Zambia Customary Land Administration System

(ZCLAS)

Deployment Guide

(v. 0.1)

February 2023

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Comments** | **Version** |
| Alexander Solovov | February 10, 2023 | Initial draft | 0.1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 5](#_Toc127825222)

[2. Platforms and Tools 7](#_Toc127825223)

[3. Java 8](#_Toc127825224)

[4. Database 9](#_Toc127825225)

[5. Tomcat 10](#_Toc127825226)

[5.1.1. JDBC driver 10](#_Toc127825227)

[5.1.2. Installing ZCLAS and GeoServer 10](#_Toc127825228)

[5.1.3. Configure database connection 11](#_Toc127825229)

[5.1.4. Configuring GeoServer 12](#_Toc127825230)

[6. Configuring Plots layer in ZCLAS 19](#_Toc127825231)

Table of Figures

[Figure 1 – context.xml file 11](#_Toc127825247)

[Figure 2 – server.xml file 11](#_Toc127825248)

[Figure 3 – GeoServer – New Workspace 12](#_Toc127825249)

[Figure 4 – GeoServer – New Style 15](#_Toc127825250)

[Figure 5 – GeoServer – data source types 16](#_Toc127825251)

[Figure 6 – GeoServer – New Vector Data Source 17](#_Toc127825252)

[Figure 7 – New Layer – Bounding Boxes 18](#_Toc127825253)

[Figure 8 – ZCLAS – Plots layer URL 19](#_Toc127825254)

[Figure 9 – ZCLAS – Map page 20](#_Toc127825255)

# Introduction

Since 2014, USAID ILRG has developed and implemented a systematic customary land documentation process resulting in the registration of ~35,000 parcels across nine chiefdoms in Eastern Province of Zambia. This process has been based on a series of data collection and validation steps in Open Data Kit (ODK), which are loaded in ODK Central and subsequently imported into a bulk certification database in Postgres database through a collection of SQL codes. These chiefdom data were collected through multiple different project cycles with data collected through different systems (and naming conventions). Within this bulk database, ODK forms are imported directly into the database. These are then validated through Microsoft Access (for text data), and parcel boundaries are confirmed or drawn (for spatial data) in QGIS. Objections and corrections maps are produced out of QGIS, and then validated again in ODK, prior to the production of certificates.

The base data of parties, parcel and tenure were (largely) normalized in 2021/2022 and placed into an administration database. Some legacy issues likely remain. ILRG made a decision to leave its partners with a basic platform that can be maintained locally. For this reason, ZCLAS application was implemented, migrating existing records from the administration database and taking over their further maintenance.

ZCLAS was developed to fill in the gap of missing customary land registration tool for using it in various chiefdoms. It was developed as a Web-application, using open source technologies to minimize licensing impact and reduce total cost of ownership (TCO) of the system.

ZCLAS automates customary land rights registration and further transactions, providing a reliable tool to track and control registration process, making it clear and transparent. A number of business rules, developed in the system, help to avoid different errors and keep land right records comprehensive, consistent and up to date. ZCLAS is a multiuser system, providing strictly controlled user access to the system functions according to the role-based security policy.

The following key functions are implemented in ZCLAS:

* Dashboard viewing
* Creating new applications
* Application editing
* Application viewing
* Application searching
* Creating and managing land parcels
* Map viewing
* Registration of new customary land rights
* Transfer of customary land rights
* Rectification of customary land records
* Surrender of customary land rights
* Customary land parcel split (subdivision) and merge (consolidation)
* Customary land records searching
* Application processing
* Application approval and rejection
* Printing customary land certificate
* Printing reports
* Management of users
* Management of groups
* Management of reference data tables
* Management of spatial layers
* Audit trail

Considering that ZCLAS allows customization of different user groups with different access rights to the systems, the following initial groups are defined by default:

* **Senior Managers** – Group of users who can make final approvals or rejections of the applications.
* **GIS Technicians** – Group of users who can do spatial and attribute updates, check and validate prepared changes to the customary land records.
* **Land Clerks** – Group of users who can collect field data, create different applications in the system and prepare changes to the customary land records.
* **Viewers** – Group of users who can only view the data and generate various reports. Data modifications are not allowed.
* **Administrators** – Group of users who can create and manage groups and users, manage system tables and reference data tables.

Although these groups are defined in the system, they can be further edited or new groups created to customize specific access rights if needed.

The purpose of this guide is to focus on the deployment of ZCLAS. Deployment procedures, described in this guide are referencing Ubuntu environment.

# Platforms and Tools

ZCLAS is developed using open-source platforms and tools, which allows running it on different operating systems. The following minimum requirements must be considered:

**Server**

Software

* Windows Server 2008/2012 or Ubuntu/Linux OS
* PostgreSQL Server 12.X with PostGIS extension
* Apache Tomcat 9.0.X
* Java OpenJDK 1.8.0\_352 (Java 8)
* GeoServer 2.9.X

Hardware

* Processor Intel Core i5 2.8 Ghz
* Memory DDR3 8 GB
* HDD SATA 500 GB
* Integrated Graphics card
* Gigabit Ethernet network controller

**Client**

Software

* Windows 7/8/10 or Ubuntu
* Mozilla Firefox

Hardware

* Processor Intel Core i3 2.5 Ghz
* Memory DDR3 2 GB
* Display 15” with 1024x768 resolution
* HDD ATA 300 GB
* Integrated Graphics card
* 10/100 Ethernet network controller or WiFi

# Java

The basic requirement is to have Java environment, which is used for execution of the application. The recommended version of Java for ZCLAS is Java EE 8. Using Java 9 and higher is not recommended as they have some missing libraries.

OpenJDK is recommended as the Oracle Java is not free anymore. Follow one of many guides on the Internet to install Java 8 (e.g. <https://docs.datastax.com/en/jdk-install/doc/jdk-install/installOpenJdkDeb.html>).

# Database

ZCLAS is using PostgreSQL for hosting its database. Recommended version is 12.X. Follow one of the guides to complete the installation (e.g. <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-18-04>).

Once PostgreSQL is installed and you can connect it from pgAdmin or command line, you will need to create ZCLAS database. Complete the following steps:

1. Open **pgAdmin** and connect to you PostgreSQL.
2. Right click on the **Databases** under server name and select **New Database**.
3. Enter “**zclas**” as a database name and set UTF-8 as database encoding on the **Definition** tab.
4. Click **OK** to create new database.
5. Select trust database in the list of databases and click **Query tool** on the toolbar or open it from main menu **Tools → Query tool**.
6. In the Query window, open **database\_creation.sql** script from **DB** folder of the installation folder.
7. Click **Execute query** button on the toolbar or simply press **F5** or select main menu **Query → Execute**.

There should be no errors after executing the script and your ZCLAS database ready for work. Alternatively, you can use **psql** command line tool to execute **database\_creation.sql** script:

psql -U *username* --password -d zclas -a -f /*path\_to\_file*/database\_creation.sql

# Tomcat

ZCLAS is running on Apache Tomcat Web-Server and recommended version is 9.X. Install Tomcat 9 following one of the guides (e.g. <https://www.howtoforge.com/tutorial/how-to-install-and-configure-apache-tomcat-on-ubuntu-20-04/>). Skip Java installation steps, since it is already installed.

If multiple Java versions are run on the server, make sure that Tomcat is referencing Java 8, installed in Chapter 3 of this guide. For an explicit referencing of Java 8, adjust or create **setenv.sh** file, under the Tomcat root folder (e.g. **/var/lib/tomcat9**). Copy and paste the following lines:

JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

JRE\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

You do not need to do it if only one version of Java was installed on your server.

Upon successful installation and running the Tomcat, it should be available by server’s IP address and port number 8080 (e.g. [http://*server\_ip*:8080](http://server_ip:8080)). Default port number can be changed in the **server.xml** configuration file under Tomcat **conf** folder (e.g. **/var/lib/tomcat9/conf**). Search for the following lines and make adjustments is required:

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

### JDBC driver

In order to be able to access PostgreSQL from ZCLAS application you need to put PostgreSQL JDBC driver into Tomcat folder. Download JDBC driver from <https://jdbc.postgresql.org/download/postgresql-9.4-1202.jdbc41.jar> and put it into Tomcat folder under **lib** (e.g. **/usr/share/tomcat9/lib**). Make sure to restart Tomcat to take effect.

### Installing ZCLAS and GeoServer

Next, let us install GeoServer and ZCLAS applications. The installation procedure is simple, requiring you to copy their WAR files into Tomcat folder.

First, download and extract **geoserver.war** from <https://sourceforge.net/projects/geoserver/files/GeoServer/2.9.1/geoserver-2.9.1-war.zip/download>.

Copy geoserver.war into Tomcat **webapps** folder (e.g. **/var/lib/tomcat9/webapps**). Next, copy **zclas.war** file from **ZCLAS** folder into the same **webapps** folder.

### Configure database connection

Open **context.xml** file under the **conf** folder (e.g. **/var/lib/tomcat9/conf**) and add the following line after **<Context>** tag:

<ResourceLink global="jdbc/zclas" name="jdbc/zclas" type="javax.sql.DataSource"/>

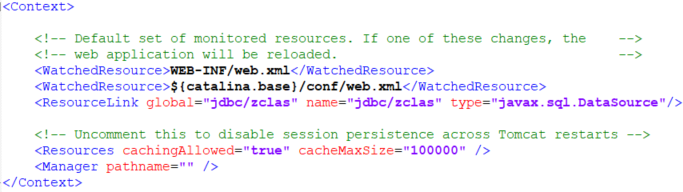


Figure 1 – context.xml file

Save and close the file. Next, open **server.xml** from the same **conf** folder and find **<GlobalNamingResources>** tag. Copy and insert the following text after this tag:

<Resource name="jdbc/trust"

**url="jdbc:postgresql://localhost:5432/zclas"**

auth="Container"

type="javax.sql.DataSource"

factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"

driverClassName="org.postgresql.Driver"

**username="your\_user\_name"**

**password="db\_password\_here"**

maxActive="100"

maxIdle="10"

maxWait="-1"

jdbcInterceptors="org.apache.tomcat.jdbc.pool.interceptor.ResetAbandonedTimer" />

Make sure to adjust highlighted values, as per your database configuration.



Figure 2 – server.xml file

Finally start (or re-start if running) Tomcat service. Even though Tomcat service will start fast, you may need to wait 2-3 minutes for all applications to be loaded by Tomcat.

Open Web-browser and enter [http://*server\_ip*:8080/zclas](http://server_ip:8080/zclas) to test ZCLAS application. If you have changed port number to 80, your URL will be [http://*server\_ip*/zclas](http://server_ip/zclas). Make sure you can see ZCLAS login screen.

### Configuring GeoServer

In order to show land plots layer in ZCLAS, it has to be published on GeoServer first. Open Web-browser and go to GeoServer application at [http://*server\_ip*:9090/geoserver](http://server_ip:9090/geoserver) or [http://*server\_ip*/geoserver](http://server_ip/geoserver) if you changed to default port 80. Once GeoServer is opened, enter default user name **admin** with password **geoserver**.

First, we have to create a workspace. Click on **Workspaces** in the left menu and then click **Add new workspace**. Enter **zclas** for **Name** and **Namespace URI**. Make it as default.

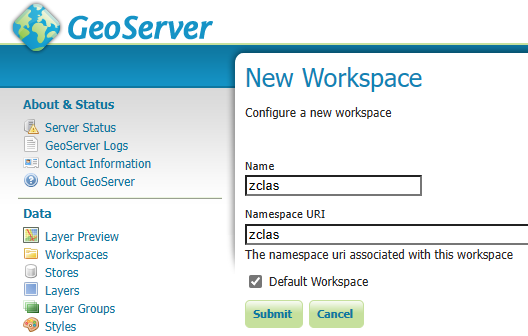


Figure 3 – GeoServer – New Workspace

Click **Submit** button to save new workspace.

Next step is to create layer style for land plots. Click on the **Styles** in the left menu and then click **Add new style** link.

Enter **Plots** into **Name** field, select **zclas** as a workspace. Copy and paste the following text into the text field:

<?xml version="1.0" encoding="ISO-8859-1"?>

<StyledLayerDescriptor version="1.0.0"

xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"

xmlns="http://www.opengis.net/sld"

xmlns:ogc="http://www.opengis.net/ogc"

xmlns:xlink="http://www.w3.org/1999/xlink"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<!-- a Named Layer is the basic building block of an SLD document -->

<NamedLayer>

<Name>plot\_polygon</Name>

<UserStyle>

<!-- Styles can have names, titles and abstracts -->

<Title>Plot Polygon</Title>

<Abstract>Plot style that draws a polygon</Abstract>

<FeatureTypeStyle>

<Rule>

<Name>rule1</Name>

<Title>Pending</Title>

<PolygonSymbolizer>

<Fill>

<CssParameter name="fill">#FFFFFF</CssParameter>

<CssParameter name="fill-opacity">0</CssParameter>

</Fill>

<Stroke>

<CssParameter name="stroke">#FFE900</CssParameter>

<CssParameter name="stroke-width">2</CssParameter>

</Stroke>

</PolygonSymbolizer>

<TextSymbolizer>

<Geometry>

<ogc:Function name="centroid">

<ogc:PropertyName>geom</ogc:PropertyName>

</ogc:Function>

</Geometry>

<Label>

<ogc:PropertyName>upi</ogc:PropertyName>

</Label>

<LabelPlacement>

<PointPlacement>

<AnchorPoint>

<AnchorPointX>0.5</AnchorPointX>

<AnchorPointY>0.5</AnchorPointY>

</AnchorPoint>

</PointPlacement>

</LabelPlacement>

<Halo>

<Radius>2</Radius>

<Fill>

<CssParameter name="fill">#FFFFFF</CssParameter>

</Fill>

</Halo>

<VendorOption name="conflictResolution">true</VendorOption>

<VendorOption name="goodnessOfFit">0</VendorOption>

<VendorOption name="autoWrap">60</VendorOption>

</TextSymbolizer>

</Rule>

<Rule>

<Name>rule2</Name>

<Title>Active</Title>

<ogc:Filter>

<ogc:PropertyIsEqualTo>

<ogc:PropertyName>status\_code</ogc:PropertyName>

<ogc:Literal>active</ogc:Literal>

</ogc:PropertyIsEqualTo>

</ogc:Filter>

<PolygonSymbolizer>

<Fill>

<CssParameter name="fill">#FFFFFF</CssParameter>

<CssParameter name="fill-opacity">0</CssParameter>

</Fill>

<Stroke>

<CssParameter name="stroke">#4EAD00</CssParameter>

<CssParameter name="stroke-width">2</CssParameter>

</Stroke>

</PolygonSymbolizer>

</Rule>

<Rule>

<Name>rule3</Name>

<Title>Historic</Title>

<ogc:Filter>

<ogc:PropertyIsEqualTo>

<ogc:PropertyName>status\_code</ogc:PropertyName>

<ogc:Literal>historic</ogc:Literal>

</ogc:PropertyIsEqualTo>

</ogc:Filter>

<PolygonSymbolizer>

<Fill>

<CssParameter name="fill">#FFFFFF</CssParameter>

<CssParameter name="fill-opacity">0</CssParameter>

</Fill>

<Stroke>

<CssParameter name="stroke">#AAAAAA</CssParameter>

<CssParameter name="stroke-width">2</CssParameter>

</Stroke>

</PolygonSymbolizer>

</Rule>

</FeatureTypeStyle>

</UserStyle>

</NamedLayer>

</StyledLayerDescriptor>

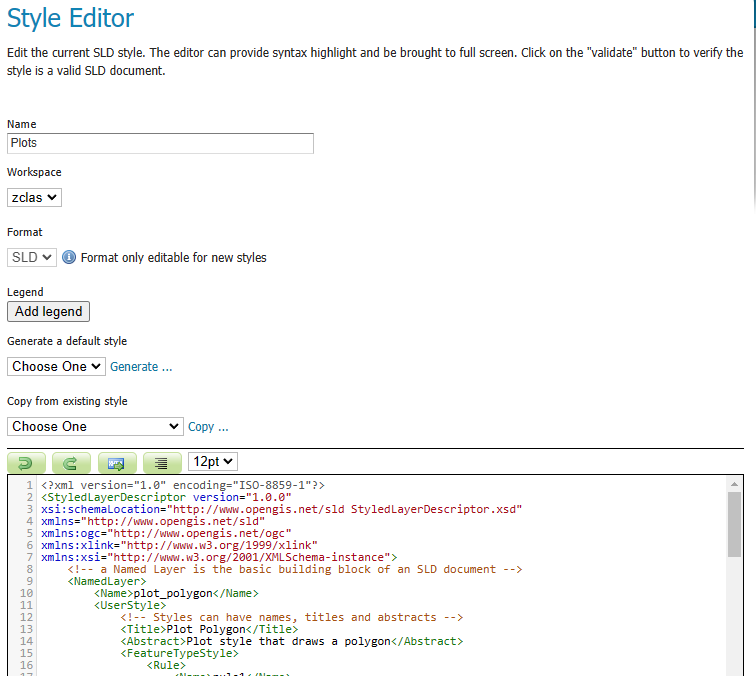


Figure 4 – GeoServer – New Style

Click **Submit** button at the end to save style.

Following the style creation, we should create now a data source (store) and publish parcels layer. Click on the **Stores** in the left menu and then click **Add new Store** link. On the **New data source** page, click **PostGIS** link.

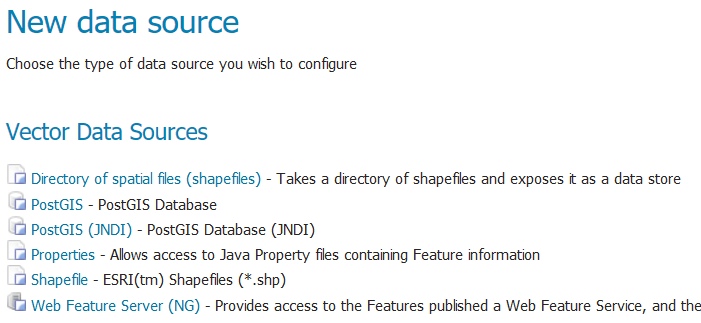


Figure 5 – GeoServer – data source types

On the next screen of New Vector Data Source fill in the following values:

**Workspace** = zclas

**Data Source Name** = zclas\_db

**Host** = localhost

**Port** = 5432

**Database** = zclas

**User** = your user name for accessing database

**Password** = your password for accessing database

**Expose primary keys** = checked

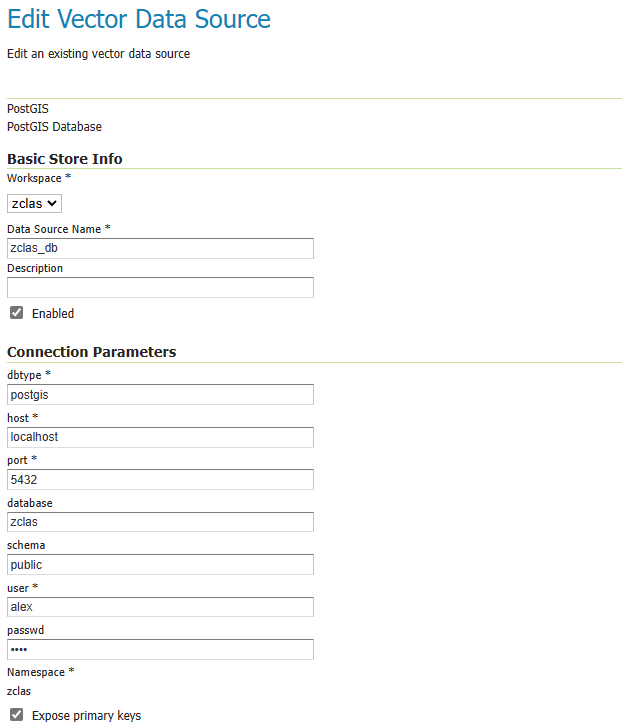


Figure 6 – GeoServer – New Vector Data Source

Leave other parameters as they are and click **Save** button at the end. If you provided correct parameters, you will get on the **New Layer** page where you can see all tables from the database.

Find **parcel** table and click **Publish** link in the **Action** column. You will get on the **Edit Layer** page. Enter **parcel** into the Name field, **Plots** into the **Title** filed and scroll down Bounding Boxes section. Under the empty fields, click **Compute from data** link and then **Compute from native bounds**. If you have any data in the database it will calculate bounding boxes based on it, otherwise default values will be used.

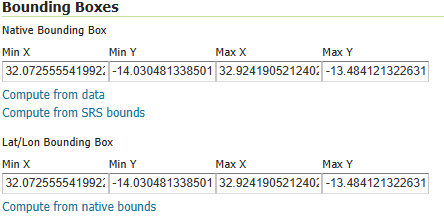


Figure 7 – New Layer – Bounding Boxes

Scroll to the top of the page and click **Publishing** tab. Scroll down to the Layer Settings and select **zclas:Plots** style in the **Default Style** dropdown box.

Finally click **Save** button at the end of the page. Now we have finished with GeoServer configuration and we can add/adjust Plots layer in the ZCLAS application. Check chapter 6 for more details.

# Configuring Plots layer in ZCLAS

ZCLAS database is configured by default with Plots layer, referencing local GeoServer. This setting may work from local computer if you set default port number of Tomcat to 80, but in any case, it will fail when accessing ZCLAS from the network. In order to fix it you need to know server name or IP address where ZCLAS is installed.

Depending on your settings, open Web-browser and go to [http://*server\_ip*:8080/zclas](http://server_ip:8080/zclas) or http://*server\_ip*/zclas or any other port number, which you configured for Tomcat. Login into ZCLAS using default user name and password. Once you have logged in, go to **Administration → Map Layers**.

Click edit icon (pencil) in front of the layer name. On the popup screen, change URL address to replace localhost with IP address or name of the server, running ZCLAS. If you are using port numbers (8080 or 9090), add them after server IP/name as well.

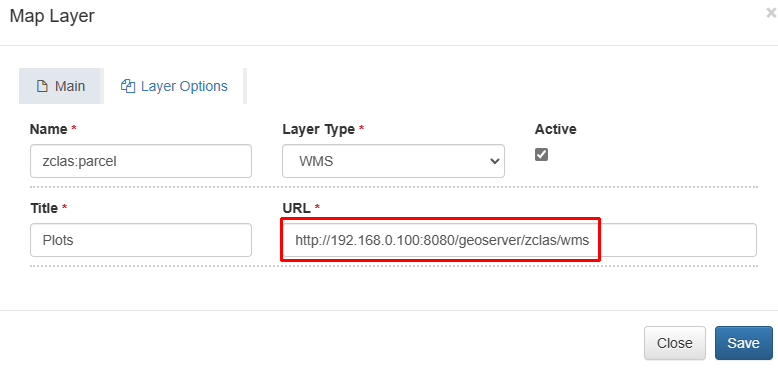


Figure 8 – ZCLAS – Plots layer URL

If by any chance you have different layer name, you can adjust it in the **Name** field. Finally click **Save** button to apply changes. If all settings are matching with published Plots layer on GeoServer, ZCLAS users will be able to see land plots on the map (once those plots are created or migrated).

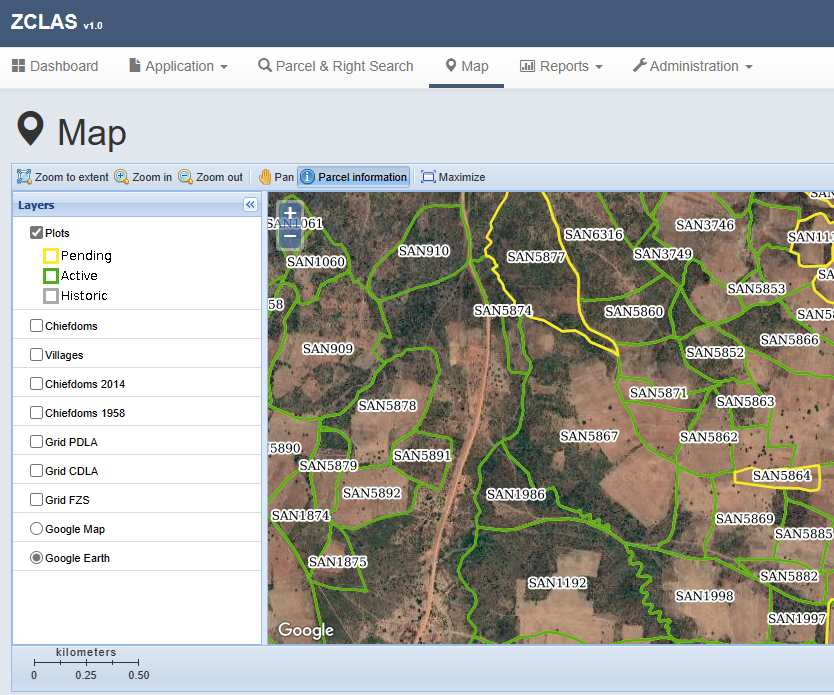


Figure 9 – ZCLAS – Map page