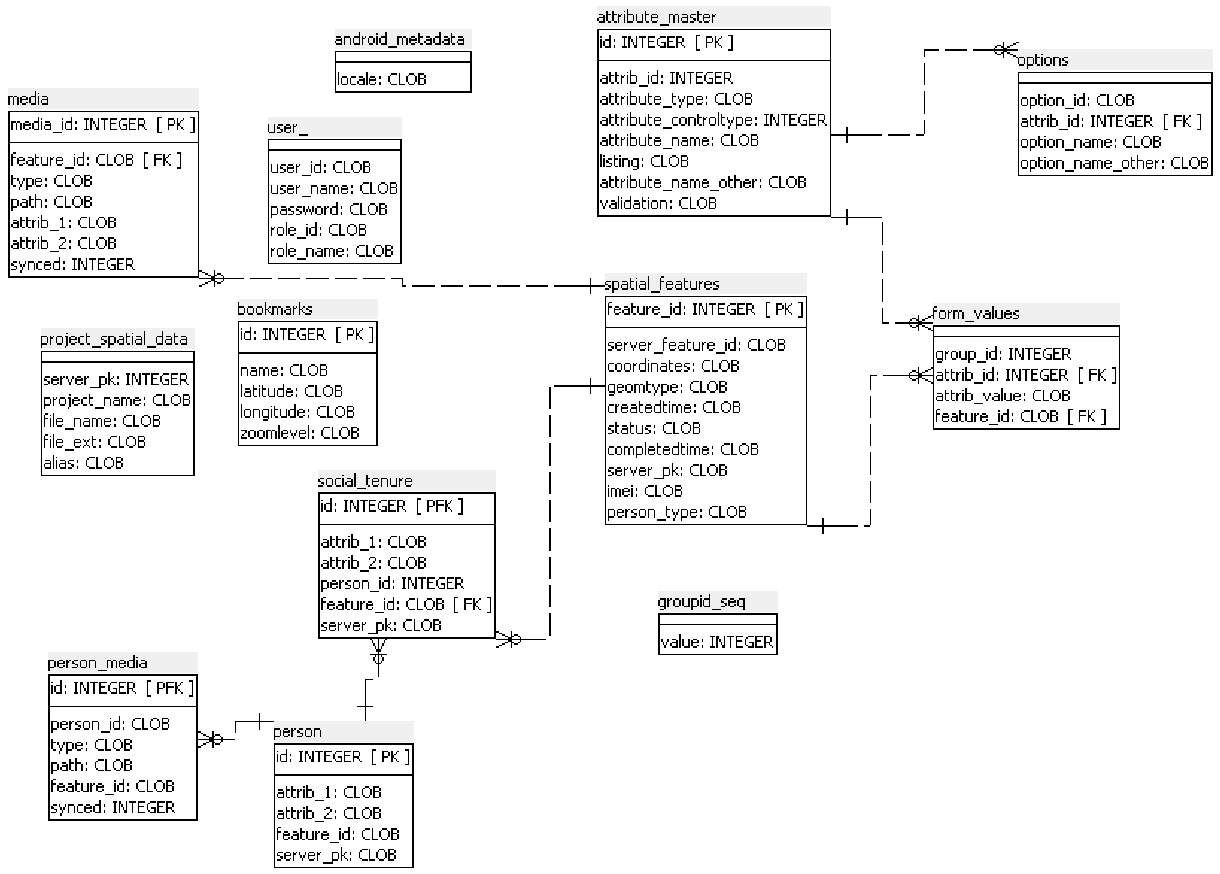


Figure 13: Mobile sqlLite ER diagram

5.2.1 Table Details

5.2.1.1 android_metadata

(Physical Name: android_metadata)

Logical Column Name	Physical Column Name	Type
Locale	Locale	Text

5.2.1.2 attribute_master

(Physical Name: attribute_master)

Logical Column Name	Physical Column Name	Type
id (PK)	id	Integer
attrib_id	attrib_id	Integer
attribute_type	attribute_type	Text
attribute_controltype	attribute_controltype	Integer

Logical Column Name	Physical Column Name	Type
attribute_name	attribute_name	Text
Listing	listing	Text
attribute_name_other	attribute_name_other	Text
Validation	validation	Text

Referenced By

form_values referencing (id)

options referencing (id)

5.2.1.3 bookmarks

(Physical Name: bookmarks)

Logical Column Name	Physical Column Name	Type
id (PK)	id	Integer
Name	name	Text
Latitude	latitude	Text
Longitude	longitude	Text
Zoomlevel	zoomlevel	Text

5.2.1.4 form_values

(Physical Name: form_values)

Logical Column Name	Physical Column Name	Type
group_id	group_id	Integer
attrib_id (FK)	attrib_id	Integer
attrib_value	attrib_value	Text
feature_id (FK)	feature_id	Text

References

attribute_master through (attrib_id)

spatial_features through (feature_id)

5.2.1.5 groupid_seq

(Physical Name: groupid_seq)

Logical Column Name	Physical Column Name	Type
Value	value	Integer

5.2.1.6 *media*
(Physical Name: media)

Logical Column Name	Physical Column Name	Type
media_id (PK)	media_id	Integer
feature_id (FK)	feature_id	Text
Type	type	Text
Path	path	Text
attrib_1	attrib_1	Text
attrib_2	attrib_2	Text
Synced	synced	Integer

References

spatial_features through (feature_id)

5.2.1.7 *options*
(Physical Name: options)

Logical Column Name	Physical Column Name	Type
option_id	option_id	Text
attrib_id (FK)	attrib_id	Integer
option_name	option_name	Text
option_name_other	option_name_other	Text

References

attribute_master through (attrib_id)

5.2.1.8 *person*
(Physical Name: person)

Logical Column Name	Physical Column Name	Type
id (PK)	id	Integer
attrib_1	attrib_1	Text
attrib_2	attrib_2	Text
feature_id	feature_id	Text
server_pk	server_pk	Text
person_subtype	person_subtype	Text

Referenced By

person_media referencing (id)

social_tenure referencing (id)

5.2.1.9 person_media

(Physical Name: person_media)

Logical Column Name	Physical Column Name	Type
id (PK) (FK)	id	Integer
person_id	person_id	Text
Type	type	Text
Path	path	Text
feature_id	feature_id	Text
Synced	synced	Integer

References

person through (id)

5.2.1.10 project_spatial_data

(Physical Name: project_spatial_data)

Logical Column Name	Physical Column Name	Type
server_pk	server_pk	Integer
project_name	project_name	Text
file_name	file_name	Text
file_ext	file_ext	Text
Alias	alias	Text

5.2.1.11 social_tenure

(Physical Name: social_tenure)

Logical Column Name	Physical Column Name	Type
id (PK) (FK)	id	Integer
attrib_1	attrib_1	Text
attrib_2	attrib_2	Text
person_id	person_id	Integer
feature_id (FK)	feature_id	Text
server_pk	server_pk	Text
village_name	village_name	Text

References

person through (id)

spatial_features through (feature_id)

5.2.1.12 spatial_features
(Physical Name: spatial_features)

Logical Column Name	Physical Column Name	Type
feature_id (PK)	feature_id	Integer
server_feature_id	server_feature_id	Text
Coordinates	coordinates	Text
Geomtype	geomtype	Text
Createdtime	createdtime	Text
Status	status	Text
Completedtime	completedtime	Text
server_pk	server_pk	Text
Imei	imei	Text
person_type	person_type	Text
hamlet_id	hamlet_id	Integer
witness_1	witness_1	Text
witness_2	witness_1	Text

Referenced By:

social_tenure referencing (feature_id)

media referencing (feature_id)

form_values referencing (feature_id)

next kin details referencing (feature_id)

deceased_person referencing (feature_id)

*5.2.1.13 user_*
(Physical Name: user_)

Logical Column Name	Physical Column Name	Type
user_id	user_id	Text
user_name	user_name	Text
Password	password	Text
role_id	role_id	Text
role_name	role_name	Text

5.2.1.14 hamlet_details
(Physical Name: hamlet_details)

Logical Column Name	Physical Column Name	Type
Id(PK)	Id	Integer
hamlet_name	hamlet_name	Text

Referenced By

spatial_features referencing (hamlet_id)

5.2.1.15 adjudicator_details
(Physical Name: adjudicator_details)

Logical Column Name	Physical Column Name	Type
Id(PK)	Id	Integer
adjudicator _name	adjudicator _name	Text

5.2.1.16 next_kin_details
(Physical Name: next_kin_details)

Logical Column Name	Physical Column Name	Type
Id(PK)	Id	Integer
next_kin _name	next_kin _name	Text
feature_id	feature_id	Text

5.2.1.17 deceased_person
(Physical Name: deceased_person)

Logical Column Name	Physical Column Name	Type
Id(PK)	Id	Integer
first _name	first _name	Text
middle_name	middle _name	Text
last_name	last _name	Text
feature_id	feature_id	Text