USAID
FROM THE AMERICAN PEOPLE

LAND TECHNOLOGY SOLUTIONS (LTS) PROJECT

MAST Deployment Guide (v 2.2)

March 2019

This document was produced for review by the United States Agency for International Development. It was prepared by SSG-Advisors under the Land Technology Solutions (LTS) contract.

Prepared for the United States Agency for International Development, USAID/E3/LU Contract Number AID-OAA-C-17-00056, Land Technology Solutions (LTS) contract.

Prepared by:

SSG-Advisors, LLC

1 Mill Street, Suite 201	2000 P Street, Suite 401
Burlington, VT 05401	Washington, DC 20036
Phone: (802) 735-1169	Phone: (202) 548-7107
Fax: (866) 595-9620	

CONTENTS

ACRONYMS & ABBREVIATIONS

LTS	Land Technology Solutions
MAST	Mobile Applications to Secure Tenure
ME&L	Monitoring, Evaluation and Learning
ERC	USAID Evaluation, Research and Communication (ERC) project
USAID	United States Agency for International Development
USAID/E3/LU	USAID's Bureau for Economic Growth, Education and Environment, Office of Land and Urban

1. INTRODUCTION

This document is a guide to setting up a MAST server and mobile application instance from scratch. It is designed to be used in conjunction with the MAST Implementation Guide, and serves as a technical manual for all the technology components that underlie the MAST platform. The intended audience is software developers, systems administrators, and other technical specialists.

It is assumed that the user has reasonable familiarity with Linux servers, Java applications, databases, and Android applications. Working knowledge of git and other tools will also be necessary. This guide makes extensive use of the Linux command line, and assumes that the user has a Linux development environment available for use, though a Windows or OSX computer can certainly be used with some modifications.

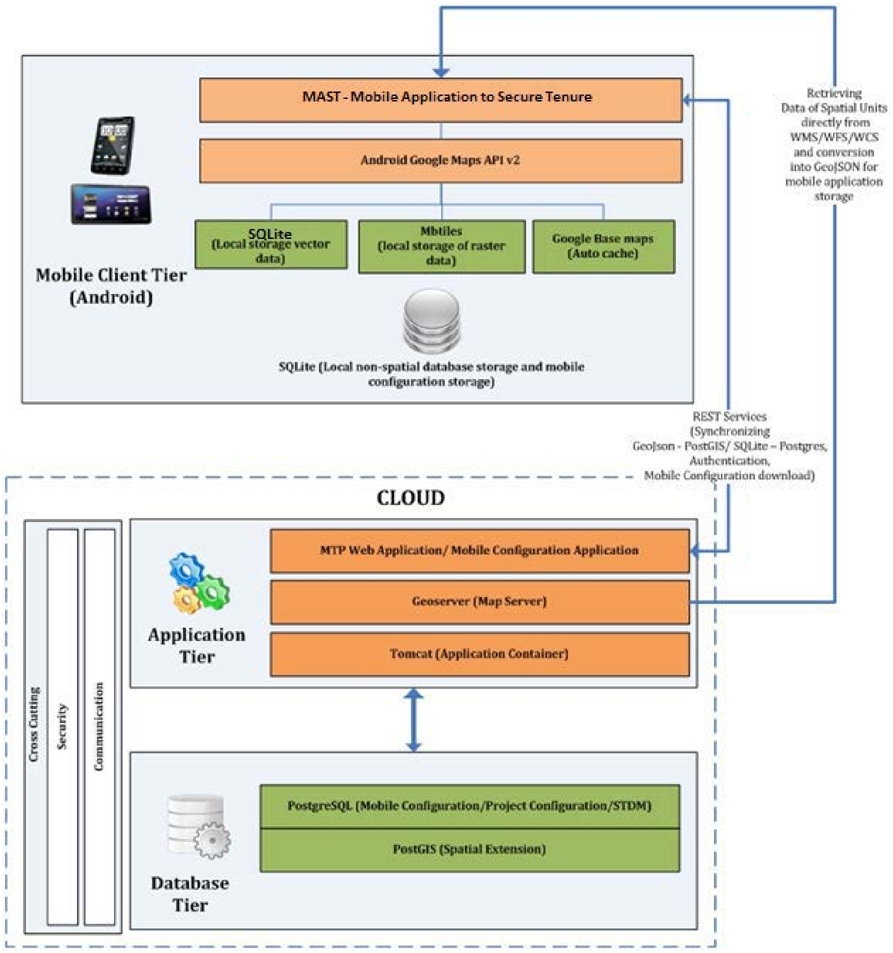

MAST - Mobile Application to Secure Tenure
Android Google Maps API v2
SQLite (Local storage vector data)
Mbtiles (local storage of raster data)
Google Base maps (Auto cache)
Mobile Client Tier (Android)
SQLite (Local non-spatial database storage and mobile configuration storage)
Retrieving Data of Spatial 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 1 – MAST infrastructure

A complete deployment of MAST requires, at a minimum, a server and enough Android devices to meet the project's needs. The server can be a physical server or a cloud server, and this guide provides information for both use cases. Android devices may be mobile phones or tablets, but should run Android

4+ and have an onboard GPS unit with sufficient accuracy to map land parcels. For improved accuracy, it is recommended that you use an external GPS unit connected to the mobile device via Bluetooth or USB.

2. MAST Data Management Infrastructure

2.1 HOSTING

Before beginning MAST deployment, first consider what type of server hosting you will be using. MAST can either be run on a cloud server, like Amazon Web Services (AWS), or on a local, physical server. Either option is viable, and the decision rests on the purpose, environment, and longevity of the project.

A cloud server is appropriate for a MAST instance that is large in scale (regional or national) and that requires consistent server uptime and availability, but does not have a physical environment that would allow these needs to be fulfilled by a local server, i.e. lacks reliable datacenters, electricity, or connectivity. Cloud servers ensure infrastructure reliability for a reasonable monthly fee, but do require an internet connection to reach. Costs tend to be reasonable, but the ability to pay them does require a consistent cash flow. Although cloud servers don't require infrastructure maintenance, they are still servers and require system administration knowledge to set up and maintain.

A local server is appropriate for two cases: when MAST is only going to be utilized in a single office or location, or if there are high quality data centers available locally. In the first case, MAST can be installed on a single computer (even a laptop) and accessed only via a local network. Advantages are the low cost and low complexity, at the expense of flexibility and reach. It also requires a qualified local technician, or a preconfigured machine sent to the location. Configuration will be somewhat different than the cloud server. In the second case, configuration will be identical to that of a cloud server, but will require a database and significant infrastructure support, in addition to overall systems support. This type of solution is only likely to be viable in instances where data sovereignty is paramount and funding is sufficient to pay for high levels of support. Cost may be high.

Amazon Web Services

This guide largely discusses hosting in terms of AWS, though any other hosting service could be used. Simply find a similar server configuration to those suggested below.

MAST can run on relatively small servers, as small as t2.large. However, for large deployments, m4.large and above are recommended. While day to day operations don't require significant resources, serving large amounts of imagery and executing topology checks can be CPU-intensive. It is also recommended that the MAST database run on a separate Amazon Relational Database Service (RDS) server. This makes database management and backup very easy. Since storage is relatively cheap on AWS, elect for 100 GB minimum on each server. SSD is preferable for performance reasons. If you wish to use a smaller server size, such as a t2-tier server, regular application function will likely be sufficient, but topology check runs may be significantly hampered and cause performance issues.

For a local server, select hardware with a similar profile to the AWS server above. A minimum of 4 GB of RAM, 200 GB of storage, and a modern, multicore processor are recommended. If there will be a very small number of users with minimal concurrency, most consumer laptops will run MAST, though performance may not be optimal.

2.2 SERVER SETUP AND PREREQUISITES

AWS Configuration

This guide assumes that the server is running Ubuntu 16.04. This is a long-term support version that is supported with free images from Amazon. Any other Linux distribution will function just as well; simply use your package manager of choice and substitute directory paths with the correct ones.

When configuring your AWS server, make sure that these ports are open via security settings.

- 22 (SSH): limit this to your own IP
- 5432 (PostgreSQL): for the database server if you wish to directly connect a database management client; limit this to your own IP
- 80 (HTTP): for the application server; open this to all IPs that should have access to MAST, or leave open to all
- 443 (HTTPS): for the application server, if using SSL

At the time of this writing, MAST does not support SSL out of the box. Support for this will be added soon.

OS and package installation

Before proceeding, update your apt packages and lists:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Install Java JDK 8 using the webupd8 installer:

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

Install PostgreSQL and PostGIS extensions, if you are using a single server for application and database. If you are using Amazon RDS or equivalent, you will not need to do this, but will need to configure your RDS database to work with PostGIS.

```
$ sudo apt install postgresql postgresql-contrib postgis
```

There are no apt packages for Tomcat, so first browse to the Apache Tomcat 8 download page at https://tomcat.apache.org/download-80.cgi and copy the link for the tar.gz archive link. At the time of this writing, it is https://www-eu.apache.org/dist/tomcat/tomcat-8/v8.5.38/bin/apache-tomcat-8.5.38.tar.gz. Then download the package and unzip it to the **/opt** folder:

```
$ wget https://www-eu.apache.org/dist/tomcat/tomcat-8/v8.5.38/bin/apache-
tomcat-8.5.38.tar.gz
$ sudo tar xzvf apache-tomcat-8.5.38.tar.gz -C /opt
```

Files will be unzipped into **apache-tomcat-8.5.38** folder. Rename it to **tomcat**.

```
$ sudo mv /opt/apache-tomcat-8.5.38/ /opt/tomcat
```

Download and install GeoServer for serving map layers. GeoServer will be deployed to the Tomcat server and therefore we need WAR package. For this guide, the latest 2.15.0 is used.

```
$ wget
'http://sourceforge.net/projects/geoserver/files/GeoServer/2.15.0/geoserver
-2.15.0-war.zip/download'
```

Rename downloaded file into **geoserver.zip** and unzip it to **geoserver** folder.

```
$ mv download geoserver.zip
$ unzip geoserver.zip -d geoserver
```

Now **geoserver** folder contains different files and folders. We need to copy geoserver.war into out Tomcat folder under Web applications subfolder - **/opt/tomcat/webapps**.

```
$ sudo cp geoserver/geoserver.war /opt/tomcat/webapps
```

Once Tomcat server is started geoserwer.war will be expanded into **geoserver** folder and deployed as Web application.

In order to connect to the MAST database, download appropriate JDBC drivers for PostgreSQL and PostGIS and put them into **/opt/tomcat/lib** folder. The following lines use the latest versions as of this writing but you may check here for postgres and here for postgis driver updates. Considering we installed Java 8 or higher, version 4.2 has to be used.

```
$ sudo wget https://jdbc.postgresql.org/download/postgresql-42.2.5.jar -P
/opt/tomcat/lib/
$ sudo wget http://central.maven.org/maven2/net/postgis/postgis-
jdbc/2.3.0/postgis-jdbc-2.3.0.jar -P /opt/tomcat/lib/
```

Configuration

Configure PostgreSQL password

First, run psql utility under postgres account and change and then change password. If you used Amazon RDS, you've already done this during the setup process and will not need to repeat the process now.

```
$ sudo -u postgres psql
postgres=# \password postgres
```

Remember this password, as it will be used to access the database in the future.

Configure Tomcat

First, open your **.bashrc** file for editing using your favorite editor.

```
$ vim ~/.bashrc
```

And add the following lines (considering Java 8 is installed):

```
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CATALINA_HOME=/opt/tomcat
```

Create or edit **setenv.sh** file in the Tomcat folder under **/opt/tomcat/bin** with the editor of your choice and set the following content:

```
CATALINA_OPTS="$CATALINA_OPTS -Dfile.encoding=UTF8 -Duser.timezone=UTC -
server -d64 -XX:NewSize=700m -XX:MaxNewSize=700m -Xms1024m -Xmx2048m -
XX:MaxPermSize=1024m -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -
XX:+CMSParallelRemarkEnabled -XX:SurvivorRatio=20 -XX:ParallelGCThreads=8"
```

This line sets different options for your Tomcat. One of the important ones are **-Xms** (sets minimum allocated memory) and **-Xmx** (sets maximum allocated memory). Make sure that you have enough RAM for configured values.

Make sure that the file is executable:

```
sudo chmod ug+x /opt/tomcat/bin/setenv.sh
```

Add the following resource reference to **/opt/tomcat/conf/context.xml** inside the **Context** tags, i.e. before </Context>:

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

Add the following lines, configuring MAST database connection, to **/opt/tomcat/conf/server.xml** file inside the **GlobalNamingResources** tags, i.e. before </GlobalNamingResources>, swapping out the host, database name, username and password strings for the correct ones. Please, make sure that the database is not created yet and you have to remember database name, configured in this setting. This name has to be used at the next steps when creating database.

```
<Resource type="javax.sql.DataSource"
name="jdbc/mast"
factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
driverClassName="org.postgresql.Driver"
url="jdbc:postgresql://host:5432/dbname"
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)
"
/>
```

2.3 APPLICATION AND DATABASE DEPLOYMENT

Initialize Database

Fetch a copy of the latest scripts from the GitHub repository:

```
$ git clone https://github.com/MASTUSAID/DB-SCRIPTS.git
```

If you don't have Git installed, you can download database creation script directly:

```
$ wget https://raw.githubusercontent.com/MASTUSAID/DB-
SCRIPTS/master/create_database.sql
```

Login to your postgres server and create a new database. Your connection string will vary depending on your server choice.

```
$ sudo -u postgres psql
postgres=# CREATE DATABASE mast;
```

If you are using RDS, you will have to enable PostGIS support by manually adding several extensions. Follow the instructions in Amazon's documentation for RDS here. You should end up with several new topology schemas.

Restore the database schema and data downloaded from the GitHub repository:

```
postgres=# \c mast
postgres=# \i create_database.sql
```

You may need to specify full path to the **create_database.sql** file if it's not in the current folder. If you have no errors, check to ensure that the database has been populated.

Get DMI Application

You can either clone and compile MAST-DMI application from the GitHub (https://github.com/MASTUSAID/MAST-DMI) or download a compiled version from releases (https://github.com/MASTUSAID/MAST-DMI/releases/). Check for the latest release and download **mast.war** from the assets list. At the time of writing this guide, the latest release is 3.1.

```
$ wget https://github.com/MASTUSAID/MAST-DMI/releases/download/3.1/mast.war
```

If you are planning to do further development of MAST or explore the code, clone and open the project in your favorite Java IDE, supporting Maven.

Before building the application, you may wish to further configure some items, such as reports. MAST reports are configured using Jasper Reports, and all templates can be found under **/src/main/resources/reports** folder. Detailed use of the Jasper report editor is outside the scope of this document, but details and help can be found online at http://community.jaspersoft.com/documentation.

Once you have WAR file available, copy it to Tomcat **webapps** folder.

```
$ sudo cp mast.war /opt/tomcat/webapps/
```

Start the Server

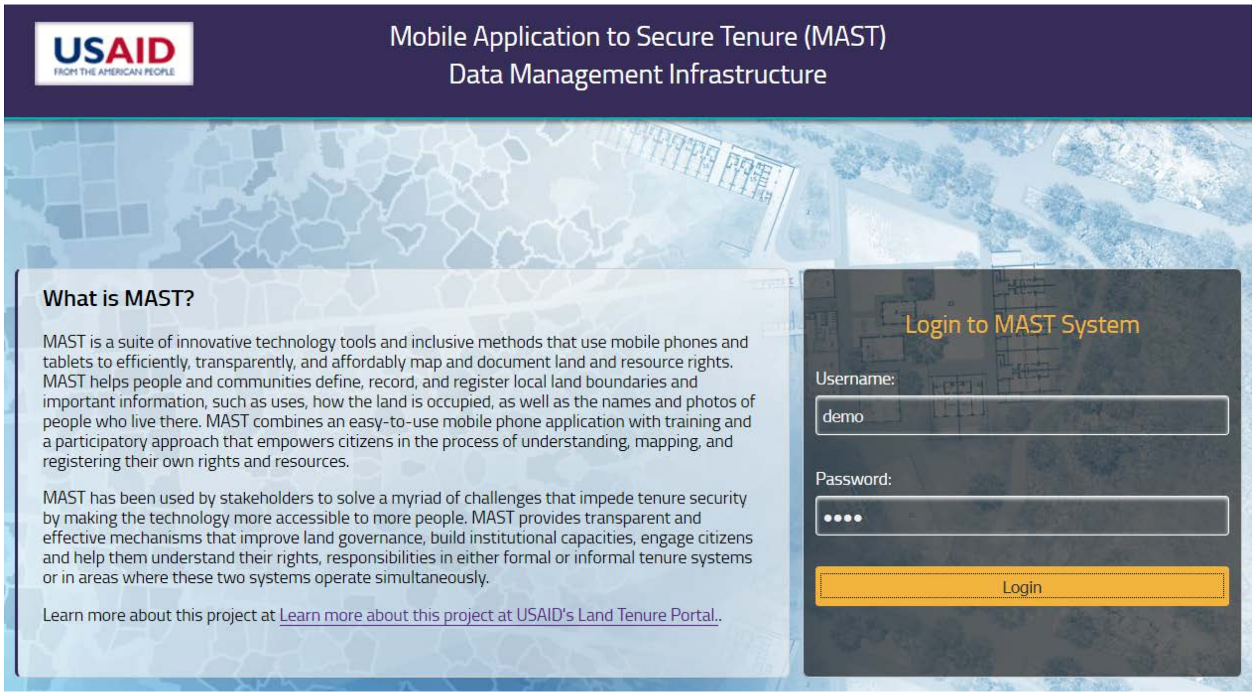
Finally, start your Tomcat server by using the following command:

```
$ sudo /opt/tomcat/bin/startup.sh
```

If you choose, you may also install Tomcat as a systemd service, though this is outside the scope of this guide.

Wait few moments (e.g. 1-2 minutes) and check that the application has been deployed by going to **http://[your_server_IP]:8080/mast**. If you are testing from the local machine, use **localhost** instead of IP address. Please, note that is you didn't configure HTTP port for Tomcat, it will be using 8080 by default. It can be changed in the **server.xml** configuration file.

In case all components were properly installed and configured, you should see login screen.


USAID
FROM THE AMERICAN PEOPLE
Mobile Application to Secure Tenure (MAST)
Data Management Infrastructure
Login to MAST System
Username:
Password:
Login

What is MAST?

MAST is a suite of innovative technology tools and inclusive methods that use mobile phones and tablets to efficiently, transparently, and affordably map and document land and resource rights. MAST helps people and communities define, record, and register local land boundaries and important information, such as uses, how the land is occupied, as well as the names and photos of people who live there. MAST combines an easy-to-use mobile phone application with training and a participatory approach that empowers citizens in the process of understanding, mapping, and registering their own rights and resources.

MAST has been used by stakeholders to solve a myriad of challenges that impede tenure security by making the technology more accessible to more people. MAST provides transparent and effective mechanisms that improve land governance, build institutional capacities, engage citizens and help them understand their rights, responsibilities in either formal or informal tenure systems or in areas where these two systems operate simultaneously.

Learn more about this project at Learn more about this project at USAID's Land Tenure Portal..

Username: demo

Password: ••••

Figure 2 – Login screen

By default, new database was created with user **demo** and password **demo**. This user has full administration rights, use it for initial setup and configuration of your own projects (refer to the user guide for more details). It's highly recommended to change this password to a more secured version or disable/delete demo user in your production. There is also one project configured for your reference and tests.

2.4 GEOSERVER CONFIGURATION

GeoServer is used by MAST DMI for displaying various map layers. You can publish and link to DMI any type of layers, but there are few important ones, which must be configured in GeoServer in order for DMI to function properly. They include land parcels layer, resources layers and area of interest layer.

Before starting GeoServer configuration, open your browser and type in IP address of your Tomcat server with geoserver at the end:

http://localhost:8080/geoserver/

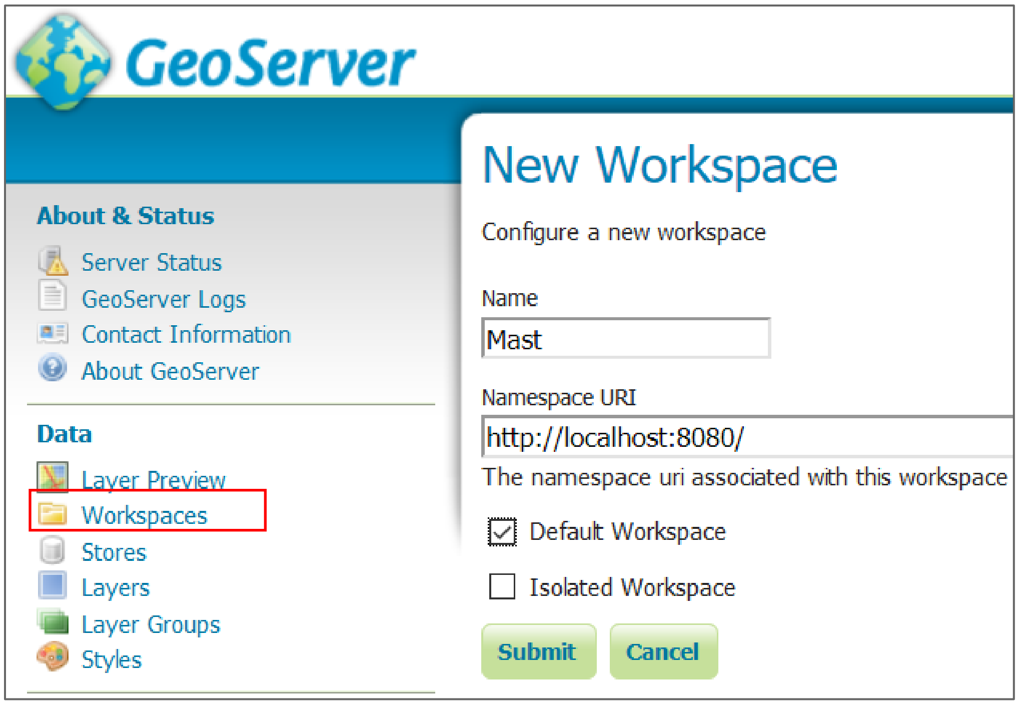
In this example we use localhost, but you can replace it with your IP address.

By default GeoServer has username **admin** with password **geoserver**. Use them for accessing GeoServer, but it's highly recommended to change this password to a more secured version.

Create namespace

First we have to create mast namespace, where all further data stores, layers and styles will be configured. Follow the steps below:

1. Login into GeoServer.
2. Click **Workspaces** in the left menu.
3. Click **Add new workspace** link.
4. Enter the following values:
    a. Name = Mast
    b. Namespace = http://localhost:8080/ (change localhost to IP address of Tomcat server if you want to access DMI from the network).
    c. Tick **Default Workspace**
5. Click **Submit button**



New Workspace

Configure a new workspace

Name: Mast

Namespace URI: http://localhost:8080/

The namespace uri associated with this workspace

[✓] Default Workspace

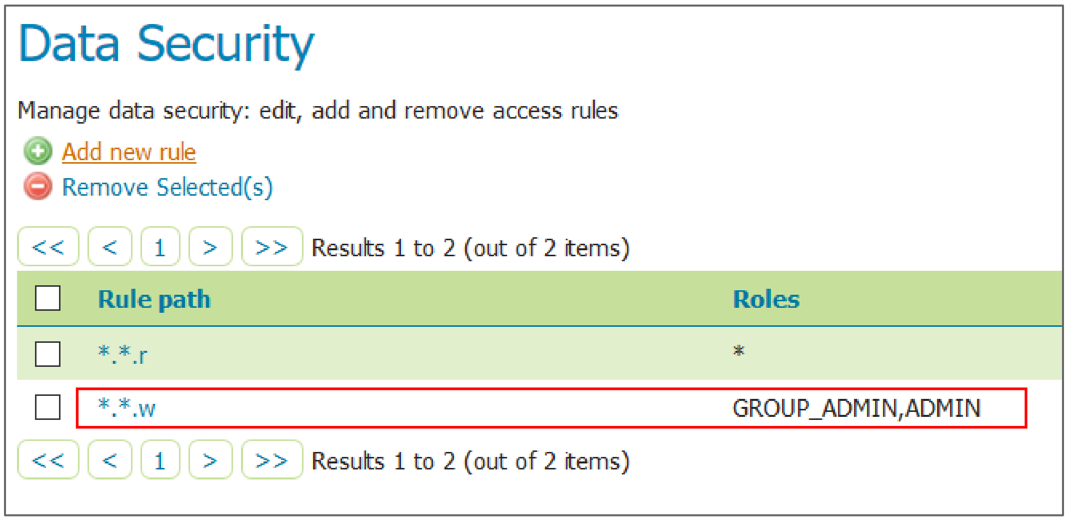
[ ] Isolated Workspace

Figure 3 – Mast workspace

Enable anonymous access

In the newer GeoServer versions, anonymous "write" access is turned off and therefore you won't be able to do spatial editing from MAST DMI. Make sure that anonymous "write" access is enabled by following the steps below:

1. Click on the **Data** option under **Security** main menu.
2. In the list of rules, check if ROLE_ANONYMOUS is present for ***.*.w** rule.



Data Security

Manage data security: edit, add and remove access rules

Add new rule

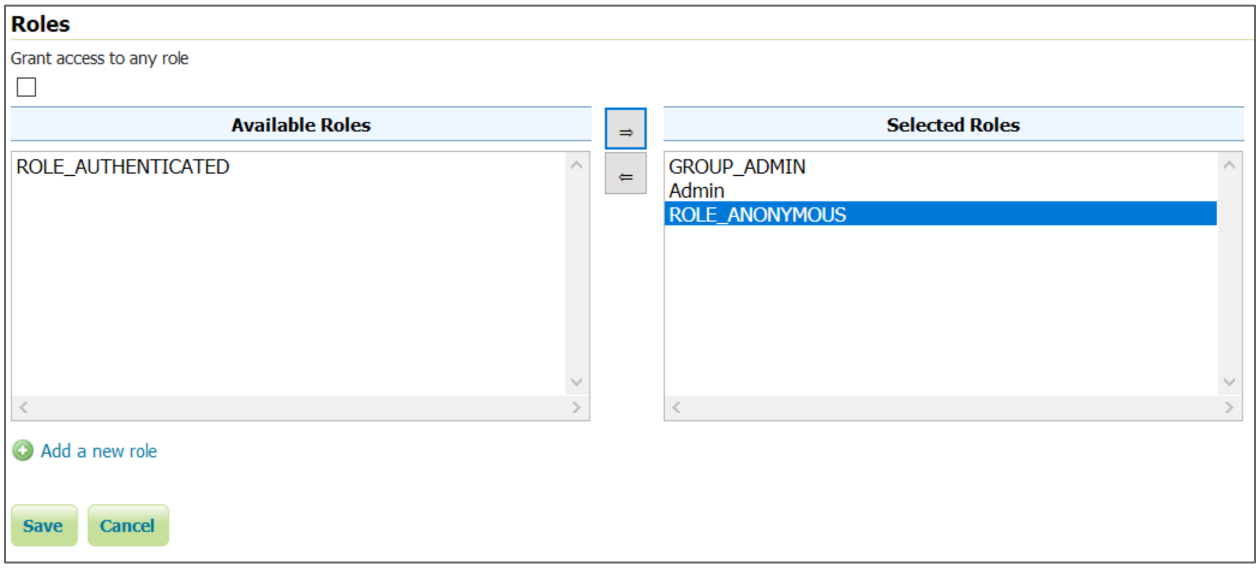
Remove Selected(s)

<< < 1 > >> Results 1 to 2 (out of 2 items)

☐	Rule path	Roles
☐	*.*.r	*
☐	*.*.w	GROUP_ADMIN,ADMIN

<< < 1 > >> Results 1 to 2 (out of 2 items)

3. If anonymous role is missing, click on the ***.*.w** rule.
4. On the next screen select ROLE_ANONYMOUS in the available roles and click add button "=>" to add it into Selected Roles.



Roles

Grant access to any role ☐

Available Roles	Selected Roles
ROLE_AUTHENTICATED	GROUP_ADMIN
	Admin
	ROLE_ANONYMOUS

Add a new role

Save Cancel

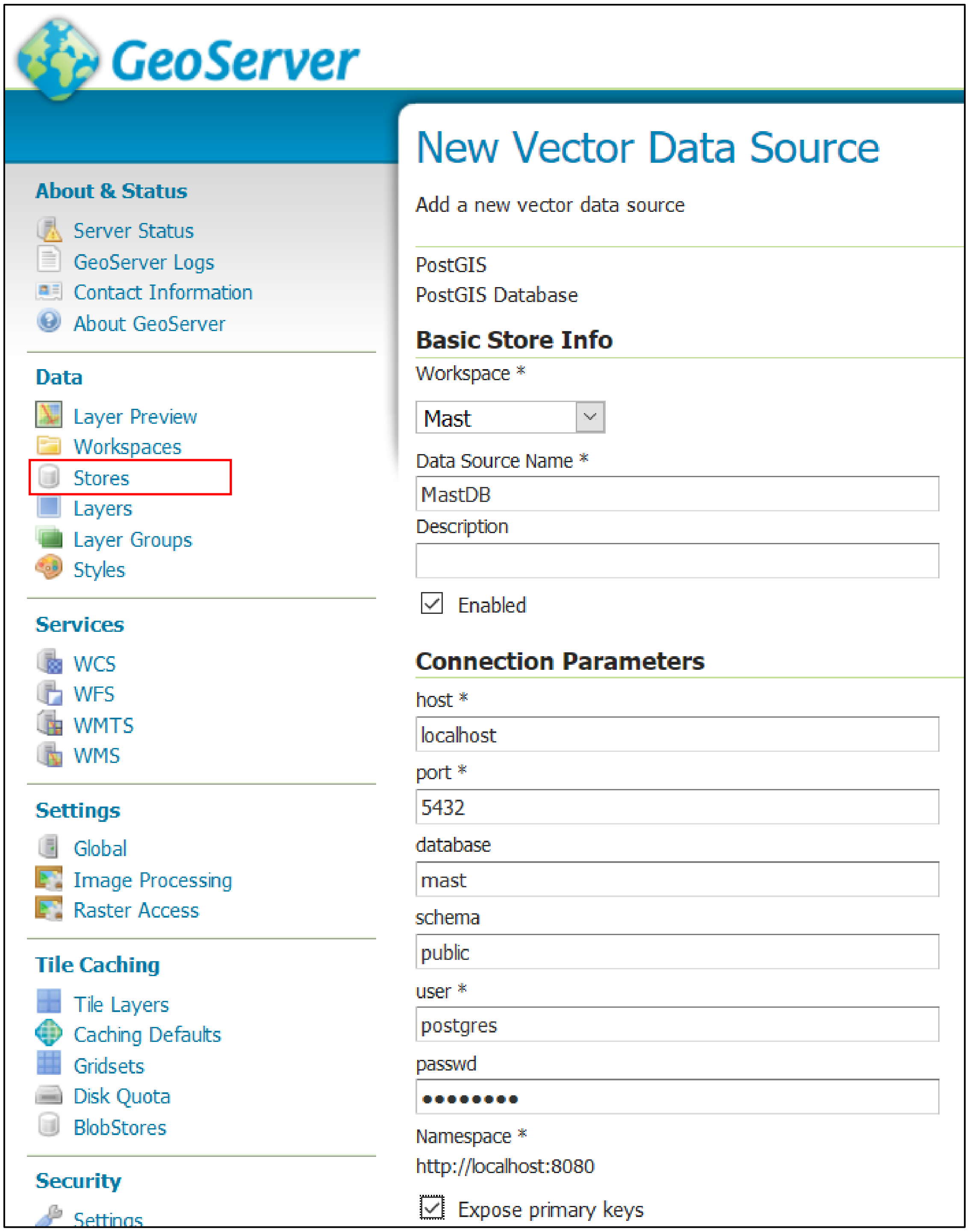
5. Click Save button.

Publish parcels layer

For displaying land parcels on the map control in DMI, it has to be published on GeoServer first. Created MAST database already has settings to for the minimum set of layers, configured to use **localhost** address. If you are testing your server from a different computer, layers IP address has to be changed in the database. Check "**Adjust layers URL**" chapter for more details.

Start publishing parcels layer by creating new data store:

1. Click **Stores** in the main menu.
2. Click **Add new store**.
3. Click on **PostGIS** vector data source.
4. Enter the following parameters:
    a. Workspace = Mast
    b. Data Source Name = MastDB
    c. Host = localhost (or IP address if your database on the different server)
    d. Port: 5432
    e. Database = mast
    f. User = postgres
    g. Password = [your_postgres_password]
    h. Expose primary keys = yes (ticked)
5. Click **Save** button



New Vector Data Source

Add a new vector data source

PostGIS
PostGIS Database

Basic Store Info

Workspace *: Mast

Data Source Name *: MastDB

Description:

[✓] Enabled

Connection Parameters

host *: localhost

port *: 5432

database: mast

schema: public

user *: postgres

passwd: ••••••••

Namespace *: http://localhost:8080

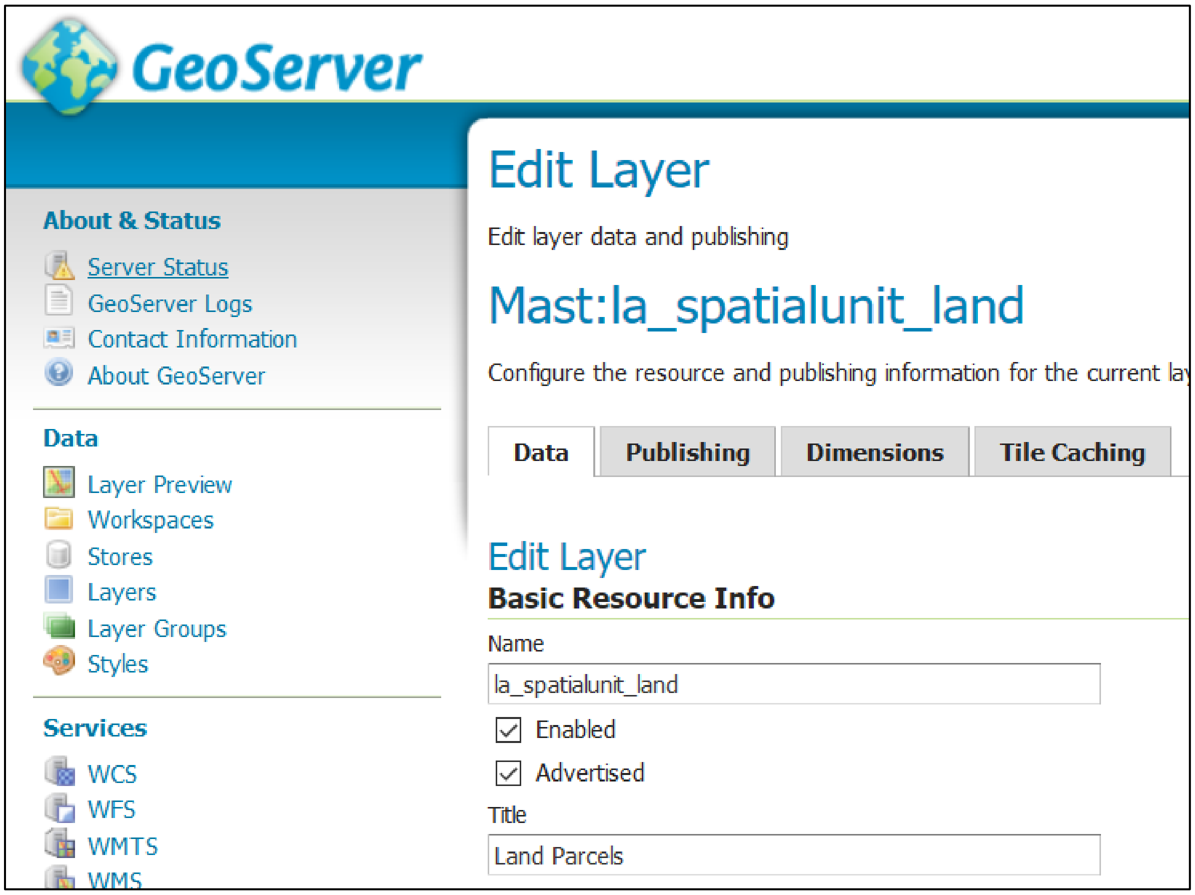
[✓] Expose primary keys

Figure 4 – New store

If all parameters provided correctly, you will see next screen with the list of tables, available in the database. If you have any failure, check chapter 2.4 Known Issues for more details regarding PostgreSQL driver in GeoServer.

Go through the listed tables and find **la_spatialunit_land**. Click **Publish** link against this table. Alternatively you can go through **Layers** menu and select **Add a new layer** option. Make sure the following parameters are set for the land parcels layer:

1) Name = la_spatialunit_land
2) Ttile = Land Parcels
3) Native SRS = EPSG:4326
4) Declared SRS = EPSG:4326
5) Click **Compute from SRS bounds** link to calculate bounding box. You can use **Compute from data** link if you have already some data.
6) Click **Compute from native bounds** link to set **Lat/Lon Bounding Box**
7) Click **Save** button



Edit Layer

Edit layer data and publishing

Mast:la_spatialunit_land

Configure the resource and publishing information for the current la

Edit Layer

Basic Resource Info

Name

la_spatialunit_land

✓ Enabled

✓ Advertised

Title

Land Parcels

Figure 5 – Parcels layer properties

After successful saving, it will appear in the list of layers (under **Layers** menu).

Publish area of interest layer

Area of interest area layer is used for displaying designated areas, which can be further assigned to specific para-surveyor for limiting his area of field surveys. Since we already created MAST database source, we simply need to publish a new layer. Follow these steps below:

1. Click **Layers** menu.
2. Click **Add a new layer** link.
3. Select **Mast:MastDB** data source.
4. Look for **la_spatialunit_aoi** table and click **Publish** link against this table.
5. Provide the following values:
    a. Name = la_spatialunit_aoi
    b. Title = AOI
    c. Native SRS = EPSG:4326
    d. Declared SRS = EPSG:4326
    e. Click **Compute from SRS bounds** link to calculate bounding box. You can use **Compute from data** link if you have already some data.
    f. Click **Compute from native bounds** link to set **Lat/Lon Bounding Box**
6. Click **Save** button

Publish resource (land) layer

MAST allows capturing various resource types in the field. They can be in the form of polygon, line or point. Follow these steps below to publish polygon type of resources (land):

1. Click **Layers** menu.
2. Click **Add a new layer** link.
3. Select **Mast:MastDB** data source.
4. Look for **la_spatialunit_resource_land** table and click **Publish** link against this table.
5. Provide the following values:
    a. Name = la_spatialunit_resource_land
    b. Title = Resource Land
    c. Native SRS = EPSG:4326
    d. Declared SRS = EPSG:4326
    e. Click **Compute from SRS bounds** link to calculate bounding box. You can use **Compute from data** link if you have already some data.
    f. Click **Compute from native bounds** link to set **Lat/Lon Bounding Box**
6. Click **Save** button

Publish resource (line) layer

For publishing line resources, follow these steps below:

1. Click **Layers** menu.
2. Click **Add a new layer** link.
3. Select **Mast:MastDB** data source.
4. Look for **la_spatialunit_resource_line** table and click **Publish** link against this table.
5. Provide the following values:
    a. Name = la_spatialunit_resource_line
    b. Title = Resource Lines
    c. Native SRS = EPSG:4326
    d. Declared SRS = EPSG:4326
    e. Click **Compute from SRS bounds** link to calculate bounding box. You can use **Compute from data** link if you have already some data.
    f. Click **Compute from native bounds** link to set **Lat/Lon Bounding Box**

6. Click **Save** button

Publish resource (point) layer

For publishing point resources, follow these steps below:

1. Click **Layers** menu.
2. Click **Add a new layer** link.
3. Select **Mast:MastDB** data source.
4. Look for **la_spatialunit_resource_point** table and click **Publish** link against this table.
5. Provide the following values:
    a. Name = la_spatialunit_resource_point
    b. Title = Resource Points
    c. Native SRS = EPSG:4326
    d. Declared SRS = EPSG:4326
    e. Click **Compute from SRS bounds** link to calculate bounding box. You can use **Compute from data** link if you have already some data.
    f. Click **Compute from native bounds** link to set **Lat/Lon Bounding Box**
6. Click **Save** button

Publish vertices layer

Vertices layer is used for showing land parcel vertices, when producing various print outs (e.g. land certificate). Therefore it has to be published on GeoServer from the **vertexlabel** table. Follow these steps below:

1. Click **Layers** menu.
2. Click **Add a new layer** link.
3. Select **Mast:MastDB** data source.
4. Look for **vertexlabel** table and click **Publish** link against this table.
5. Provide the following values:
    a. Name = vertexlabel
    b. Title = vertexlabel
    c. Native SRS = EPSG:4326
    d. Declared SRS = EPSG:4326
    e. Click **Compute from SRS bounds** link to calculate bounding box. You can use **Compute from data** link if you have already some data.
    f. Click **Compute from native bounds** link to set **Lat/Lon Bounding Box**
6. Switch to **Publishing** tab and select **point** style in the **Default Style** dropdown box.
7. Click **Save** button

Adjust layers URL

Default MAST database comes with predefined list of layers, configured for http://localhost:8080 address. If you have different port number or planning to test MAST application from the network, using real IP address of the server, you need to update you layers in the **la_spatialsource_layer** table. Simply run the following command on your MAST database:

update la_spatialsource_layer set location = 'http://*your_ip_address:port_number*/geoserver/wfs?';

Change "your_ip_address" and "port_number" to appropriate values, according to your configuration. Commands on Ubuntu:

```
$ sudo -u postgres psql
postgres=# \c mast
mast=# update la_spatialsource_layer set location =
'http://your_ip_address:port_number/geoserver/wfs?';
```

2.4 KNOWN ISSUES

Java

When installing Tomcat on Windows system with using Java 8, there might be an issue with starting MAST DMI application. In the log files of Tomcat you can find the following error:

Property 'passwordEncoder' threw exception; nested exception is org.jasypt.exceptions.EncryptionOperationNotPossibleException

In order to resolve it, download Java 7 Cryptography Extensions from the following link:

http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

Extract and copy over downloaded files into your JRE installation folder under **lib/security** (e.g. **C:\Program Files\Java\jre1.8.0_101\lib\security** and/or **C:\Program Files\Java\jdk1.8.0_101\jre\lib\security**). They have to be placed into jre folder according to your JAVA_HOME configuration.

GeoServer JDBC driver for PostgreSQL

GeoServer comes with PostgreSQL JDBC driver and sometimes it may not match your database version. Since we are deploying JDBC driver into Tomcat libraries, you can try and delete this driver from the GeoServer folder. Geoserver folder will be created under Tomcat **webapps** folder after you start your Tomcat for the first time. Go to the GeoServer's lib folder(e.g. /opt/tomcat/webapps/geoserver/WEB-INF/lib) and delete **postgresql** file (e.g. postgresql-42.1.1.jar).

After deleting the file, you will need to restart your Tomcat. Make sure to give some time (e.g. 30 sec) before starting Tomcat again.

```
$ sudo /opt/tomcat/bin/shutdown.sh
$ sudo /opt/tomcat/bin/startup.sh
```

3. MAST Mobile Application

3.1 APK SETUP

The easiest way to get started with MAST mobile application is to download it from the latest releases on GitHub. Check https://github.com/MASTUSAID/MAST-MOBILE/releases/ and look for the latest release. Download **mast.apk** from the assets list, copy to your Android mobile device and run for installation. At the time of writing this guide, the latest release is 3.1, available at https://github.com/MASTUSAID/MAST-MOBILE/releases/download/3.1/mast.apk.

3.2 FRESH COMPILATION

IDE setup

First, download and install Android Studio from https://developer.android.com/studio/index.html. During setup, you will be prompted to download any versions of the Android SDK that you wish to use. For the purposes of MAST, download the SDK for Android 6.0 Nougat (API level 23).

If you wish to test the application before building, create an Android Virtual Device to ensure your install works properly. Click *Tools* > *Android* > *AVD Manager*, and click the *Create Virtual Device* button. Choose a phone and SDK level to emulate, such as the Pixel with the latest Android version. Before clicking *Finish*, click on *Show Advanced Settings*. Under *Memory and Storage,* ensure that the SD card is set to *Studio-Managed* and is at least 200 MB.

Configure, Compile, and deploy application

To build the application package, first fetch the source code from GitHub:

```
$ git clone https://github.com/MASTUSAID/MAST-MOBILE.git
```

As of this writing, several settings are hardcoded into the application before compiling. In the Studio project explorer, open *Java* > *Util* > *CommonFunctions.java*. Set the SERVER_IP string to the IP of your DMI server and save the file.

If you are using Android Studio version greater than 2.2.3, open the gradle.build file and ensure that the following value is set for the gradle version:

```
dependencies {
        classpath 'com.android.tools.build:gradle:2.3.3'
    }
```

To build the application package (APK file), select *Build* > *Generate Signed APK*. Android Studio will request that you create a signed keystore and key file to use to sign the application, if you have not already. Follow the onscreen instructions. Further information is available online.

Once you have the final APK, transfer a copy to each device that you plan on using. To install the application, use a file browser application to open and run the APK. Note that you will have to ensure that the devices are set to run applications from alternative sources (i.e. not the Google store), by going to the devices security settings. The exact location of the setting varies from device to device.

3.2 KNOWN ISSUES

Google Maps API Key

When compiling MAST-MOBILE project, Google Maps API key may not be working and it will require replacement with a new key. Open **AndroidManifest.xml** file and replace value for the following setting:

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="XXXXXXXXXXXXXXXXXX" />
```

Put new key instead of XXXXXXXXXXXX value.