SOLA Community Server & Open Tenure

Administration Guide

for Release 2020a

Revision History

Date	Version	Description	Author
May 30, 2016	0.1	Initial Draft	Andrew McDowell, Alexander Solovov & Neil Pullar
May 8, 2018	1.0	Correcting & updating Ubuntu details & adding Intel ConnectStick details	Neil Pullar
May 31, 2020	2.0	Initial draft of FAO commissioned upgrade & update	Neil Pullar
June 12 2020	2.1	FAO feedback and additional Email Service details	Neil Pullar
July 14 , 2020	2.2	Updated to reflect latest version of Payara5 version 2020-2 (build 547)	Neil Pullar
October 2, 2020	2.3	Updates of installation components versions and references	Alexander Solovov
Dcember 1, 2020	2.4	Updates to Docker installation procedure	Alexander Solovov

Table of Contents

1. About this Guide

1.1 Licensing

The source code for the SOLA Community Server, Web Admin and Open Tenure Android (& iOS) is licensed under the Berkeley Software Distribution (BSD) 3-clause license (a.k.a. New BSD or Modified BSD).

This guide describing system administration functions and processes associated with Community Server and Web Admin is considered part of these software applications and hence covered by the same license. The details of this license are:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the FAO nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

2. Server Installation on Windows

This chapter assumes the SOLA server components are being installed on a 64bit Windows Desktop computer running Windows 10 (on either 32bit and 64bit Windows operating systems). If the operating system architecture is 32bit, use 32bit installers in place of the 64bit installers noted in the installation steps.

2.1 Download Installers

The SOLA deployment package containing the SOLA services EAR file and database build files can be downloaded from:

- http://github.com/OpenTenure/latest-release [SOLA Community Server & Web Admin]

Alternatively if you decide to do a Docker based installation you only need to download these github repositories:

- http://github.com/OpenTenure/docker [[SOLA Community Server & Web Admin]

Once you have downloaded the docker github repository go directly to Section **Error! Reference source not found.** (for Windows 10 Professional) or Section **Error! Reference source not found.** (for Ubuntu 18.04)**Error! Reference source not found.** to complete a Docker installation[1].

For a conventional (non-Docker) installation, you will also require a number of installation packages for the components supporting the SOLA applications:

- Latest update of Java 8 OpenJDK (64bit) (https://adoptopenjdk.net)
- Payara Server Community Edition (Full) - Version 5.2020.4 is recommended. The most recent stable version of Payara Server is available at https://www.payara.fish/downloads/payara-platform-community-edition/
- The most recent release of PostgreSQL 12. Download 64bit version (Win x86-64) (https://www.enterprisedb.com/downloads/postgresql)
- The latest release of Postgis 3 for PostgreSQL 12. Download the pg12x64 version executable (http://download.osgeo.org/postgis/windows/pg12/)
- GeoServer (http://geoserver.org/download/). Version 2.18.0 (Web archive - WAR file) is used for this guide.
- Notepad++ (https://notepad-plus-plus.org/downloads/) Text editor to help with updates to configuration files (**Windows installations only)**.
- If you are installing on a recently built Windows Operating System, you will also need to install the Visual C++ Redistributables. These are required for the Postgres uuid-ossp extension. The Visual C++ Redistributables you require is the vc_redist.x86.exe from https://www.microsoft.com/en-us/download/details.aspx?id=48145.

2.1.1 Visual C++ Redistributable & Notepad++

1) Install the **Visual C++ Redistributable** package on the server by right clicking the vc_redist.x86.exe installer and choosing Run as Administrator.
2) Optionally install **Notepad++** if you don't have a suitable plain text editor already installed.

2.2 Java 8

2.2.1 OpenJDK 8

1) If there is an existing installation of Oracle Java, uninstall it using the regular Windows uninstall process (Click Start, select Settings-System-Apps & features, select Java program and click on Uninstall button)

[1] Can also be installed on Apple Macs – install Docker Desktop Community Edition (Section 2.6.1) and then as described for the Ubuntu installation from Section 3.6.3 onwards

2) Download the latest msi installation file for OpenJDK 8 64bit for Windows from https://adoptopenjdk.net
3) Run the AdoptOpenJDK msi file (select, right click, I**nstall**) and include "Set JAVA_HOME variable" in addition to the default "JDK with Hotspot" to be installed when prompted.
4) Check the JAVA_HOME environment setting. In File Manager, right click on **This PC → Properties → Advanced system settings → Environment Variables** and check that
    a. there is no JAVA_HOME in your user variables (top panel)
    b. the JAVA_HOME variable as a system variable (bottom panel) is **C:\Program Files\AdoptOpenJDK\jdk-8.0.252.09-hotspot** - will change as new versions of OpenJDK are released.

2.2.2 Oracle Java

Alternatively to installing OpenJDK Java you can install the Oracle version of Java, noting that as of 2020, Oracle have indicated that there will be a charge to use their version of Java.

1) Run the Java 8 JDK 64bit installer by right clicking the installer and choosing Run as Administrator. You should exclude the installation of the JDK Source Code. Use the default install location.

2.3 PostgreSQL

To run the install for PostgreSQL you will need administrator privileges on your computer.

1) Run the PostgreSQL 12 installer by right clicking the installer and choosing Run as Administrator.
2) Leave the default location for the installation directory (C:\Program Files…), but you can change the data directory to use a folder outside of C:\Program Files … such as C:\PostgreSQLdata\12
3) At the Password form, enter a suitable password for the postgres superuser. You will need to use this password frequently, so ensure it is a password you will remember.
4) Use the default port of 5432 and the default locale
5) Proceed with the installation – it may take several minutes. Once complete, close the installer – **do not run StackBuilder at this time**.
6) Verify PostgreSQL has been installed correctly by starting pgAdmin (Start menu → All Programs → PostgreSQL13 → pgAdmin4) and attempting to connect to the new database [In File Manager locate the pgAdmin4 executable and right click and **Pin to Start**]
    a. If a PostgreSQL 12 (localhost:5432) server is not listed under the Servers node, in the Object Browser view, click the plug tool in the top left of the tool bar and enter the details for your PostgreSQL server.
    b. Right click this database and choose Connect
    c. When prompted, enter your postgres password.
7) You will be informed when there is a new version of pgAdmin4. To install these new updates go to https://www.postgresql.org/ftp/pgadmin/pgadmin4, select the latest version and then the windows version to download the installation file (.exe file type). Right click this file once downloaded, select **Run as administrator** and this will overwrite the old version and install the new version of pgAdmin4.

If you encounter issues while installing PostgreSQL, or you get an unexpected error while trying to connect to the PostgreSQL database, you may need to uninstall and reinstall Postgres. If you do need to reinstall, follow these steps

1) Uninstall PostgreSQL using the Control Panel
2) Locate the PostgreSQL installation directory (e.g. in Program Files) and delete it completely

3) When you re-run the installation, choose the install location to be C:\ rather than C:\Program Files.

2.3.1 PostGIS

1) Run the PostGIS 3 installer by right clicking the installer and choosing Run as Administrator.
2) When installing the PostGIS component, **clear/uncheck** the Create spatial database option on the Choose Components form. If you do not uncheck this option, you can simply delete the additional spatial database at a later time.
3) If prompted to update pgAdmin during the file copy process, you can optionally do so.

2.3.2 Configure PostgreSQL to allow access from other computers

1. By default PostgreSQL is locked down to prevent access from computers other than the localhost. If your database server will be used as the production server, then it may be a good option to leave PostgreSQL in this locked down state. If the database will be used for testing or training, it may be desirable to access the database server from other computers. To allow these connections it is necessary to make some configuration changes.
   Edit the **pg_hba.conf** file (by default in the ..\main subfolder of the PostgreSQL Data folder) and add the following line to this file (change to your IP address range)
   host all all 192.168.1.1/24 md5
2. Configure PostgreSQL to accept connections from SOLA applications.
   This can be achieved from psql with the ALTER SYSTEM Command (as Administrator) type:
   sudo su - postgres
   psql
   # ALTER SYSTEM SET listen_addresses='*';
   \q
   $ logout

2.3.3 Add to Path Environment

1) In File Manager, right click on **This PC** and navigate Properties → Advanced system settings → Environment Variables → System Variables and select **PATH,**
2) Click **Edit,** add then the **New** button and add C:\Program Files\PostgreSQL\12\bin and **OK**
3) Click on **New** and C:\Program Files\PostgreSQL\12\lib and **OK** (3 times) (to exit Advanced system settings
4) Restart computer

2.3.4 Create SOLA Database

Where there is no configured PostgreSQL database backup and it is the very first time you create a new SOLA database in PostgreSQL, follow these steps:

- Locate the database folder in the Community Server deployment package, copy to the computer hosting PostgreSQL (if necessary) and **Run as Administrator** the **create_database.bat** build script in the **database** folder.
- A Command Prompt will display and ask a sequence of questions about the database connection. Use the default values shown in square brackets, except for the DB Password and Create or Replace the database question (answer Y).
- You should fill the database with sample data initially to help test all components are correctly configured. You can rerun the create_database script at any stage to rebuild a clean database without the sample data.

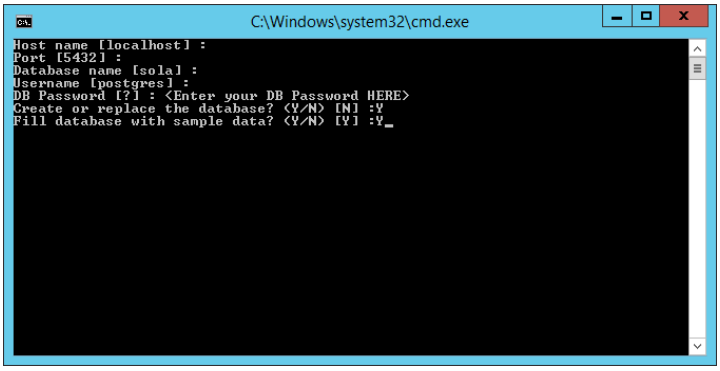


Figure 1 – Enter create_database script

The script should take about a minute to run. If it only takes a few seconds, it will indicate an issue with script. Open the build.log file (it will be in the same directory as the create_database script) to see the error/warning messages.

Known error/warning messages

- *The system cannot find the path specified* means the psql.exe could not be located. **Edit** create_database.bat in your preferred text editor (e.g. Notepad++),

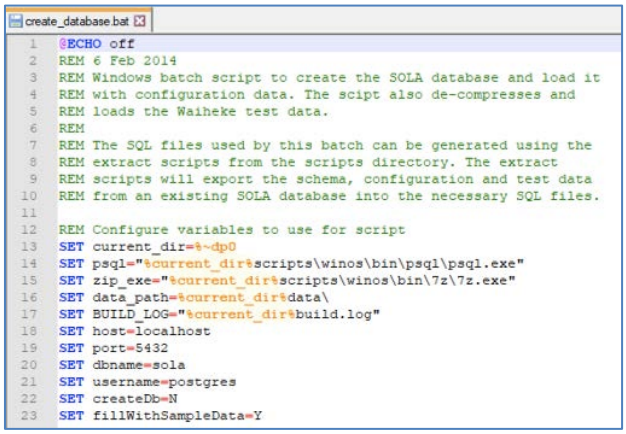


Figure 2 – To edit line 14 of create_database.bat

Go to line 14 and replace that line that includes the folder path for the psql executable in your Postgresql installation.
e.g. SET psql="C:\Program Files\Postgresql\12\bin\psql.exe"
Save the changes and re-run create_database.bat.

2.3.5 Alternative approach to SOLA database creation

Where there is an existing Community Server setup and you want to replicate the same configuration and database for a new Community Server setup, follow these steps:

1. Using the PG Admin tool with a connection to an existing Community Server SOLA database, make a **TAR** backup of the complete SOLA database of interest
   OR using SOLA Web Admin, make a database backup.
   Copy the database backup file to a convenient location on the server to be used for the new Community Server setup
2. Using PG Admin, right click on the PostgreSQL Server (that refers to Port 5432) and create a new database with **Name** sola, **Owner** Postgres and (on **Definition** tab) **Template** as template1
3. Right click on the newly created **sola** database and select **Restore**. Browse for backup file created in step 1 and the click on **Restore** button.

4. To clear all claim records in the backup from the exisiting Community Server use PG Admin, right click on the **sola** database which has been restored and select the Query Tool. Add the following script to the Query panel and click on the **Execute** icon (lightning icon).

```
delete from opentenure.claim_comment;
delete from opentenure.claim_comment_historic;
delete from opentenure.claim_uses_attachment;
delete from opentenure.claim_uses_attachment_historic;
delete from opentenure.attachment_chunk;
delete from opentenure.attachment;
delete from opentenure.attachment_historic;
delete from opentenure.claim_location;
delete from opentenure.claim_location_historic;
delete from opentenure.party_for_claim_share;
delete from opentenure.party_for_claim_share_historic;
delete from opentenure.claim_share;
delete from opentenure.claim_share_historic;
delete from opentenure.claim;
delete from opentenure.claim_historic;
delete from opentenure.party;
delete from opentenure.party_historic;
alter sequence opentenure.claim_nr_seq restart with 1;
```

Figure 3 – SQL script to clear any existing claims

5. Update the SOLA database schema (if there have been new releases of the software since the original setup.

   In PGadmin, navigate through the left hand column and right click on **SOLA PostgreSQL Server-Databases-sola-schemas-system-version** and select **View/Edit – All data**. Note what was the latest update (the version records follow the convention YYMM_version)

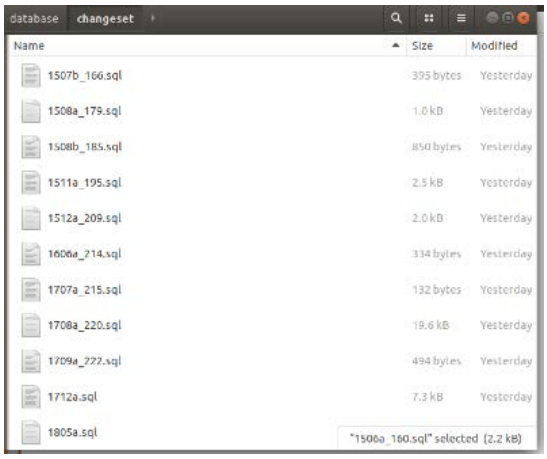


Figure 4 – PGadmin listing of system.version database table

   In Figure 4 the latest version record is **1805a.sql** (database update)
6. Check the latest version of the (SOLA) database folder that you have "cloned" from http://github.com/OpenTenure/database. Look in the **database/changescript** folder of this download from github and see if there are any more recent database update script.
7. If there are more recent changes, sequentially run those update scripts in PGadmin:
   - In PGadmin, select the sola database, right mouse click and select **Query tool.**
   - in File Manager, double click the oldest of the unrun scripts, select all the text (CTRL + A) and copy (CTL + C) it
   - Paste into the PGadmin Query tool panel

- Click on the run icon (Lightning symbol)
- Check that it runs without error

Repeat for all the more recent update scripts in the database/changescript folder.

In PGadmin repeat step 5, the query of the (system) version table (right mouse click on version table and select **View/Edit – All data**. This should now include all the most recent updates.

2.4 Payara Server 5

2.4.1 Installation

Payara Server version 5.2020.4 was the current version of Payara in October 2020 and has been used as the application server for SOLA Community Server, SOLA Web Admin, Geoserver and Jasper Reports.

Payara Server can be installed in two ways. The easiest way to install an appropriately configured Payara Server instance is to copy an already configured instance (remembering to stop Payara Server before making the copy).

Option 1 – Use a copy of previously configured instance of Payara Server 5 for SOLA Community Server

In this option, you only need to change the password for your SOLA PostgreSQL database:

1) Edit the **domain.xml** file as Administrator to be found:
   C:\payara5\glassfish\domains\domain1\config directory
   Search for the string "**<property name="Password" value="** about line 84

```
58    <jdbc-connection-pool datasource-classname="org.postgresql.ds.PGConnectionPoolDataSource" name="sola" res-type=
59      <property name="TargetServerType" value="any"></property>
60      <property name="BinaryTransfer" value="true"></property>
61      <property name="UnknownLength" value="2147483647"></property>
62      <property name="DisableColumnSanitiser" value="false"></property>
63      <property name="UseSpNego" value="false"></property>
64      <property name="SspiServiceClass" value="POSTGRES"></property>
65      <property name="ProtocolVersion" value="0"></property>
66      <property name="SendBufferSize" value="-1"></property>
67      <property name="ReceiveBufferSize" value="-1"></property>
68      <property name="LoadBalanceHosts" value="false"></property>
69      <property name="ReadOnly" value="false"></property>
70      <property name="LogUnclosedConnections" value="false"></property>
71      <property name="GssLib" value="auto"></property>
72      <property name="DefaultRowFetchSize" value="0"></property>
73      <property name="AllowEncodingChanges" value="false"></property>
74      <property name="PrepareThreshold" value="5"></property>
75      <property name="SocketTimeout" value="0"></property>
76      <property name="HostRecheckSeconds" value="10"></property>
77      <property name="ConnectTimeout" value="10"></property>
78      <property name="PreparedStatementCacheQueries" value="256"></property>
79      <property name="Ssl" value="false"></property>
80      <property name="PreparedStatementCacheSizeMiB" value="5"></property>
81      <property name="LoginTimeout" value="0"></property>
82      <property name="ServerName" value="localhost"></property>
83      <property name="TcpKeepAlive" value="false"></property>
84      <property name="password" value="sola2018"></property>
```

Figure 5 – domain.xml (password defined on line 84)

In the screenshot above, the password is shown as **sola2018**. You need to edit the password to reflect your chosen password (and then make sure you remember it).

2) Check that there are no Payara Server or Payara instances already running (running Command Prompt as Administrator):
   $ cd \payara5\bin
   $.\asadmin list-domains
   A message should display

domain1 not running
list-domains executed successfully

3) Start the SOLA domain, **domain1**.
   Using the cd command, navigate to the ..\payara5\bin folder
   $ cd \payara5\bin
   $.\asadmin start-domain domain1
4) To check that Payara is running correctly browse to the Payara Server Admin console at **http://localhost:4848 (or http://domainname.org:4848 or http://ipaddress:4848)** in your web browser.
5) Stop Payara Server with the **Stop** button in Payara Server Admin Console, skip the next section (on how to edit the domain.xml file of a copy of a previously configured instance of Payara Server 5) and proceed to Section 2.4.2.

Option 2 – Make a fresh Payara Server installation and configure it for SOLA Community Server

1) Download the Payara Server zip bundle from the web site
   https://www.payara.fish/downloads/payara-platform-community-edition/
   This download is approximately 150Mb
2) Unzip (and move or copy) into:
   C:\payara5
3) The .NET Framework 3.5 is required to create the Payara Server Windows Service. Ensure the .NET Framework 3.5 Features are installed/enabled on your computer. This can be done using the Add Roles and Features Wizard on Server 2008 and Server 2012. See https://technet.microsoft.com/en-us/library/dn482071.aspx for details.

   The .NET Framework 3.5 should be enabled on Windows 10 by default. You can check from **Control Panel → Programs and Features** and clicking the **Turn Windows features on or off** link. The Windows Features dialog should show the .NET Framework with a blue square in the checkbox as shown below. To access Programs and Features in Windows 10, right click the Windows icon in the bottom left of the screen and select **Programs and Features** from the popup menu.

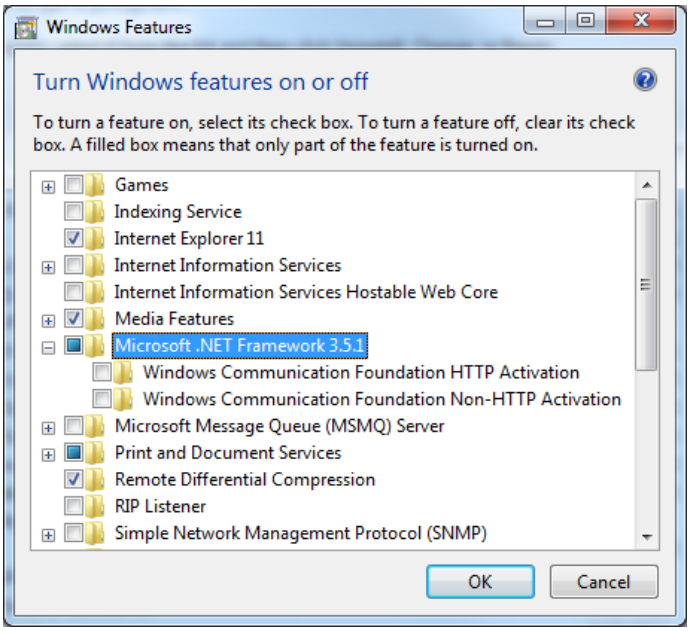


Figure 6 – Turn Windows features on or off

4) Download the appropriate PostgreSQL jdbc 4.2 jar file from https://jdbc.postgresql.org/download.html (eg postgresql42.2.16.jar) and copy it into:
   C:\payara5\glassfish\domains\domain1\lib

5) Download the 4 missing jackson JAR files from https://www.dropbox.com/sh/9uf3rbicau2i1uz/AACuFWjjI31ObaHIuAzELchFa?dl=0 and copy the 4 jackson JAR files into this folder: C:\payara5\glassfish\modules

6) It is advisable to explicitly set the location of the JDK for Payara Server to use. Check location of JDK (through the JDK_HOME system environment variable) as described in Section 2.2. If OpenJDK has been installed this should be:
C:\Program Files\AdoptOpenJDK\jdk-8.0.232.09-hotspot\bin
Edit this file:
C:\payara5\glassfish\config\asenv.bat
Add a new line at the end of this file
set AS_JAVA=<fully qualified path to your JDK>
eg. set AS_JAVA=C:\Program Files\AdoptOpenJDK\jdk-8.0.232.09-hotspot
OR (if you have used Oracle Java)
set AS_JAVA=C:\Program Files\Java\jdk1.8.0_231

2.4.2 Run Payara Server as Service

1) To establish Payara Server as a service open Windows PowerShell or a Command Prompt as an Administrator and change to the Payara bin folder
cd C:\payara5\bin
.\asadmin create-service --name PayaraSOLA domain1
This will create a Windows Service called PayaraSOLA

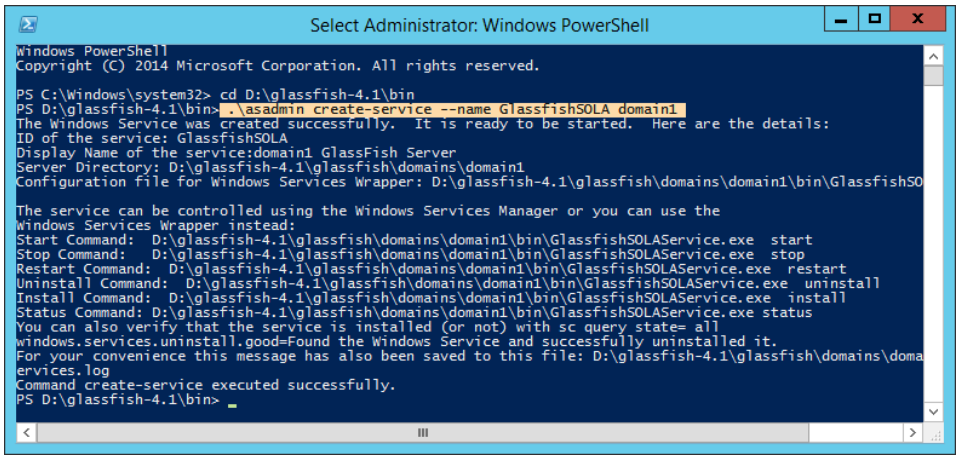


Figure 7 – PowerShell display to create service

2) If you are using Powershell type the following two commands to change the service display name and start it.
Set-Service -Name PayaraSOLA -DisplayName "Payara SOLA"
Start-Service -Name PayaraSOLA

3) If you are using a Command Prompt type:
sc config PayaraSOLA DisplayName= "Payara SOLA"
sc start PayaraSOLA
If you are familiar with the Windows Services Control Panel, you will be able to use that to start and stop the new PayaraSOLA service.

4) Stop the service (and hence Payara Server 5) before proceeding.

2.4.3 Get and install domain SSL certificate

If SOLA Community will be accessed through the internet by way of reference to a domain name (e.g. https://opentenure.org) then it is necessary to purchase the domain name from a domain registry (such as godaddy.com or cloudflare.com) and protect that domain with a SSL certificate. This involves the following steps for the GoDaddy domain registry and so with other domain registries there may be minor variations:

1) On your Domain Registry website, log on to your account and register your domain name (if you do not have any) and setup DNS records to point to your server IP address.
2) Open command line and navigate to the bin folder of Java JDK (e.g. C:\Program Files\Java\jdk1.8.0_231\bin).
3) Generate new entry in **keystore.jks** with information of your domain:

   keytool -keysize 2048 -genkey -alias *www.yourdomain.com* -keyalg RSA –dname "CN=*www.domain.com*,O=*company*,L=*city*,S=*State*,C=*Country*" -keystore keystore.jks
   Enter password of your keystore when asked. (Please note that the GoDaddy domain registry requires at least 2048 bits keysize.) **CN** is your sites domain name, **O** is your company name, **L** is the city, **C** is the 2 character country code. There are more options you could specify if you want. But these are enough. **alias** is the key you will use to refer this certificate.

4) Create the request file for submitting to Godaddy (or any other Domain Registry):
   keytool -certreq -alias *www.yourdomain.com* -keystore keystore.jks -file cert_req.csr
   Enter password of you keystore when asked.
5) Open **cert_req.csr** file in the text editor and copy all content between and including the following text:
   —–BEGIN NEW CERTIFICATE REQUEST—–

   —–END NEW CERTIFICATE REQUEST—–
6) Go to GoDaddy domain registry website and enter copied content into CRS text area, when requesting certificate:

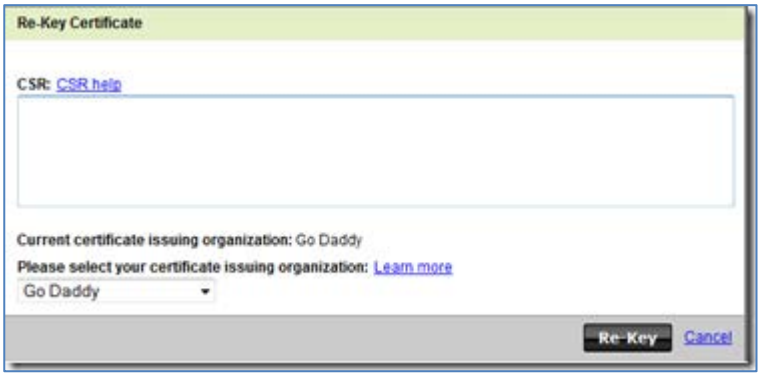


Figure 8 – SSL Certificate Installation

7) After finishing with certificate request you will receive approval email to your address.
8) After approval you need to download a zip file which contains all certificates you need. During this step you will be asked for which server you are downloading certificates. You could select **Other** because Payara is not listed. Your download will contain 4 files:
   a. gd_bundle.crt
   b. gd_cross_intermediate.crt
   c. gd_intermediate.crt
   d. *yourdomain.com*.crt

   First 3 of them are certificates belonging to godaddy.com. They are used to verify your domain's certificate. They may already contained in your **cacerts.jks** but there is no harm importing them in your **keystore.jks**
9) Import these certificates to your **keystore.jsk** using following steps:
   keytool -import -alias root -keystore keystore.jks -trustcacerts -file gd_bundle.crt
   keytool -import -alias cross -keystore keystore.jks -trustcacerts -file gd_cross_intermediate.crt
   keytool -import -alias intermed -keystore keystore.jks -trustcacerts -file gd_intermed.crt
   keytool -import -alias www.yourdomain.com -keystore keystore.jks -trustcacerts -file yourdomain.com.crt

10) Restart Payara
11) Open Payara administration console and go to **Configurations** => **server-config** => **HTTP Service** => **Http Listeners** and select **http-listener-2**
12) On the right side of the page select **SSL** tab on top and enter your certificate alias (used in the previous steps) into the **Certificate NickName** field.
13) If you want your server to use standard SSL port (443), select **General** tab and enter **443** into the Port field.
14) Finally save settings by pressing **Save** button.
15) Restart Payara server and test https connection by going to https://your.domain.com. Check for certificate info in the address bar.

Note – Sometimes the Payara Admin Console gives an error when creating a new http listener and assigning a certificate alias to it. In such case, try the asadmin command line as follows:
asadmin> create-ssl --type http-listener --certname sampleCert http-listener-1

2.5 JasperReports Server

SOLA currently uses JasperReports Server version 6.3.0[2]. (Significant problems have been experienced in attempts to install various 6.4 versions on the same Payara Server instance as SOLA Community Server).

Download jasperreport-server-cp-6.3.0-bin.zip from
http://community.jaspersoft.com/project/jasperreports-server/releases.

1) Extract **jasperreports-server-cp-6.3.0-bin** folder from the archive.
2) In the **.../jasperreports-server-cp-6.3.0-bin/buildomatic/sample_conf** folder locate the **postgresql_master.properties** and copy it to the **buildomatic** folder.
3) Rename the file you copied to **default_master.properties**.
4) Edit the **default_master.properties** file to add the settings for your database and application server. At the top of the file comment out **appServerType = tomcat7** on line 31 by adding a # at the start of the line, then uncomment **# appServerType = glassfish3** on line 39 by removing the #.
5) Comment out the **appServerDir** setting on line 43 and uncomment the **appServerDir** setting on line 58. Edit this setting so that it matches the root folder of your Payara installation (e.g. C:\\payara5 or /etc/paraya5).
6) Scroll down to line 67 and enter the **postgres user password**. Change **dbHost**, **dbUsername** and **dbPort** if they are different from default values.
7) Make sure PostgreSQL is **running**.
8) **Stop** the PayaraSOLA service if it is running.
   $ sc stop PayaraSOLA or use Services panel
9) Open a Command Prompt window and go to **buildomatic** folder.
   $ cd c:\Downloads\jasperreports-server-cp-6.3.0-bin\buildomatic
10) From the command line run :
   $ js-install-ce.bat minimal
11) Check **buildomatic/logs/js-install-ce-<date>.log** file for any errors.
12) Start the PayaraSOLA service to check reports server has been deployed successfully.
   $ sc start PayaraSOLA
13) When you open the Payara Server Admin console sometimes it takes a while to deploy and show JasperReports server in the list of applications. Once it is there click on the **launch** link next to the **jasperserver** application. Alternatively open the following URL - http://localhost:8080/jasperserver[3] (for default Paraya setup) (or http://domainname.org:8080/jasperserver or http://ipaddress:8080/jasperserver).

[2] The SOLA Docker installation successfully uses JasperReports version 7.2 (May 2020)
[3] For JasperReports v7 or later http://localhost:8080

*If the jasperserver app does not start check the **payara5/glassfish/domains/domain1/auto deploy** folder to see if there is a file named **jasperserver.war_deployFailed** file. If this is the case check the jasperserver install log file in the **jasperreports-server-cp-6.3.0-bin/buildomatic/logs** folder and if there are no error messages, restart the workstation/server and check again if jasperserver has auto deployed and you are able to open the jasperserver application.*

14) If the JasperReports server Admin console is successfully opened, login using **jasperadmin** username with password **jasperadmin**.

3. Server Installation on Linux

These instructions are for installing OpenJDK 8 (update 232), Payara Server 5 (Build 194.1), PostgreSQL 12 and PostGIS 3.0 on Ubuntu 18.04.

3.1 Installation Preparation

Ubuntu uses the **apt** tool (Application Package Tool) to install and upgrade applications. **apt** is capable of downloading pre-configured installation packages from various online repositories. The process to do an **apt** installation involves the following Terminal commands:

$ sudo apt update
$ sudo apt install <<application name>>

For example to install the following applications used in SOLA related routines:

$ sudo apt update
$ sudo apt install p7zip-full
$ sudo apt install nautilus-admin
$ sudo apt install git
$ sudo apt install net-tools
$ sudo apt install openjdk-8-jre
$ sudo apt -y install icedtea-netx icedtea-plugin

However, please note that some of the packages required for SOLA are not directly available through these repositories due to licensing restrictions and/or the lack of a suitable install package and the installation of those packages will require different methods.

3.2 OpenJDK Java

OpenJDK is freely available for Linux distributions and can be installed using apt. Due to licensing restrictions (and future user licence charges, the Oracle Java JDK is not recommended. Steps that need to be followed to download and install Java 8[4] are as follows:

1. To remove Oracle JDK (if previously installed) – ascertain where it has been installed. In the following instructions it is assumed that it has been installed in /opt/jdk/[version] (eg [version] = "jdk1.8.0_231"
   $ sudo update-alternatives –-remove "java" "/opt/jdk/[version]/bin/java"
   $ sudo update-alternatives –-remove "javac" "/opt/jdk/[version]/bin/javac"
   $ sudo update-alternatives –-remove "javaws" "/opt/jdk/[version]/bin/javaws"
   $ sudo rm -r /opt/jdk/[version]
2. To install OpenJDK, open a terminal and type
   $ sudo apt update
   $ sudo apt install openjdk-8-jdk
   $ java -version
   The following should be displayed:

```
neil@CS1-STK2M364CC:~$ java -version
openjdk version "1.8.0_232"
OpenJDK Runtime Environment (build 1.8.0_232-8u232-b09-0ubuntu1~18.04.1-b09)
OpenJDK 64-Bit Server VM (build 25.232-b09, mixed mode)
```

3. To make sure OpenJDK is the default JVM run these instructions:
   $ sudo update-alternatives –-config java
   If it is the only version of java you will see the following displayed:
   There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
   Nothing to configure.
   If there is more than one version of java, you will be prompted to specify which version should be used as the default JVM (the Java 8 version)
4. Now to set the JAVA_HOME environment variable, (using a Terminal window), make the following commands:
   $ JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

[4] The use of Java 9, Java 10 or Java 11 is not recommended for SOLA (May 2020)

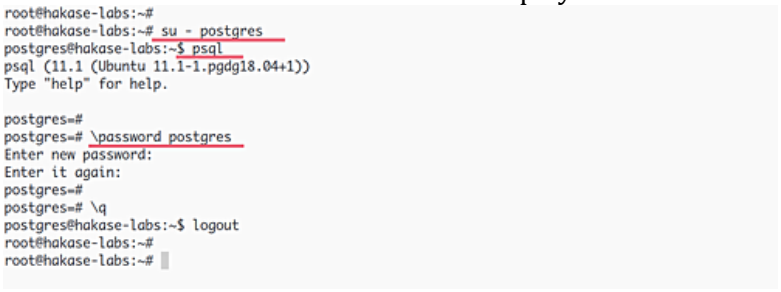
$ export JAVA_HOME
$ JRE_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

5. Check this change has been applied
   $ source /etc/environment
   $ echo $JAVA_HOME
   This should display:
   / usr/lib/jvm/java-8-openjdk-amd64

3.3 PostgreSQL

Download the latest version of PostgreSQL using the apt repository following these steps:

1. Remove any old versions of PostgreSQL and PostGIS
   $ dpkg -l | grep post to list the PostgreSQL and PostGIS packages
   $ sudo apt --purge remove <<each of the PostgreSQL and PostGIS packages>>
2. Identify the optimal combination of different versions of both PostgreSQL and PostGIS by referring to
3. Confirm that for your preferred combination of Ubuntu, PostgreSQL and PostGIS there is the required apt packages by a search of "postgresql-nn1-postgis-nn2-scripts" (where nn1 is the version of PostgreSQL and nn2 is the version of PostGIS) on the https://www.ubuntuupdates.org website to check
   (for instance in May 2020 the optimal combination was Ubuntu 18.04 (bionic), PostgreSQL 12 and PostGIS 3)
3. Import the repository signing key, and update the package lists with this Terminal command
   $ sudo wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add –
   $ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'
   $ sudo apt update
4. Install PostgreSQL 12 use this Terminal command.
   $ sudo apt -y install postgresql-12 postgresql-contrib-12
5. Add a password to the default PostgreSQL user postgres
   $ sudo su – postgres
   $ psql
   # \password postgres
   Enter and re-enter your new password
   Exit from psql and revert to Ubuntu root user
   # \q
   $ logout
   And this screenshot should be what is displayed



6. Configure PostgreSQL to allow access from other computers
   By default PostgreSQL locked down to prevent access from computers other than the localhost. If your database server will be used as the production server, then it may be a good option to leave PostgreSQL in this locked down state. If the database will be used for testing or training, it may be desirable to access the database server from other

computers. To allow these connections it is necessary to make some configuration changes.

Edit the **/etc/postgres/12/main/pg_hba.conf** file and add the following line (the first in the range of IP assigned by your wifi hotspot/router) to this file
host all all 192.168.1.1/24 md5

and check that this line is also present in the pg_hba.conf file

local all all trust

7. Configure PostgreSQL to accept connections from SOLA applications
   This can be achieved from psql with the ALTER SYSTEM command as follows:
   $ sudo su - postgres
   $ psql
   # ALTER SYSTEM SET listen_addresses='*';
   \q
   $ logout
8. Run PostgreSQL as a service
   This command may take some time to complete as it will download and install the PostgreSQL 12 database server. Once the installation is complete, PostgreSQL will be running as a service.

   You can start, stop and restart PostgreSQL, reload the configuration files and list the status of the server using the ***service*** command:

   $ sudo service postgresql *[status, reload, start, restart and stop].*

   Therefore make the following commands:
   $ sudo service postgresql reload
   $ sudo service postgresql restart
   The configuration files for PostgreSQL will be installed in **/etc/postgres/12/main**
   The data directory for PostgreSQL will be **/var/lib/postgres/12/main**

3.3.1 PostGIS

Using the PostgreSQL and PostGIS version numbers identified at the beginning of the PostgreSQL installation (Section 3.3)

$ sudo apt install postgis postgresql-12-postgis-3 postgresql-12-postgis-3-scripts

3.3.2 PostgreSQL extensions

The **uuid-ossp** PostgreSQL extension provides GUID generation functions. It must be installed as an extension into PostgreSQL. The source files for this extension are included in the postgresql-contrib-11 package which is installed as part of the initial PostgreSQL installation. It is also necessary to create a postgis extension in order to complete the PostGIS installation (note that with this extension, the extension name does not need to be enclosed with double quotes (").
To create an extension in the database you must be logged in as **postgres**.

$ sudo su - postgres
$ psql
# CREATE EXTENSION "uuid-ossp";
# CREATE EXTENSION postgis;
# /q
$ exit

3.3.3 PostgreSQL Admin Console (PGadmin4)

1. Install PGadmin4 (uses the same repository key as PostgreSQL)
   $ sudo apt install pgadmin4 pgadmin4-apache2 -y

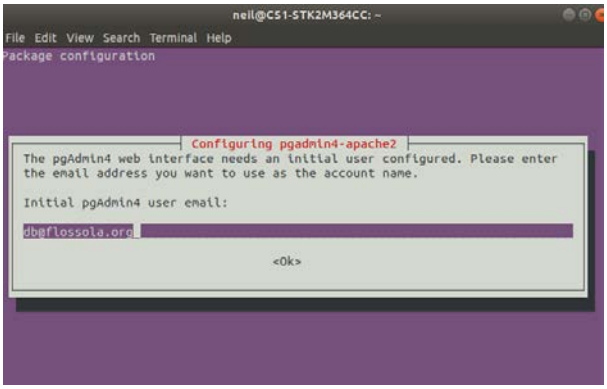


Figure 9 – Enter email address as PGadmin4 user name

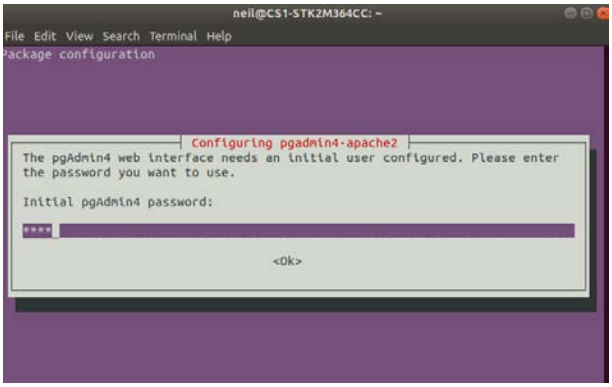
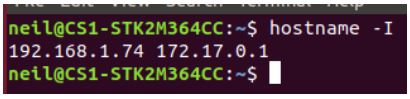


Figure 10 – Enter PGadmin4 password

2. To run PGadmin4, first determine the IP address of the host server with the terminal command:
$ hostname -I



Using the first IP address from the display resulting from the hostname command, type the url **Error! Hyperlink reference not valid.** into your web browser (e.g. http://192.168.1.74/pgadmin4 in the screenshot example above). This same URL address will work for ALL devices connecting to the same LAN (wireless hotspot).

The following web page should display.

Figure 11 – PGadmin4 Logon

If the PGadmin logon page does not display, type in http://localhost/pgadmin4 (or http://domainname.org/pgadmin4 or http://ipaddress/pgadmin4) in a web browser on your server. If that displays OK, check that you have made the necessary changes to

pg_hba.conf described in Section 3.3 remembering to reload and restart (using the $ sudo service postgresql reload (and then restart)

3. If it still does not work you should delete and completely remove all the pgadmin4 packages and reinstall pgadmin4. In December 2019 these packages are **pgadmin4 pgadmin4-apache2 pgadmin4-common pgadmin-doc**

   To confirm what the pgadmin packages are, type the terminal command

   $ dpkg –list pgadmin4* to confirm the 4 packages identified above – if there are different packages, note the package names for the next command

   $ sudo apt -- purge remove pgadmin4 pgadmin4-apache2 pgadmin4-common pgadmin-doc

   $ sudo rm -r /var/lib/pgadmin4

3.3.4 Create SOLA Database

Where there is no configured PostgreSQL database backup and it is the very first time you create a new SOLA database in PostgreSQL, follow these steps:

1) As **root**, restart the Postgresql database to release all connections to the template database
   $ sudo service postgresql restart
2) To create the database you must be logged in as **postgres**.
   $ sudo su - postgres -c "createdb -T postgres sola"
   SOLA database name is usually assumed to be sola
3) Locate the database folder in the Community Server deployment package, copy this folder to the computer hosting PostgreSQL (if not stored there already).
   For those installing in Ubuntu, the following changes are required to create_database.sh
   Line 19 – check and modify to reflect installed version of Postgresql e.g.
   psql="/usr/lib/postgresql/12/bin/psql"
   AND
   Line 24 will probably need to be modified to reflect runtime version 7zip e.g.
   zip_exe="/usr/lib/p7zip/7zr"
   In this sequence of steps the database folder has been copied to ~/Desktop/install/database-master
   $ chmod -R +rwx ~/Desktop/install/database-master
   $ cd ~/database-master
   $ ./create_database.sh

Remember to check the build.log to ensure no problems were encountered.

3.3.5 Alternative approach to SOLA database creation

Refer to Section 2.3.5.

3.4 Payara Server 5

3.4.1 Installation

Payara Server version 5.2020.4 was the current version of Payara in October 2020 and has been used as the application server for SOLA Community Server & SOLA Web Admin.

Payara Server can be installed in two ways. The easiest way to install an appropriately configured Payara Server instance is to copy an already configured instance (remembering to stop Payara Server before making the copy) – refer to Section 2.4.1 Option 1.

Follow these steps if you want to make a fresh Payara Server installation:

1) Download the current Community Edition version of Payara Server from the web site https://www.payara.fish/downloads/payara-platform-community-edition/
   This download is approximately 150Mb
2) Unzip downloaded zip file into the /etc/payara5 folder
   $ sudo unzip payara-5-2020-4.zip -d /etc

$ sudo mkdir /etc/payara5
$ sudo chmod ug+rwx -R /etc/payara5

3) Now provide appropriate world/other rights for the /etc/payara5 folder:
$ sudo chmod –R ug+rwx /etc/payara5
$ sudo chmod –R ug+rwx /etc/payara5/glassfish/domains/domain1
Restart your computer
4) Download the appropriate PostgreSQL jdbc 4.2 jar file from https://jdbc.postgresql.org/download.html (eg postgresql-42.2.16.jar) and copy it into /etc/payara5/glassfish/domains/domain1/lib folder
5) Download the 4 missing jackson JAR files from https://www.dropbox.com/sh/9uf3rbicau2i1uz/AACuFWjjI31ObaHIuAzELchFa?dl=0 and copy them into this folder: /etc/payara5/glassfish/module
6) It is advisable to explicitly set the location of the JDK for Payara Server to use within the Payara setup.
Check location of JDK
$ echo $JAVA_HOME
This should display this if OpenJDK has been installed:
/ usr/lib/jvm/java-8-openjdk-amd64
Edit this file:
/etc/payara5/glassfish/config/asenv.conf
Add a new line at the end of this file
AS_JAVA="<fully qualified path to your JDK> "
eg. AS_JAVA="/usr/lib/jvm/java-8-openjdk-amd64"
OR (if you have used Oracle Java)
AS_JAVA=="/usr/lib/jvm/jdk1.8.0_231"

3.4.2 Run Payara Server as a Service

1) Establish Payara Server as a service
$ sudo touch /etc/systemd/system/payara_sola.service
$ sudo chmod +rw /etc/systemd/system/payara_sola.service
Edit as Administrator the **/etc/systemd/system/payara_sola.service** file and add these lines:

```
[Unit]
Description = Payara Server 5
After = syslog.target network.target
[Service]
ExecStart=/etc/payara5/bin/asadmin start-domain
ExecReload=/etc/payara5/bin/asadmin restart-domain
ExecStop=/etc/payara5/bin/asadmin stop-domain
TimeoutStartSec=0
Type = forking


[Install]

WantedBy = multi-user.target
```

2) Create Payara Server Service and Start on Start-up
$ sudo systemctl daemon-reload
$ sudo systemctl enable payara_sola
To start this service (and similarly stop)
$ sudo systemctl start payara_sola
To check Payara is running
$ sudo systemctl status payara_sola

At this point you will have a Paraya Server instance that is configured to run as a service under the root user account.

3.5 JasperReports Server

SOLA currently uses JasperReports Server version 6.3.0[5]. (Significant problems have been experienced in attempts to install various 6.4 versions on the same Payara Server instance as SOLA Community Server).

Download jasperreport-server-cp-6.3.0-bin.zip from
http://community.jaspersoft.com/project/jasperreports-server/releases

1) In the **.../jasperreports-server-cp-6.3.0-bin/buildomatic/sample_conf** folder locate the **postgresql_master.properties** and copy it to the **buildomatic** folder.
2) Rename the file you copied to **default_master.properties**.
3) Edit the **default_master.properties** file to add the settings for your database and application server. At the top of the file comment out **appServerType = tomcat7** on line 31 by adding a **#** at the start of the line, then uncomment **# appServerType = Payara3** on line 39 by removing the **#**.
4) Comment out the **appServerDir** setting on line 43 and uncomment the **appServerDir** setting on line 58. Edit this setting so that it matches the root folder of your Payara installation (e.g. C:\payara5 or /etc/paraya5).
5) Scroll down to line 67 and enter the **postgres user passwordm**. Change **dbHost**, **dbUsername** and **dbPort** if they are different from default values.
6) Make sure PostgreSQL is **running**.
7) **Stop** the PayaraSOLA service if it is running.
   $ sudo service payara_sola stop
8) Open a Terminal window and go to **buildomatic** folder.
   $ cd ~/Downloads/jasperreports-server-cp-6.3.0-bin/buildomatic
9) From the command line run :
   $ sudo ./js-install-ce.sh minimal
10) Check **buildomatic/logs/js-install-ce-<date>.log** file for any errors.
11) Start the PayaraSOLA service to check reports server has been deployed successfully.
    $ sudo service payara_sola start
12) When you open the Payara Server Admin console sometimes it takes a while to deploy and show JasperReports server in the list of applications. Once it is there click on the **launch** link next to the **jasperserver** application. Alternatively open the following URL - http://localhost:8080/jasperserver for version 6.3.0 or http://localhost:8080 for version 7 or later (for default Paraya setup) or http://domainname.org:8080 or http://ipaddress:8080 etc.

    If the jasperserver app does not start check the **payara5/glassfish/domains/domain1/auto deploy** folder to see if there is a file named **jasperserver.war_deployFailed** file. If this is the case check the jasperserver install log file in the **jasperreports-server-cp-6.3.0-bin/buildomatic/logs** folder and if there are no error messages, restart the workstation/server and check again if jasperserver has auto deployed and you are able to open the jasperserver application.
13) If the JasperReports server Admin console is successfully opened, login using **jasperadmin** username with password **jasperadmin**.

[5] The SOLA Docker installation successfully uses JasperReports version 7.2 (May 2020)

4. Docker installation

Docker is operating system level virtualization software. It delivers software in isolated, self-contained containers that bundle their own software, libraries and configuration files and communicate with each other through well-defined channels[6].

Docker allows an installation of a SOLA Community Server application with all associated software components (**Payara Server**, **PostgreSQL**, **GeoServer**, **JasperReports**, an automated database backup service and database administration console (**pgAdmin**)) to be completed in few minutes.

Docker engine can be freely downloaded and used on Linux type operating systems, Windows and Mac environments. Additionally to Docker, Docker Compose[7] is used, allowing running and managing multiple Docker containers with a single command. Docker Compose is a wrapper over Docker engine, executing Docker commands behind the scene.

It is important to note, that Docker installation comes fully configured and doesn't require further configurations of Payara Server, GeoServer and JasperReports. Once installed, it is ready to go. The only remaining piece that can be additionally configured is Email Service (see Section 9), if email notifications are required in your deployment.

4.1 System requirements

Windows

- Windows 10 64-bit: Pro, Enterprise, Education (Build 16299 or later) or Windows 10 Home (Build 1903 or later);
- Enabled WSL 2 feature for Windows 10 Home or enabled Hyper-V and Containers Windows features for Windows 10 Pro, Enterprise, Education;
- 64 bit processor with Second Level Address Translation (SLAT);
- 4GB system RAM;
- Enabled BIOS-level hardware virtualization support;

Mac

- Mac hardware must be 2010 or a newer model, with Intel's hardware support for memory management unit (MMU) virtualization, including Extended Page Tables (EPT) and Unrestricted Mode. You can check to see if your machine has this support by running the following command in a terminal: `sysctl kern.hv_support` If your Mac supports the Hypervisor framework, the command prints `kern.hv_support: 1`
- macOS must be version 10.14 or newer. That is, Mojave or Catalina. Recommend upgrading to the latest version of macOS. If you experience any issues after upgrading your macOS to version 10.15, you must install the latest version of Docker Desktop to be compatible with this version of macOS;
- At least 4 GB of RAM;
- VirtualBox prior to version 4.3.30 must not be installed as it is not compatible with Docker Desktop;

Ubuntu

- Ubuntu Focal 20.04 (LTS), 64-bit;
- Ubuntu Bionic 18.04 (LTS), 64-bit;

[6] https://en.wikipedia.org/wiki/Docker_(software)
[7] https://docs.docker.com/compose/

- Ubuntu Xenial 16.04 (LTS), 64-bit;

4.2 Installation

Please, refer to detailed installation instructions as per your operating systems:

Operating system	Link
Windows 10 Professional, Enterprise, Education	https://docs.docker.com/docker-for-windows/install/
Windows 10 Home	https://docs.docker.com/docker-for-windows/install-windows-home/
CentOS	https://docs.docker.com/engine/install/centos/
Debian	https://docs.docker.com/engine/install/debian/
Fedora	https://docs.docker.com/engine/install/fedora/
Ubuntu	https://docs.docker.com/engine/install/ubuntu/
Mac	https://docs.docker.com/docker-for-mac/install/

Windows and Mac installations already contain Docker Compose component, whereas for Linux type systems, it should be installed separately. Please, refer to this page - https://docs.docker.com/compose/install/ for further details and complete Docker Compose installation for Linux type OS.

While installing on Windows 10, you may need to enable Hyper-V support. You can check it by clicking **Start** menu and start typing "Turn Windows Features". Once it is found, click on it and make sure **Hyper-V** checkbox is checked, including sub-groups under it.

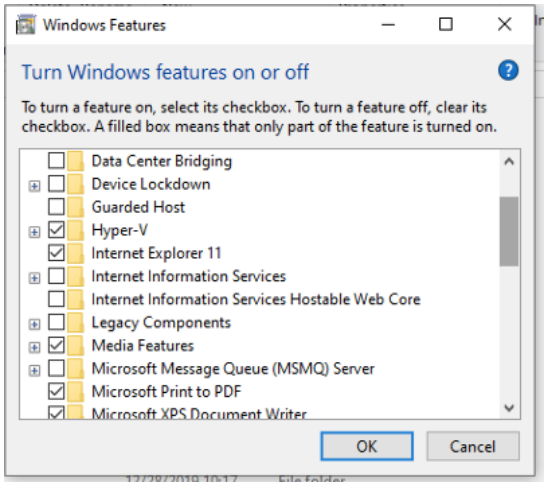


Figure 12 – Enable Windows Hyper-V

It is recommended to use WSL engine over Windows container for Windows 10 installation. Make sure you don't select Windows container option while installing Docker. Otherwise, if it's installed already, run Docker Desktop, go to Settings, select "**Use the WSL 2 based engine**" option and click **Apply and Restart** button.

Additional information about Docker installation can be found at https://docs.docker.com/engine/install/

4.3 Configuring and running SOLA Community Server and associated services

** Please, note that for installation of SOLA Community Server Docker images, you will need to have a reliable Internet connection, capable to download few gigabytes from the Internet.*

Configuring and running SOLA Community Server consists of 3 simple steps:

1) Download required files;
2) Configure parameters;
3) Run containers;

Download required files

You only need 2 small files to configure and run your SOLA Community Server environment. Before downloading these file, create or select a folder where you want to keep your configuration parameters and data folder. By default data folder is created at the same location, where configuration files are, but it can be changed (see below in configuration part). For Windows users, it's recommended to use a folder on C: drive.

For downloading required files, you can use this link - https://tinyurl.com/docker-cs and extract **docker-compose.yaml** and **.env** files into your folder.

Alternatively, you can download them directly from GitHub repository at https://github.com/OpenTenure/docker/tree/master/compose. In this case, you will have to click on the file and then select "Raw" button at the right top of file's header. It will open original file in the browser and you will have to save it to your folder (e.g. CTRL+S). Don't forget to save both files.

Configure parameters

You can adjust various port numbers, database name, password and data storage location by opening and editing **.env** file using any text editor (e.g. Notepad). You don't need to change these parameters if there is no such need. Just observe it to find out how to access SOLA Community Server and any relevant services.

The important part though is checking port numbers to make sure they are not overlapping with existing ones, opened on your server/computer. Users of Linux type operating systems can do it by running the following command:

```
netstat -ntlp | grep LISTEN
```

Windows users can use the following command:

```
netsh int ipv4 show excludedportrange protocol=tcp
```

or if it's not available:

```
netstat -a
```

Knowing your port numbers, check SOLA Community Server configuration parameters, referring to the table below and if changes are required, make them to the **.env** file.

Parameter	Default value	Description
CS_ADMIN_PORT	4849	Payara administration console port number
CS_HTTP_PORT	8989	Community Server port number for accessing it over HTTP protocol (e.g. http://server-name:8989)
CS_HTTPS_PORT	8990	Community Server port number for accessing it over HTTPS protocol (e.g. https://server-name:8990)
DB_PORT	5434	PostgreSQL database port number
DB_ADMIN_PORT	8991	pgAdmin port number for administering PostgreSQL
GEOSERVER_PORT	8985	GeoServer port number
REPORTS_PORT	8996	JasperReports Server port number
TIME_ZONE	UTC	Time zone for containers. Full list can be found at https://en.wikipedia.org/wiki/List_of_tz_database_time_zones (select from TZ database name column)
DB_NAME	sola_cs	Community Server database name
DB_PASSWORD	OpenTenure	Database password
DB_ADMIN_EMAIL	admin@opentenure.org	Email address, used as a user name for accessing pgAdmin. Password as per DB_PASSWORD value
HOST_IP_OR_NAME	localhost	IP addres or host name, used to access GeoServer. It should be hosting server (running this installation) name or its full DNS name or IP address. Don't use localhost if you are planning accessing your server from the network. At least server name is recommended here.
DB_DATA	./data/db	Folder location for storing database data (e.g. databases). "./" is equivalent to the current folder.
DB_ADMIN_DATA	./data/db_admin_backups	Folder location for storing database backups, created by pgAdmin tool (manual)
DB_BACKUPS	./data/db_auto_backups	Folder location for storing automatic (scheduled) database backups

GEOSERVER_DATA	./data/geoserver	Folder location for storing GeoServer data (layers, styles, etc.)

** Please, make sure, that most of the configuration parameters will be used only for creating Docker containers. It means that if you make changes to the parameters after container is up and running, it will not have an effect. In such case, you will have either remove a container and create it again or configure your changes in the services (containers) by using its operating system commands or through the user interface of appropriate services (e.g. changing DB password in PostGIS store configuration on GeoServer can be done though GeoServer admin).*

Run containers

Finally open command line and go to your folder with **docker-compose.yaml** and **.env** files and type the following command:

```
docker-compose up -d
```

This simple command will make all job by pulling required Docker images from Internet, creating containers and runnig them. If there are any issues, you will be able to see the errors in your command line. Otherwise, successful installation will take 2-7 minutes for the first start (after downloading is finished), depending on your computer/server performance.

If you are running on Windows or Mac, you can open Docker Desktop dashboard and check each container by clicking on it and watching logs outputs. Upon successful loading, logs output will stop at some stage showing final messages. For instance **ot-cs** (Community Server) container will show something similar to the following messages:

```
[#|2020-11-27T13:46:03.040+0000|INFO|Payara
5.2020.6|javax.enterprise.system.jmx|_ThreadID=158;_ThreadName=Thread-
22;_TimeMillis=1606484763040;_LevelValue=800;_MessageID=NCLS-JMX-00005;|
JMXStartupService has started JMXConnector on JMXService URL
service:jmx:rmi://0.0.0.0:8686/jndi/rmi://0.0.0.0:8686/jmxrmi|#]
```

All containers inside the group should be in green color. If it's not the case, click on the container and check for issues. You may need to start this specific container again.

From the command line you can use the following commands to check logs and status:

```
docker logs ot-cs
docker ps -a
```

4.4 Installation issues

A common problem experienced is that one of the Docker services will fail to "create" because the specified port on the host system is in use. Please, check "Configure parameters" in the previous section to adjust port numbers.

4.5 Useful Docker Commands

docker-compose stop	Must be run from the folder of the docker-compose.yml file	Stops all SOLA Community Server services
docker-compose start	Must be run from the folder of the docker-compose.yml file	Starts all SOLA Community Server services
docker-compose logs	Must be run from the folder of the docker-compose.yml file	Displays logs from all services

docker-compose config	Must be run from the folder of the docker-compose.yml file	Displays the docker-compose.yml file with environment variables substituted
docker ps -all		Lists all docker containers
docker image ls		Lists all docker images
docker exec -it *container-id* bin/bash	Container-id can be obtained from docker ps -all command. With pgadmin container substitute "sh" for bin/bash	Opens terminal within specified container

4.6 Accessing SOLA Community Servers components in Docker

When the system hosting Docker does not have an associated domain name, you need to use the IP address of the host system or its host name (e.g. http://192.168.0.100 or http://my-server).

The following table shows an example of various services, which can be accessed under default parameters (assuming your IP address is 192.168.0.100 and host name is my-server).

Community Server module	**URL with Ports Exposed**	**Initial login**[8]
SOLA Community Server	http://my-server:8989 http://192.168.0.100:8989 https://my-server:8990 https://192.168.0.100:8990	test/test
SOLA Web Admin	http://my-server:8989/admin http://192.168.0.100:8989/admin https://my-server:8990/admin https://192.168.0.100:8990/admin	test/test
Payara Admin	http://my-server:4849 http://192.168.0.100:4849 https://my-server:4849 https://192.168.0.100:4849	admin/admin
PostgreSQL	my-server:5434 192.168.0.100:5434	postgres/OpenTenure
pgAdmin Console	http://my-server:8991 http://192.168.0.100:8991	admin@opentenure.org/OpenTenure
GeoServer Admin Console	http://my-server:8985 http://192.168.0.100:8985	admin/geoserver

[8] All login details need to be changed and strengthened when Community Server goes into "production"

Community Server module	URL with Ports Exposed	Initial login[8]
Jasper Server Admin Console	http://my-server:8996 http://192.168.0.100:8996	jasperadmin/jasperadmin

4.7 Uninstalling SOLA Community Server Docker installation

This uninstall removes the SOLA Docker installation completely (software and data) except for backups made through the automated database service – ***so use with caution***.

docker-compose down -v --rmi all --remove-orphans	Must be run from the folder of the docker-compose.yml file	Stops and completely removes all SOLA services & associated images
docker system prune docker image prune docker volume prune		Removes all other SOLA artifacts

4.8 Database Backups & Restore

4.8.1 Automated Backups

Currently daily automated database backups are scheduled and these are kept for 7 days before being deleted. Weekly backups are kept for 4 weeks and monthly backups kept for 6 months. These scheduling parameters are configurable using the environment variables in the docker-compose file.

Backup files are stored in the folder, configured in the **DB_BACKUPS** variable of **.env** file.

4.8.2 Manual Backups – using pgAdmin

A manual database backup can also be initiated by users using pgAdmin (which is running in the ot-db-admin docker container).

Select the Community Server database, right mouse click, select **Backup** and provide backup name in the **Filename** field. The resulting backup file will appear in the **admin_opentenure.org** folder, located under the folder, configured in **DB_ADMIN_DATA** variable (e.g. …/data/ db_admin_backups/admin_opentenure.org).

4.8.3 Restoration of SOLA Community Server database

The easiest way of restoring SOLA Community Server databse is using pgAdmin tool, bundled as part of SOLA Community Server Docker installation.

1. Select **Databases** and right click to **Create** a new Database (e.g. sola2). If you are restoring the Community Server database, you will need to **Drop** it first before creating a new one to use for the database restore. If there have been previous Community Server sessions, a message will display that informs you of these other sessions and that you cannot Drop and Delete the old database. To overcome this run these docker-

compose commands in the directory containing the docker-compose.yml file for your implementation.

```
$ cd ~/ docker – or whatever folder contains docker-compose.yml
$ docker-compose stop db
$ docker-compose start db
```

Open a new instance of pgAdmin4 and try to **Drop** and **Create** a new Community Server database again using old name.

2. Select new Database, right click and select **Restore**

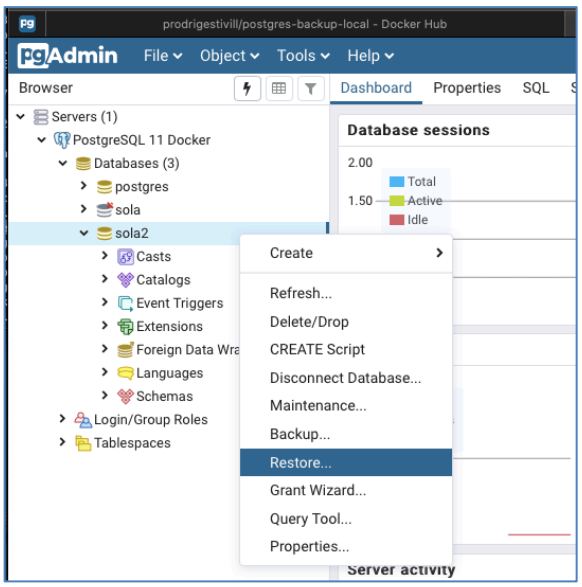


Figure 13 – PGadmin - Restore

3. Click on Filename **....** Button
4. Click on **All Files** & then the toolbar **upload** icon 

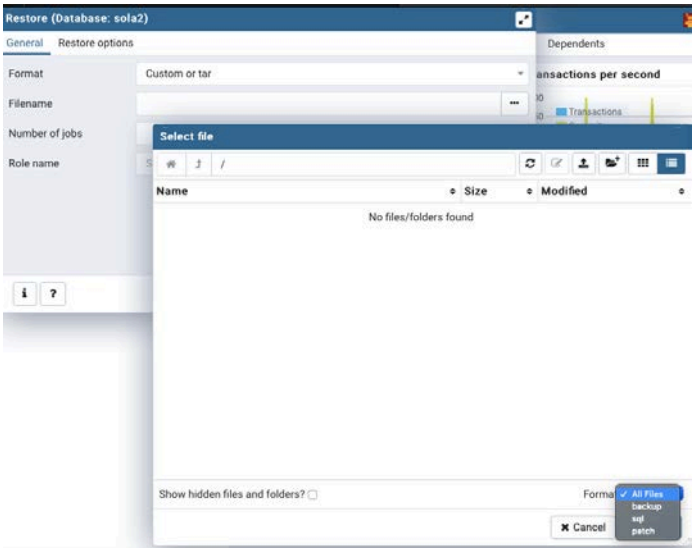


Figure 14 – PGadmin – Restore – Select All files

5. Drag the backup file into pgAdmin4 upload panel (or alternatively copy your backup into **admin_opentenure.org** folder, located under the folder, configured in **DB_ADMIN_DATA**)
6. Click on upload icon and then click on the backup file and the **Select** box

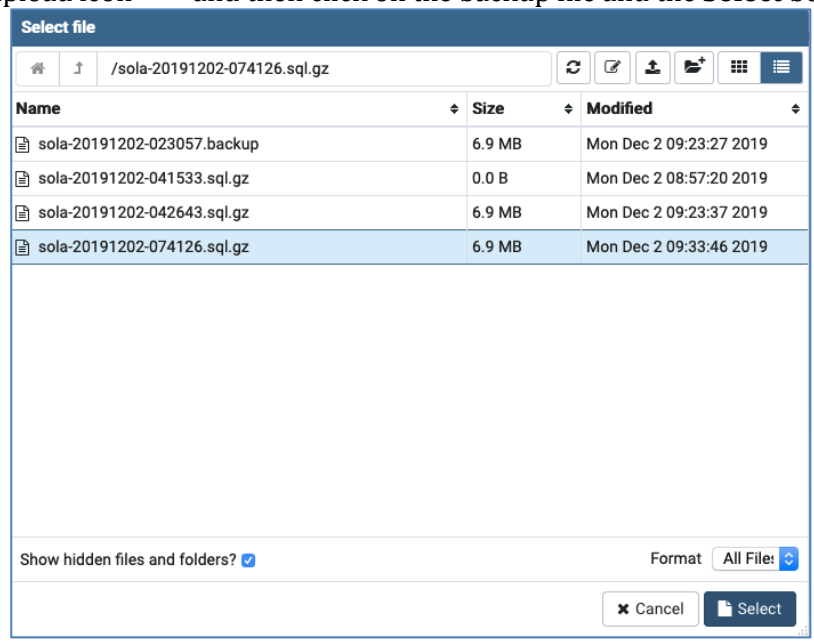


Figure 15 – PGadmin – Restore – Select backup file

7. Click on **Restore** button

4.9 Offline Installation of SOLA Community Server and associated services

In preparation for an offline installation, **while connected to the Internet**:

1. Download docker-compose and .env files from https://minhaskamal.github.io/DownGit/#/home?url=https://github.com/OpenTenure/docker/blob/master/compose
2. Download the Docker Desktop / Engine software for Windows from https://hub.docker.com/editions/community/docker-ce-desktop-windows and save the installation file to a USB stick (to store all files necessary for installing Community Server on the offline host system)
3. Download imagery covering the new community's land ideally as a GeoTif file with srid = 4326 and copy to USB stick
4. On a test Docker installation clear any existing docker images from the system

```
$ docker stop $(docker ps -a q)
$ docker rm $(docker ps -a q)
$ docker image rm $(docker image ls)
```

5. Download the Docker image for the SOLA Community Server software from Docker Hub

```
$ docker pull solovov/ot-cs
$ docker save –output sola-cs-srv.tar
```

6. Download the Docker image for the SOLA PostgreSQL database for Community Server from Docker Hub

```
$ docker pull solovov/ot-db
```

```
$ docker save –output sola-cs-db.tar
```

7. Download the Docker image for the PostgreSQL PGadmin4 software from Docker Hub

```
$ docker pull solovov/ot-dbadmin
$ docker save –output pgadmin4.tar
```

8. Download the Docker image for the automated PostgreSQL database backup software from Docker Hub

```
$ docker pull prodrigestivill/postgres-backup-local
$ docker save –output pgBackup.tar
```

9. Download the Docker image for the Geoserver software from Docker Hub

```
$ docker pull solovov/ot-geoserver
$ docker save –output geoserver.tar
```

10. Download the Docker image for the Jasper Server software from Docker Hub

```
$ docker pull solovov/ot-reports
$ docker save –output jasperserver.tar
```

11. Copy the (6) output files onto the USB stick
12. Clear the test Docker environment of all downloaded Docker artifacts

```
$ docker system prune
$ docker image prune
$ docker volume prune
```

13. Restart (or close down) the Docker software using the Docker icon in the toolbar on your computer
14. Download the Docker Desktop software (appropriate to the future host system operating system) and save the installation file to the USB stick with the output (tar) files.

On the host system not connected to the Internet:

1. Create a docker folder (C:\docker)
2. Unzip docker-compose and .env files into **compose** folder, created under docker folder.
3. Install the Docker Desktop software (as described earlier in Section 4)
4. Check that **Docker** and **docker-compose** are running
   In a Command/Terminal window type:

```
$ docker --version
```

This should display similar output:
Docker version 19.03.9, build 9d988398e7

```
$ docker-compose –version
```

This should display similar output:
docker-compose version 1.25.4, build 8d51620a

5. Copy the output (tar) files into the **Downloads** folder on the host system
6. Navigate to the Downloads folder and then run the following Terminal commands to load the output (tar) files

```
$ cd c:\Downloads
$ docker load sola-cs-db.tar
$ docker load sola-cs-svr.tar
$ docker load pgadmin4.tar
$ docker load pg-backup-local.tar
```

```
$ docker load geoserver.tar
$ docker load jasperserver.tar
$ docker load portainer.tar
$ docker load traefik1.tar
```

7. Complete the Docker installation (as for online installation) from Section 4.

5. Configure Payara Server 5

This section describes the deployment of SOLA Community Server and Geoserver and all configuration changes required in Payara Server in order for SOLA Community Server and all the associated software to be able to run correctly. These instructions primarily apply to a non-Docker installation (as these deployments and configuration changes occur largely automatically in Docker installations)

Before starting, check status of the Payara service and make sure it is running (refer to Section 2.4.2 (Windows) or Section 3.4.2 (Ubuntu)).

5.1 JDBC Connection

Open the Payara Admin Console at **http://localhost:4848** (or **http://domainname.org:4848** or **http://ipaddress:4848**).

1) In the Payara Server Admin Console, locate the **Resources → JDBC → JDBC Connection Pools** node
2) Click the New button
3) In the New **JDBC Connection Pool** (**General** tab)
   - Pool Name = sola
   - Resource Type =javax.sql.ConnectionPoolDataSource
   - Database Driver Vendor select Postgresql
4) Click on the **Next** for first time use or the **Additional Properties** tab and set the following values:
   - DatabaseName=<your database name>
   - Password = <your postgres password>
   - PortNumber = 5432
   - ServerName = localhost
   - User = postgres
5) Click Finish, then select the Connection Pool you just created and click **Ping**. You should get a Ping Succeeded message.
6) It is now necessary to add a new JNDI resource for the connection pool. The JNDI name is the value used by the Java Naming Directory Interface (JNDI) resolution to reference/locate the sola connection pool resource. Go to the **JDBC → JDBC Resources** node and click **New**...
7) Set the **JNDI Name** to jdbc/sola. It is important to use this name as this is the value referenced by the MyBatis connection configuration files.
8) Select **sola** as the **Pool Name** and click OK to create the new JDBC Resource.
9) Click on **OK**

5.2 Security Realm

SOLA delegates authentication of user credentials to a Payara JDBC Security Realm. It is possible to configure a variety of Security Realms in Payara, and any of these could be used for SOLA. The JDBC Security Realm is used because it references the user credentials directly from the sola database which greatly simplifies the administration of user details using SOLA Web Admin.

Open the Payara Admin Console at **http://localhost:4848** (or **http://domainname.org:4848** or **http://ipaddress:4848)**.

1) In the Payara Server Admin Console, navigate to the **Configurations → server-config → Security → Realms** node, select **New**...
2) Set the following values. All other values should remain blank.
   a. Name = SolaRealm
   b. Class Name = com.sun.enterprise.security.auth.realm.jdbc.JDBCRealm
   c. JAAS Context = jdbcRealm
   d. JNDI = jdbc/sola

e. User Table = system.appuser
f. User Name Column = username
g. Password Column = passwd
h. Group Table = system.user_roles
i. Group Name Column = rolename
j. Digest Algorithm = SHA-256

3) Click **OK** to save the new security realm.
4) Click the **Configurations → server-config → Security** node
5) Check the **Default Principal To Role Mapping** checkbox to enable this setting. SOLA does not include any specific role mapping configuration and relies on the default mapping provided by Payara Server (i.e. roles from the security realm have a one to one mapping to roles in the application based on role name).
6) **Save** this change

5.3 JVM Settings

This section customizes a minimal set of JVM settings for Community Server.

Open the Payara Admin Console at **http://localhost:4848** (or **http://domainname.org:4848** or **http://ipaddress:4848)**.

1) In the Payara Server Admin Console, navigate to the **Configurations → server-config → JVM Settings** node
2) Click the **JVM Options** tab
3) Change -client to -server. This will ensure the Server JVM is used to run the application domain rather than the client JVM.
4) Change the -Xmx512m option to **-Xmx1280m**. This will allow the JVM to use up to 1.25GB of RAM.
5) Select and **Delete** the JVM setting **+UseOpenJSSE**
6) Click the **Add JVM Option** button and enter **-Xms640m** in the blank Value field.
7) To recognize non-default Geoserver Server data directory (outside of payara5 folder)
   Windows Only specify a folder for Geoserver data (so that you can utilize pre-configured layers required by Community Server).
   Click the **Add JVM Option** button and enter:
   -DGEOSERVER_DATA_DIR=C:\geoserver_data in the blank Value field
   Ubuntu Only: Specify a folder for Geoserver data (so that you can utilize pre-configured layers required by Community Server).
   Click the **Add JVM Option** button and enter:
   -DGEOSERVER_DATA_DIR=/var/lib/geoserver_data in the blank Value field
8) JVM Option for JasperReports Server The latest Payara Server versions are bundled with Jersey libraries of 2.x version, whereas JasperReports Server is using Jersey 1.x. As a result sometimes after a Payara Server restart, a wrong version of Jersey library might be picked up, which will result in JasperReports admin failing to deploy and all Payara Server deployed applications running very slow or failing altogether. Therefore, before installing JasperReports Server, add a JVM
   Click the **Add JVM Option** button and enter:
   -Dcom.sun.enterprise.overrideablejavaxpackages=javax.ws.rs,javax.ws.rs.core,javax.ws.rs.ext
   in the blank Value field
9) **Save** your changes. You should get a message indicating the save was successful. Changes to the JVM settings will also require a **restart** of Payara Server, however you should complete all of the configurations below before restarting.
10) As a check on your JVM settings you can make the following command to list all JVM settings for domain1 (the assumed Community Server Payara domain)
    Ubuntu and Mac
    $ cd /etc/payara5/bin
    $ ./asadmin list-jvm-options

Windows

cd c:\payara5\bin

asadmin list-jvm-options

The following displays this listing of all JVM settings (default and specific settings for Community Server are in **bold**)

-
Djava.ext.dirs=${com.sun.aas.javaRoot}/lib/ext${path.separator}${com.sun.aas.javaRoot}/jre/lib/ext${path.separator}${com.sun.aas.instanceRoot}/lib/ext
-Djdk.corba.allowOutputStreamSubclass=true
-Djavax.management.builder.initial=com.sun.enterprise.v3.admin.AppServerMBeanServerBuilder
-Djdk.tls.rejectClientInitiatedRenegotiation=true
-Dosgi.shell.telnet.maxconn=1
-Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.config.serverbeans.AppserverConfigEnvironmentFactory
-Djava.endorsed.dirs=${com.sun.aas.installRoot}/modules/endorsed${path.separator}${com.sun.aas.installRoot}/lib/endorsed
-DGEOSERVER_DATA_DIR=/var/lib/sola-docker-data/geoserver_data
-Dgosh.args=--nointeractive
-
Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.shell,org.apache.felix.gogo.runtime,org.apache.felix.gogo.shell,org.apache.felix.gogo.command,org.apache.felix.shell.remote,org.apache.felix.fileinstall
-Xms640m
-Djava.security.auth.login.config=${com.sun.aas.instanceRoot}/config/login.conf
-Dfelix.fileinstall.disableConfigSave=false
-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy
-Dfelix.fileinstall.bundles.new.start=true
-Dorg.glassfish.grizzly.DEFAULT_MEMORY_MANAGER=org.glassfish.grizzly.memory.HeapMemoryManager
-Djavax.xml.accessExternalSchema=all
-Dosgi.shell.telnet.port=6666
-Dfelix.fileinstall.log.level=2
-Xbootclasspath/p:${com.sun.aas.installRoot}/lib/grizzly-npn-bootstrap-1.7.jar --> JDK versions: min(1.8.0.121), max(1.8.0.160) (Inactive on this JDK)
-Djava.awt.headless=true
-Dfelix.fileinstall.poll=5000
-DGEOSERVER_DATA_DIR=/var/lib/geoserver_data
-Dcom.sun.enterprise.overrideablejavaxpackages=javax.ws.rs,javax.ws.rs.core,javax.ws.rs.ext
-Djdbc.drivers=org.apache.derby.jdbc.ClientDriver
-XX:+UnlockDiagnosticVMOptions
-Dfelix.fileinstall.dir=${com.sun.aas.installRoot}/modules/autostart/
-Xbootclasspath/p:${com.sun.aas.installRoot}/lib/grizzly-npn-bootstrap-1.6.jar --> JDK versions: min(1.8.0), max(1.8.0.120) (Inactive on this JDK)
-Xbootclasspath/p:${com.sun.aas.installRoot}/lib/grizzly-npn-bootstrap-1.8.jar --> JDK versions: min(1.8.0.161), max(1.8.0.190) (Inactive on this JDK)
-Xbootclasspath/p:${com.sun.aas.installRoot}/lib/grizzly-npn-bootstrap-1.8.1.jar --> JDK versions: min(1.8.0.191), max(1.8.0.500)
-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks
-DANTLR_USE_DIRECT_CLASS_LOADING=true
-Dfelix.fileinstall.bundles.startTransient=true
-Dorg.glassfish.grizzly.nio.DefaultSelectorHandler.force-selector-spin-detection=true
-Dcom.ctc.wstx.returnNullForDefaultNamespace=true
-server
-Dosgi.shell.telnet.ip=127.0.0.1
-Dcom.sun.enterprise.security.httpsOutboundKeyAlias=s1as
-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks
-Dorg.jboss.weld.serialization.beanIdentifierIndexOptimization=false
-XX:NewRatio=2
-Xmx1280m

5.4 Logger Settings

For development purposes it is useful to capture the SQL queries that are sent from Payara Server to the database in the Payara Log. The Payara Log is displayed in the Output View by Netbeans making it very convenient to follow the SQL the application sends to the database as the application is running. Note that these settings are not required in production unless it is necessary to capture the database queries for debugging purposes.

Open the Payara Admin Console at **http://localhost:4848** (or **http://domainname.org:4848** or **http://ipaddress:4848)**.

1) In the Payara Server Admin Console, navigate to the **Configurations → server-config → Logger Settings** node
2) Click the **Log Levels** tab (for first time use, you may need to **Add** each of these logger settings)
3) Confirm there is a logger name **java.sql** and set the Log Level to **FINE**.
4) Confirm there is a logger called **java.sql.Connection** with the Log Level set to **FINE**. Note that the java.sql and java.sql.Connection loggers must both be set to FINE to log SQL statements.
5) Confirm there is a logger called **org.sola.services** with the Log Level set to **INFO**. This is the logger used by the LogUtility in SOLA. Exception messages and other useful details are recorded in the Payara Log through this logger.

6) Confirm there is a logger called **java.sql.ResultSet** with the Log Level set to **OFF**. This logger will capture all results returned from the database however logging every result has a significant performance impact. This logger should only be turned to the FINE level for debugging specific issues and then turned OFF again.
7) **Restart** Payara Server

5.5 Deploy SOLA applications

Installation and database creation file for the SOLA applications (Community Server & Web Admin) are available from these Github repositories:

- https://github.com/OpenTenure/latest-release

Open the Payara Admin Console at **http://localhost:4848** (or if not accessing on the host system **domainname.org:4848** or **ipaddress:4848)**
To deploy the relevant applications:

SOLA Services ear file Installation

1) In the Payara Server Admin Console, navigate to to **Applications** node.
2) Click the **Deploy** button. On the Deploy Applications or Modules page, click the Choose File button, browse to the SOLA deployment package and select the **sola-cs-services-ear.ear** SOLA services ear file and click **Open**.
3) On the Deploy Applications or Modules page click the **OK** button. Deployment will take 1 to 2 minutes. **Warning** - Do not navigate away from the Deploy Applications or Modules page during this time otherwise the deployment will be aborted.

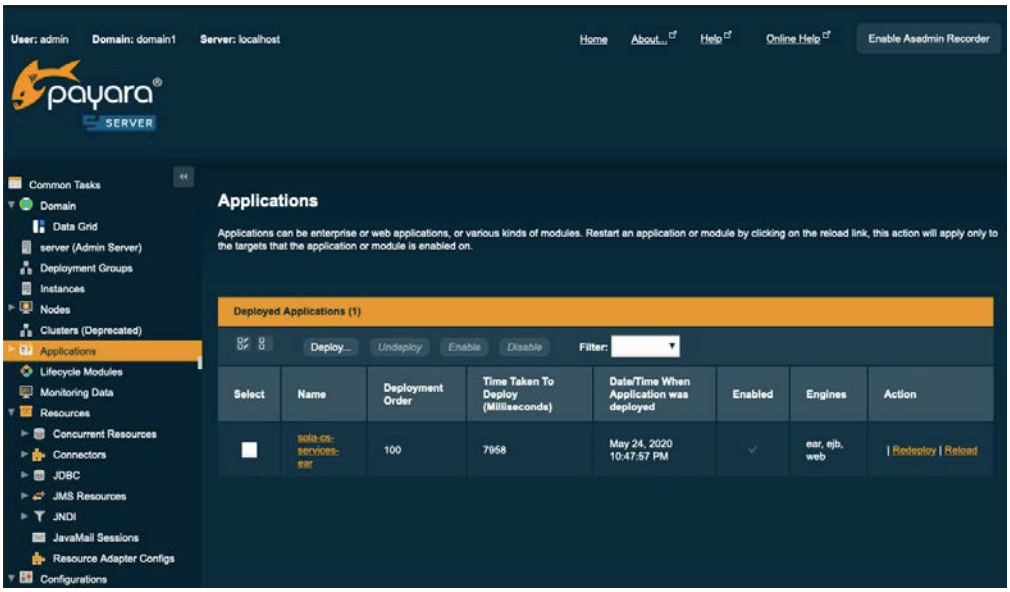


Figure 16 – Payara Admin Console with SOLA Community Server application deployed

5.6 Deploy GeoServer

Geoserver is an essential component of SOLA Community Server. Geoserver can be installed beside the SOLA Community Server on the same Payara domain (domain1). For more complicated implementation scenarios, you should consider a separate Payara domain for Geoserver.

Please note that Community Server uses Geoserver version 2.18.0 which is downloadable from http://geoserver.org/download - use the Web Archive (WAR) package.

5.6.1 Deploy Geoserver in Payara Server

Extract the geoserver.war file from the downloaded geoserver zip file.

Open the Payara Admin Console at **http://localhost:4848** (or if not accessing on the host system **domainname.org:4848** or **ipaddress:4848**

1) Navigate to the Applications node and click the Deploy button
2) Click Browse and navigate to the geoserver.war file, select and click Open.
3) Click OK to deploy the WAR. Geoserver will process the file and after 1 to 2 minutes you will be returned to the Applications page indicating that the geoserver application is deployed. **Note**: do not navigate away from the Payara processing page otherwise the deployment will be aborted.

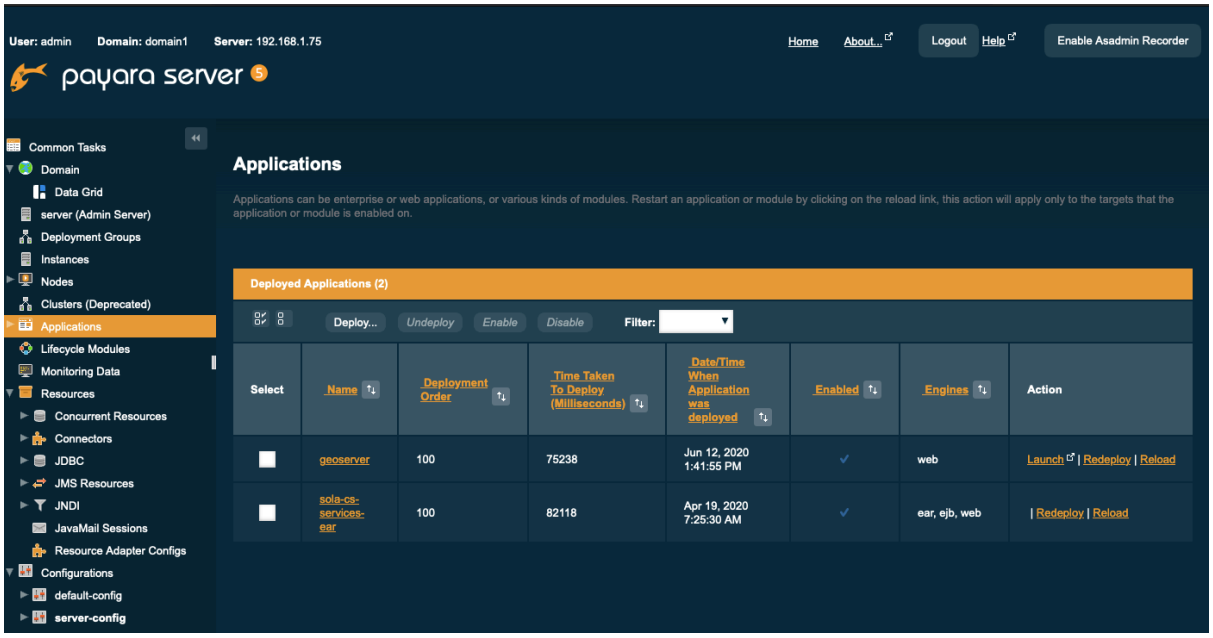


Figure 17 – Payara Admin Console with Geoserver deployed

5.6.2 Configuring Claims and Boundaries Layers

This sections describes Claims and Boundaries layers configuration and required if you didn't use pre-configured GeoServer layers, described in section Error! Reference source not found..

5.6.2.1 Creating SLD Styles

Before publishing the claims and boundaries layers, you have to create SLD styles for colouring parcel shapes based on claim status as well as boundaries polygons.

Claims style:

1) Start the GeoServer admin console at http://localhost:8085/geoserver/web or http://domainname.org:8085/geoserver/web or http://ipaddress:8085/geoserver/web (docker installation) or http://localhost:8080/geoserver/web or http://domainname.org:8080/geoserver/web or http://ipaddress:8080/geoserver/web (non-docker installation)
2) Logon to GeoServer Admin console (default account credentials are username: **admin**, password: **geoserver**).
3) Go to **Workspaces** in the left side panel and click the **Add new workspace** link.
4) Name the workspace as **opentenure** and assign it a url of http://localhost:8080/geoserver. You can also select it as a default namespace.
5) Save the namespace.
6) Go to **Styles** in the left side panel and click **Add a new style.**

7) Enter **claims** in the Name field and select **opentenure** workspace. Copy and paste the following text into the style text area and click the **Submit** button at the bottom of the page.

Note – if you are viewing this document in a PDF viewer, you can use this link - https://www.dropbox.com/s/8i3usbheuh6oxdm/ClaimPolygonStyle.txt?dl=0 to download claims style in the text format for more convenient use.

```
<?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>claim_polygon</Name>
  <UserStyle>
  <!-- Styles can have names, titles and abstracts -->
   <Title>Claim Polygon</Title>
   <Abstract>Claim style that draws a polygon</Abstract>
   <FeatureTypeStyle>
    <Rule>
     <Name>rule1</Name>
     <Title>Unmoderated claims</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>mapped_geometry</ogc:PropertyName>
       </ogc:Function>
      </Geometry>
      <Label>
       #<ogc:PropertyName>nr</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>Claims for review</Title>
   <ogc:Filter>
  <ogc:And>
   <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>status_code</ogc:PropertyName>
    <ogc:Literal>unmoderated</ogc:Literal>
   </ogc:PropertyIsEqualTo>
   <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>expired</ogc:PropertyName>
    <ogc:Literal>true</ogc:Literal>
   </ogc:PropertyIsEqualTo>
  </ogc:And>
   </ogc:Filter>
  <PolygonSymbolizer>
  <Fill>
   <CssParameter name="fill">#FFFFFF</CssParameter>
   <CssParameter name="fill-opacity">0</CssParameter>
  </Fill>
  <Stroke>
      <CssParameter name="stroke">#FF9500</CssParameter>
  </Stroke>
  </PolygonSymbolizer>
 </Rule>
    <Rule>
     <Name>rule4</Name>
     <Title>Reviewed claims</Title>
   <ogc:Filter>
   <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>status_code</ogc:PropertyName>
    <ogc:Literal>reviewed</ogc:Literal>
   </ogc:PropertyIsEqualTo>
   </ogc:Filter>
     <PolygonSymbolizer>
      <Fill>
       <CssParameter name="fill">#FFFFFF</CssParameter>
       <CssParameter name="fill-opacity">0</CssParameter>
      </Fill>
   <Stroke>
    <CssParameter name="stroke">#73FF00</CssParameter>
   </Stroke>
     </PolygonSymbolizer>
    </Rule>
  <Rule>
  <Name>rule3</Name>
  <Title>Moderated claims</Title>
   <ogc:Filter>
   <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>status_code</ogc:PropertyName>
    <ogc:Literal>moderated</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>
    </Stroke>
      </PolygonSymbolizer>
     </Rule>
    </FeatureTypeStyle>
   </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

Boundaries style:

1) While you are in the admin console of GeoServer go to **Styles** in the left side panel and click **Add a new style.**
2) Enter **boundaries** in the Name field and select **opentenure** workspace. Copy and paste the following text into the style text area and click the **Submit** button at the bottom of the page.

Note – if you are viewing this document in a PDF viewer, you can use this link - https://www.dropbox.com/s/yc0jr2efbbrcroh/BoundariesStyle.txt?dl=0 to download boundaries style in the text format for more convenient use.

```
<?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>boundary_polygon</Name>
     <UserStyle>
       <!-- Styles can have names, titles and abstracts -->
       <Title>Boundary Polygon</Title>
       <Abstract>Boundary style that draws a polygon</Abstract>
       <FeatureTypeStyle>
         <Rule>
           <Name>rule1</Name>
           <Title>Districts</Title>
           <ogc:Filter>
             <ogc:PropertyIsEqualTo>
               <ogc:PropertyName>type_code</ogc:PropertyName>
               <ogc:Literal>district</ogc:Literal>
             </ogc:PropertyIsEqualTo>
           </ogc:Filter>
           <TextSymbolizer>
             <Geometry>
               <ogc:Function name="centroid">
                 <ogc:PropertyName>geom</ogc:PropertyName>
```

```
          </ogc:Function>
        </Geometry>
        <Label>
          <ogc:PropertyName>name</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>
      <PolygonSymbolizer>
        <Fill>
          <CssParameter name="fill">#FFFFFF</CssParameter>
          <CssParameter name="fill-opacity">0</CssParameter>
        </Fill>
        <Stroke>
          <CssParameter name="stroke">#BA4A04</CssParameter>
        </Stroke>
      </PolygonSymbolizer>
    </Rule>
    <Rule>
      <Name>rule2</Name>
      <Title>Villages</Title>
      <ogc:Filter>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>type_code</ogc:PropertyName>
          <ogc:Literal>village</ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:Filter>
      <TextSymbolizer>
        <Geometry>
          <ogc:Function name="centroid">
            <ogc:PropertyName>geom</ogc:PropertyName>
          </ogc:Function>
        </Geometry>
        <Label>
          <ogc:PropertyName>name</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>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#FFFFFF</CssParameter>
              <CssParameter name="fill-opacity">0</CssParameter>
            </Fill>
            <Stroke>
              <CssParameter name="stroke">#8DBA04</CssParameter>
            </Stroke>
          </PolygonSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

5.6.2.2 Creating Claims Layer

1) While you are in the admin console of GeoServer, go to **Stores** and add a new store.
2) Click on **PostGIS** in the **Vector Data Source** list.
3) Select **opentenure** workspace and enter **opentenure** as a **Data Source Name**. Make sure checkbox **Enabled** is ticked.
4) In the **Connection Parameters** provide the following SOLA database settings as per your configuration
    a. **host** – database IP address or host name.
    b. **port** – database port number (default, 5432).
    c. **database** – database name (default, sola).
    d. **user** – database user to make a connection (default, postgres).
    e. **password** – database user password.
5) Click **Save** to save the new store.

Onse the strore is created, you have to create a new layer.

1) Go to **Layers** and click **Add a new layer**.
2) In the **Add layer from** list select **opentenure:opentenure**.
3) On the next page click **Configure new SQL view…**
4) Enter **claims** value in the **View Name** field and provide the following **SQL statement**:

```
select id, nr, (challenge_expiry_date < now()) as expired, mapped_geometry,
status_code
from opentenure.claim
where status_code in ('unmoderated','moderated','reviewed') and
challenged_claim_id is null
```

5) Click **Save** button.

6) On the next page enter **claims** in the **Name** field and **Claims** into **Title** field. Make sure **Enabled** and **Advertised** checkboxes are selected.
7) Enter SRS name into **Declared SRS** field (e.g. EPSG:4326).
8) Provide values in the **Bounding Boxes** section. You can click **Compute from data** and **Compute from native bounds** to calculate bounding box from the data source. If you have no data in the database, these bounding boxes will be empty and you can update them at a later stage.
9) Set **opentenure:claims** as a **Default Style** on the **Publishing** tab.
10) Finally click **Save** button to save claims layer.
11) Check layer is working by going to **Layer Preview** and clicking **OpenLayers** link next to the claims layer. If you have loaded sample data, you should see few parcels on the map. Otherwise, if sola database is empty, the map will be displayed empty as well.

5.6.2.3 Creating Boundaries Layer

Creating boundaries layer is similar to creating claims layer. At this time you don't need to create a new store, but reuse the one created in the previous section.

1) While you are in the admin console of GeoServer, go to **Layers** and click **Add a new layer**.
2) In the **Add layer from** list select **opentenure:opentenure**.
3) On the next page click **Configure new SQL view…**
4) Enter **boundaries** value in the **View Name** field and provide the following **SQL statement**:

   select id, name, type_code, status_code, geom from
   opentenure.administrative_boundary
   where status_code in ('approved')

5) Click **Save** button.
6) On the next page enter **boundaries** in the **Name** field and **Boundaries** into the **Title** field. Make sure **Enabled** and **Advertised** checkboxes are selected.
7) Enter SRS name into **Declared SRS** field (e.g. EPSG:4326).
8) Provide values in the **Bounding Boxes** section. You can click **Compute from data** and **Compute from native bounds** to calculate bounding box from the data source. If you have no data in the database, these bounding boxes will be empty and you can update them at a later stage.
9) Set **opentenure:boundaries** as a **Default Style** on the **Publishing** tab.
10) Finally click **Save** button to save boundaries layer.
11) Check layer is working by going to **Layer Preview** and clicking **OpenLayers** link next to the boundaries layer. If sola database is empty, the map will be displayed empty as well.

5.6.3 Adjusting databse connection settings for pre-configured layers

If you are using pre-configured Claims and boundaries layers, described in section Error! Reference source not found., you need to adjust database connection settings, by doing the following steps:

1. Start the GeoServer admin console at http://localhost:8085/geoserver/web or http://domainname.org:8085/geoserver/web or http://ipaddress:8085/geoserver/web (docker installation) or http://localhost:8080/geoserver/web or http://domainname.org:8080/geoserver/web or http://ipaddress:8080/geoserver/web (non-docker installation)

2. Logon to GeoServer Admin console (default account credentials are username: **admin**, password: **geoserver**).
3. Click on **Stores** in the left side panel and then click on the **opentenure** store name (don't confuse with **opentenure** Workspace name).
4. Go through the settings and adjust the following ones as per your database configuration:
    a. **host** – database IP address or host name.
    b. **port** – database port number (default, 5432).
    c. **database** – database name (default, sola).
    d. **user** – database user to make a connection (default, postgres).
    e. **password** – database user password.
5. Tick "**Expose primary keys**" checkbox.
6. Select empty value in the "**SSL mode**" dropdown list.
7. Save changes.

Once changes are saved successfully, verify that **claims** and/or **boundaries** layers can be opened. You can check it from **Layer Preview** menu in the left side panel. In the list of layers, click "**OpenLayers**" link for one of the layers. If your databse is empty, you will see an empty map, otherwise existing clmais or boundaries should be visible.

In the case of any errors, they will be displayed either in the browser or proposed for save/download. Review the error and make appropriate corrections as per error message text.

5.6.4 Publishing GeoServer Map Layers

SOLA applications uses Geoserver to publish the imagery layers through a WMS service which can then be used within the SOLA applications. For instance, Geoserver can be used to host satellite / orthophoto imagery[9] relevant to a specific implementation of SOLA Community Server. It is also used to create the **Claims** and **Community Areas** SQL views using the SOLA database where the geometry of those features is stored. The symbology of such SQL views is defined using a SLD (xml) style definition which has been predefined and is loaded, along with the SQL definition of the view in Geoserver from the geoserver_data sub-directory.

SOLA Web Admin (https://localhost:8181/admin or if not accessing on the host system **https://domainname.org:8181/admin** or **https://ipaddress:8181/admin**) is used to incorporate the WMS service layers (published by Geoserver) into the SOLA Community Server application[10].

[9] Note that the possibilities of using Geoserver (data sources that GeoServer can access) are more than what has been described. The standard GeoServer installation supports the loading and serving of the following data formats (list of possible raster layers you can configure in GeoServer):

- GeoTIFF
- GTOPO30
- WorldImage
- ImagePyramid
- ImageMosaic
- Other data sources are supplied as GeoServer extensions. Extensions are downloadable modules that add functionality to GeoServer. Extensions are available at the GeoServer download page.

[10] As at June 2020, with Docker implementations, after any logon involving the ":8181" port, there is an error where the returned url contains the docker container ID rather than "localhost" etc. The work around is to edit the url in the browser to remove the container ID and replace it with the "localhost" etc

5.6.5 Publish Satellite Imagery

The following steps assume the satellite (or orthophoto or drone) imagery has been supplied in GeoTIFF format and that the supplied file is less than 2 Gigabytes[11]

8. Start the Geoserver admin console at http://localhost:8085/geoserver/web or http://domainname.org:8085/geoserver/web or http://ipaddress:8085/geoserver/web (docker installation) or http://localhost:8080/geoserver/web or http://domainname.org:8080/geoserver/web or http://ipaddress:8080/geoserver/web (non-docker installation)
9. Logon to GeoServer Admin console (default account credentials are username: **admin**, password: **geoserver**).
10. Go to **Workspace** in the left side panel
    - Click on **Add new workspace** link
    - In **Name** field add **opentenure**
    - In **Namespace URI** add **http://localhost/geoserver**
    - Ensure **Default workspace** is ticked
11. Go to **Stores** in the left side panel.
    - Click **Add new store** link.
    - Under category Raster data sources click **GeoTIFF** (if the imagery file is a geoTIFF file).
      Note: Geoserver can utilise the Image Mozaic plug-in from the open source Geotools library. This plug-in allows for an image pyramid format Geoserver Data Store to be created from a GeoTIFF file and this will enhance performance particularly when the imagery is displayed through the Open Tenure software on mobile devices. Where there is a image pyramid structure select ImageMosaic (rather than WorldImage)
    - For **Workspace** choose **opentenure**,
    - For **Data Source Name** enter community or locality name covered by the imagery.
    - For **URL** choose file:coverages button and use the Browse link to find the geoTIFF file (or the mosaic file folder).
    - Click **Save.** The screen New Layer will appear.
12. Add Layer After saving the Store the screen New Layer will show up.
    - Click **Publish**.
    - For **Name** enter <<community or locality name>>.
    - Make sure both **Enabled** and **Advertised** check boxes are ticked
    - For **Title** enter the same community or locality name.
    - In **Declared SRS** click **Find**. Search for **4326** which is the SRID of WGS84 (geographic coordinates) datum.
13. Make sure **SRS handling** has the value "**Force Native to Declared**".
14. Click on **Compute** from data link and then **Compute from native bounds** link
15. Click **Save**.
16. Test the Layer by navigating to **Layer Preview** in the left side panel.
    - Find in the list of layers the imagery layer name.
    - Under column Common Formats click on **OpenLayers**.
    - You will be brought to a new web/tab page and if everything is fine you should see the image.

[11] If the imagery GeoTIFF file is greater than 2 Gbytes, then it should be converted into a Mosaic file structure that is recognized by Geoserver. There are Geoserver plug-ins that perform this conversion

5.6.6 Other Useful Map Features

Although not essential, sometimes it is very useful in the map windows of SOLA Community Server to have some additional map features to help users to "locate" themselves when they are viewing imagery displayed in both software applications. Typically these map features include roads, rivers, village names and administrative boundaries but this will vary from place to place and their inclusion will depend on availability.

In the following descriptions of how to incorporate these other map features into SOLA Community Server, it is assumed that the map features are available in ESRI Shapefile format and that implementing these as wms layers through Geoserver (rather than loading them into the sola PostgreSQL database and customizing the SOLA Community Server software) is the best approach.

OpenStreetMap Data

Often a good source for these map features is from the OpenStreetMap which allows its data to be copied, distributed, transmited and adapted, as long as you credit OpenStreetMap and its contributors. A suggested credit is: "© OpenStreetMap contributors".

If you use this data you must also make it clear that the data is available under the Open Database License, and if using their map tiles, that the cartography is licensed as CC BY-SA. You may do this by linking to the OpenStreetMap copyright page.

Often on the internet you can find a series of OpenStreetMap data extracts for a country nicely organized by data themes that largely coincide with the suggested map feature types. Alternatively, you can use a GIS (such as the open source QGIS with the Quick OSM Plugin) to extract the suggested map feature types yourself.

Publishing Shapefile Map Layers in GeoServer

For each map feature shapefile follow these steps in the Geoserver Admin Console (http://localhost:8085/geoserver):

1. For each map feature to be included in Open Tenure and Community Server, go to **Stores** and add new store. Click on **Shapefile** in the **Vector Data Source** list.
2. Select **opentenure** workspace and enter ***<<mapFeature1>>*** as a **Data Source Name** and for URL choose file:data button and use the Browse link to find the Shapefile file. Click Save. The screen New Layer will appear.
3. Add Layer After saving the Store the screen New Layer will show up.
    - Click Publish.
    - For Name enter <<***mapFeature1***>>.
    - Make sure both Enabled and Advertised check boxes are ticked
    - For Title enter the same community or locality name.
    - In Declared SRS click Find. Search for 4326 which is the SRID of WGS84 (geographic coordinates) datum.
    - Add Layer After saving the Store
4. Make sure SRS handling has the value "Force declared".
5. Click on Compute from data link and then Compute from native bounds link
6. Go to the Publishing tab and select an appropriate value from the Default Style values
7. Click Save.
8. Test in the Layer Preview in the left side panel

Once a Geoserver layer has been created for each shapefile, create a new Layer Group incorporating each of these new layers as well as the imagery to be used with Open Tenure:

1. Click on Layer Groups (under Data in left hand panel), and on Add new layer grou[

2. Enter **Name** (eg ImageryWithRefLayers) and **Title** (eg. Imagery with Reference Layers) leave **Workspace** blank.
3. Under Layers, click on **Add Layer** and add the imagery layer
4. Above the Layers, click on the Find Button and enter the appropriate SRID/EPSG code for the coordinate system used by the SOLA Community Server map control. Then click on the returned value and check that the desired SRID/EPSG code is displayed.
5. Click on the Generate Bounds button so that the 4 Bounds fields are filled.
6. Continue and **Add Layer** for each of the Shapefile layers.
7. Click on Save
8. Test in the Layer Preview in the left side panel.

5.7 Confirming all SOLA Community Server components are running

In a non-Docker installation, if you have sequentially followed the instructions sequentially in all the preceeding sections (excluding the "Docker" descriptions in Sections **Error! Reference source not found.** and **Error! Reference source not found.**) you should be able to access all the components through your web browser with the URLs described below:

Community Server module	**URL**	**Initial login**[12]
SOLA Community Server	http://sola.org:8080 http://ipaddress:8080	test/test
SOLA Web Admin	https://sola.org:8181/admin http://ipaddress:8181/admin	test/test
PostgreSQL	Only accessible through PGadmin or terminal commands	postgres/sola
PG Admin Console	http://sola.org/pgadmin http://ipaddress/pgadmin	db@flossola.org/sola
Payara Server 5 Admin Console	http://sola.org:4848 http://ipaddress:4848	admin/admin
Geoserver Admin Console	http://sola.org:8080/geoserver/web http://ipaddress:8080/geoserver/web	admin/geoserver
Jasper Server Admin Console	http://sola.org:8080/jasperserver http://ipaddress:8080/jasperserver	jasperadmin/ jasperadmin

Figure 18 – URLs for Community Server & components (non-Docker)

[12] All login details need to be changed and strengthened when Community Server goes into "production"

6. Initial SOLA Community Server configuration

6.1 General

Most of these initial configurations are made using SOLA Web Admin (however, Geoserver, PGadmin and Jasper Reports admin console may also be needed in some situations.

The settings that can be made with Web Admin can be categorised according to when they are most likely to be changed:

- Initial Community Server setup (this Section 6)
- As part of Email configuration (Section 9),
- As part of Report configuration (Section 10)
- As part of Dynamic Tab configuration (Section 11)
- After Community Server has been operational for a while and there is a need to fine-tune the operation of the Community Server (Sections 6 - 11).

Within SOLA Web Admin on the **System settings** form, the Description field for each setting gives a good indication of its purpose and the default values give an indication of the format required. Like the **Reference** data settings, each setting has an **Active status** that is set depending on whether this functionality is relevant to the community Community Server is being established for.

To run SOLA Web Admin application open the following link in your web browser https://localhost:8181/admin or https://hostIPaddress:8181/admin.

By default SOLA Web Admin has the login credentials user **test** and password **test**.

6.2 System Settings

Accessed through the **Settings-System settings** menu option
To change any system setting:

1. Click on the **Pencil** icon and edit accordingly
2. Click on the **Save** button where an edit is made
3. Finally select **Cache → Cache reset** menu item, to clear the old cached value.

The settings that need to be confirmed or changed in the initial setup of Community Server for a new community setup are shaded and made bold in the following table.

Setting Name	Initial Value	Description
account-activation-timeout	70	Account activation timeout in hours. After this time, activation should expire.
boundary-print-country-name		Country name for adding at the end of the boundary location description
boundary-print-crs-description	Unit: degree Geodetic CRS: WGS 84 Datum: World Geodetic System 1984 Ellipsoid: WGS 84 Prime meridian: Greenwich	Description of Coordinate Reference System, which will be listed in the legend area.
boundary-print-produced-by	Community Server of OpenTenure solution	Name of report producer.
cs_server_url	**http://server:8080**	**This setting is used for Docker or similar environments where Community Server sometimes cannot be located by the Reports Server for generating parcel map, used in Claims certificate. The URL should be without ending "/", e.g. http://server:8080. If this setting is empty, Community Server address will be defined automatically.**
claim_certificate_report_url	**/Certificate/certificate/ClaimCertificate**	**URL to the claim certificate report (template), hosted on the reporting server**
community-name	**Open Community**	**Community name**
db-utilities-folder	/usr/lib/postgresql/12/bin	Full path to PostgreSQL utilities (bin) folder (e.g. C:\Program Files\PostgreSQL\12\bin). Used for backup/restore implementation of SOLA Web admin application
docs_for_issuing_cert	signed_cert	List of document type codes, required to set certificate issued status

Setting Name	Initial Value	Description
email-admin-address		Email address of server administrator. If empty, no notifications will be sent
email-admin-name		Name of server administrator
email-body-format	html	Message body format. text - for simple text format, html - for html format
email-enable-email-service	0	Enables or disables email service. 1 - enable, 0 - disable
email-mailer-jndi-name	mail/sola	Configured mailer service JNDI name
email-msg-claim-approve-moderation-body	Dear #{userFirstName}, Claim ##{claimNumber} has been approved. You are receiving this notification as the #{partyRole} Regards, SOLA OpenTenure Team	Claim moderation approval notice body
email-msg-claim-approve-moderation-subject	SOLA OpenTenure - claim moderation approval	Claim moderation approval notice subject
email-msg-claim-approve-review-body	Dear #{userFirstName}, Claim ##{claimNumber} has passed review stage with success. You are receiving this notification as the #{partyRole} Regards,<br /**>SOLA OpenTenure Team**	Claim review approval notice body
email-msg-claim-approve-review-subject	SOLA OpenTenure - claim review approval	Claim review approval notice subject
email-msg-claim-challenge-approve-moderation-body	Dear #{userFirstName}, Claim challenge ##{challengeNumber} has been moderated. You are receiving this notification as the #{partyRole} Regards,<br /**>SOLA OpenTenure Team**	Claim challenge moderation approval notice body
email-msg-claim-challenge-approve-moderation-subj	**SOLA OpenTenure** - claim challenge moderation	Claim challenge moderation approval notice subject
email-msg-claim-challenge-approve-review-body	Dear #{userFirstName}, Claim challenge ##{challengeNumber} has passed review stage. You are receiving this notification as the #{partyRole} Regards,<br /**>SOLA OpenTenure Team**	Claim challenge review approval notice body
email-msg-claim-challenge-approve-review-subject	**SOLA OpenTenure** - claim challenge review	Claim challenge review approval notice subject
email-msg-claim-challenge-reject-body	Dear #{userFirstName}, Claim challenge ##{challengeNumber} has been rejected with the following reason: "#{challengeRejectionReason}" Claim challenge comments: #{challengeComments} You are receiving this notification as the #{partyRole} Regards,<br /**>SOLA OpenTenure Team**	Claim challenge rejection notice body
email-msg-claim-challenge-reject-subject	**SOLA OpenTenure** - claim challenge rejection	Claim challenge rejection notice subject
email-msg-claim-challenge-submitted-body	Dear #{userFullName}, New claim challenge ##{challengeNumber} has been submitted to challenge the claim ##{claimNumber}. You are receiving this notification as the #{partyRole} Regards,<br /**>SOLA OpenTenure Team**	New claim challenge body text
email-msg-claim-challenge-submitted-subject	**SOLA OpenTenure** - new claim challenge to the claim ##{claimNumber}	New claim challenge subject text
email-msg-claim-challenge-updated-body	Dear #{userFullName}, Claim challenge ##{challengeNumber} has been updated. Follow this link to check updated information. You are receiving this notification as the #{partyRole} Regards,<br /**>SOLA OpenTenure Team**	Claim challenge update body text
email-msg-claim-challenge-updated-subject	**SOLA OpenTenure** - claim challenge ##{challengeNumber} update	Claim challenge update subject text

Setting Name	Initial Value	Description
email-msg-claim-challenge-withdraw-body	Dear #{userFirstName}, Claim challenge ##{challengeNumber} has been withdrawn by community recorder. You are receiving this notification as the #{partyRole} Regards, SOLA OpenTenure Team	Claim challenge withdrawal notice body
email-msg-claim-challenge-withdraw-subject	**SOLA OpenTenure** - claim challenge withdrawal	Claim withdrawal notice subject
email-msg-claim-reject-body	Dear #{userFirstName}, Claim ##{claimNumber} has been rejected with the following reason: "#{claimRejectionReason}" The following comments were recorded on the claim: #{claimComments} You are receiving this notification as the #{partyRole} Regards, SOLA OpenTenure Team	Claim rejection notice body
email-msg-claim-reject-subject	**SOLA OpenTenure** - claim rejection	Claim rejection notice subject
email-msg-claim-submit-body	Dear #{userFullName}, New claim ##{claimNumber} has been submitted. You can follow its status by this address. You are receiving this notification as the #{partyRole} Regards, SOLA OpenTenure Team	New claim body text
email-msg-claim-submit-subject	**SOLA OpenTenure** - new claim submitted	New claim subject text
email-msg-claim-updated-body	Dear #{userFullName}, Claim ##{claimNumber} has been updated. Follow this link to check claim status and updated information. Regards, SOLA OpenTenure Team	Claim update body text
email-msg-claim-updated-subject	**SOLA OpenTenure** - claim ##{claimNumber} update	Claim update subject text
email-msg-claim-withdraw-body	Dear #{userFirstName}, Claim ##{claimNumber} has been withdrawn by community recorder. You are receiving this notification as the #{partyRole} Regards, SOLA OpenTenure Team	Claim withdrawal notice body
email-msg-claim-withdraw-subject	**SOLA OpenTenure** - claim withdrawal	Claim withdrawal notice subject
email-msg-failed-send-body	Message send to the user #{userName} has been failed to deliver after number of attempts with the following error: #{error}	Message text for delivery failure
email-msg-failed-send-subject	Delivery failure	Subject text for delivery failure of message
email-msg-notifiable-subject	SOLA REGISTRY - #{actionToNotify} action on property #{baUnitName}	Action on Interest subject text
email-msg-notifiable-submit-body	Dear #{notifiablePartyName}, this is to inform you that one #{actionToNotify} action has been requested by #{targetPartyName} on the following property: #{baUnitName}. Regards, #{sendingOffice}	Action on Interest body text
email-msg-pswd-restore-body	Dear #{userFullName}, You have requested to restore the password. If you didn't ask for this action, just ignore this message. Otherwise, follow this link to reset your password. Regards, SOLA OpenTenure Team	Message text for password restore
email-msg-pswd-restore-subject	**SOLA OpenTenure** - password restore	Password restore subject
email-msg-reg-body	Dear #{userFullName}, You have registered on SOLA OpenTenure Web-site. Before you can use your account, it will be reviewed and approved by Community Technologist. Upon account approval, you will receive notification message. Your user name is #{userName} Regards, SOLA OpenTenure Team	Message text for new user registration on OpenTenure Web-site. Sent to user.
email-msg-reg-subject	**SOLA OpenTenure** - enrolment	Subject text for new user enrolment on OpenTenure Web-site. Sent to user.
email-msg-user-activation-body	Dear #{userFullName}, Your account has been activated. Please use #{userName} to login. Regards, SOLA OpenTenure Team	Message text to notify Community member account activation on the Community Server Web-site

Setting Name	Initial Value	Description
email-msg-user-activation-subject	SOLA OpenTenure account activation	Subject text to notify Community member account activation on the Community Server Web-site
email-msg-user-registration-body	New user "#{userName}" has been registered registered on **SOLA OpenTenure** Web-site.	Message text for new user registration on OpenTenure Web-site
email-msg-user-registration-subject	New user enrolment	Subject text for new user registration on OpenTenure Web-site. Sent to administrator.
email-send-attempts1	2	Number of attempts to send email with first interval
email-send-attempts2	2	Number of attempts to send email with second interval
email-send-attempts3	1	Number of attempts to send email with third interval
email-send-interval1	1	Time interval in minutes for the first attempt to send email message.
email-send-interval2	120	Time interval in minutes for the second attempt to send email message.
email-send-interval3	1440	Time interval in minutes for the third attempt to send email message.
email-service-interval	10	Time interval in seconds for email service to check and process scheduled messages.
enable-reports	**1**	**Indicates whether reports are enbled or disabled. 1 - enabled, 0 - disabled**
max-file-size	10000	Maximum file size in KB for uploading.
max-uploading-daily-limit	100000	Maximum size of files uploaded daily.
moderation_date	**2019-10-01**	**Closing date of public display for the claims. Date must be set in the format "yyyy-mm-dd". If date is not set or in the past, "moderation-days" setting will be used for calculating closing date.**
moderation-days	**1**	**Duration of moderation time in days**
offline-mode	**0**	**Indicates whether Community Server is connected to the Internet or not. 0 - connected, 1 - not connected**
ot-community-area	POLYGON((170.5165512337329 -45.860430202361904,170.5230529083897 -45.860021252324394,170.52946875234488 -45.86035548577669,170.52865336080504 -45.868230061485164,170.51637957235656 -45.86800594287391,170.5165512337329 -45.860430202361904))	Open Tenure community area where parcels can be claimed – usually modified using the Community Server **Settings-Open Tenure-Community area** menu option
ot-title-plan-crs-proj4	+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs	Custom Coordinate Reference System in Proj4 format, used for generating map image for claim certificate or community areas in OpenTenure. It should match to ot-title-plan-crs-wkt setting. If not provided, WGS84 will be used as default.
ot-title-plan-crs-wkt	GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]	Custom Coordinate Reference System in WKT format of the map image, generated for claim certificate in OpenTenure
product-code	scs	SOLA product code. sr - SOLA Registry, ssr - SOLA Systematic Registration, ssl - SOLA State Land, scs - SOLA Community Server – **DO NOT CHANGE**
product-name	SOLA Community Server	SOLA product name
pword-expiry-days	90	The number of days a users password remains valid
report_server_pass	jasperadmin	Reporting server user password.
report_server_url	**http://test.opentenure.org:8090**	**Reporting server URL.**
report_server_user	jasperadmin	Reporting server user name. -**DO NOT CHANGE**
reports_folder_url	**/Reports/community_server**	**Folder URL on the reporting server containing reports to display on the menu.**
requires_spatial	0	Indicates whether spatial representation of the parcel is required (mandatory). If values is 0, spatial part can be omitted, otherwise validation will request it.
system-id	**TEST**	**A unique number that identifies the installed SOLA system. This unique number is used in the br that generate unique identifiers.**

Figure 19 – Community Server System Settings

6.3 Community Area (extent)

This refers functionality refers to the definition of the **extent** of the area of interest to the community (and not the various (non-claim) community areas the community wishes to map)

Accessed through the **Settings → Open Tenure → Community area** menu option

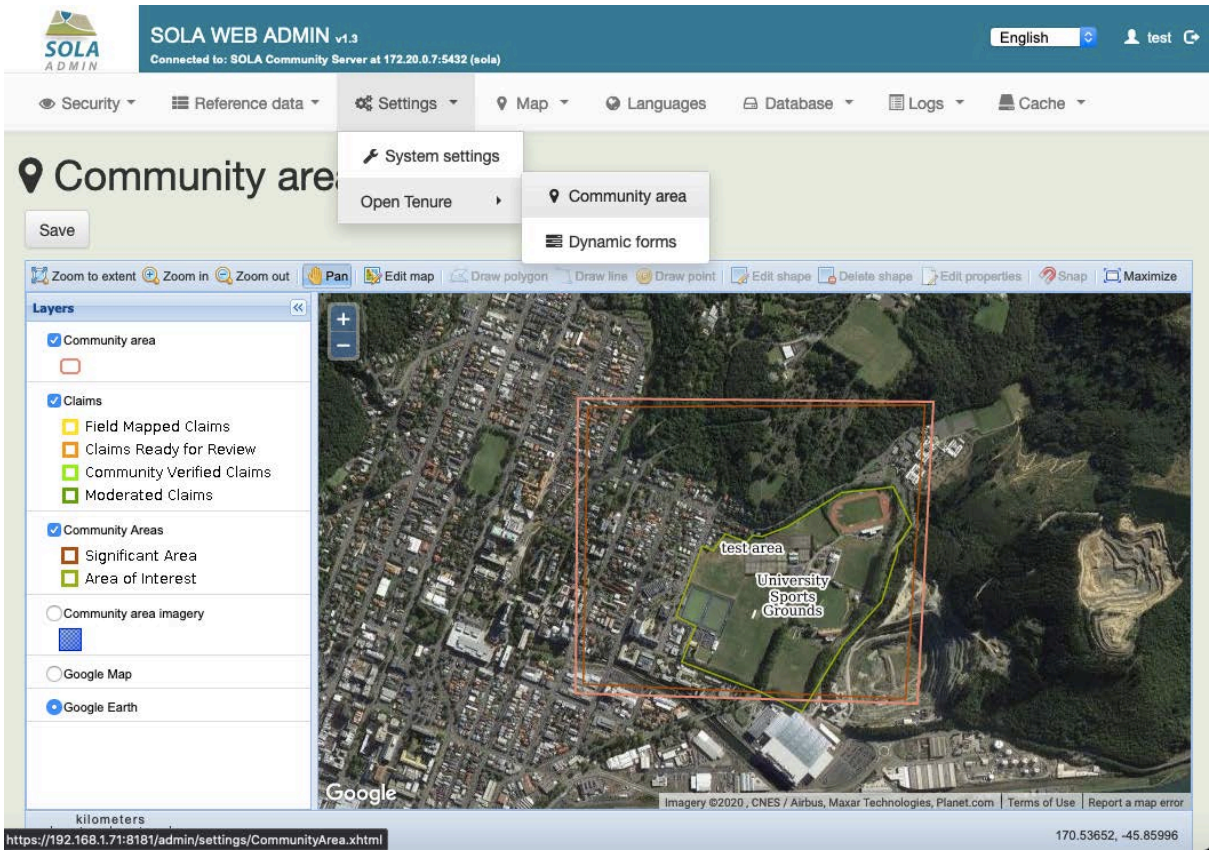


Figure 20 – Modify Community Area (extent of area of interest to community)

To change the community area:

1. **Select** the **Community area** layer from the map legend on the left of the screen
2. **Select** the **Edit map** toolbar icon
3. **Select** the **Delete shape** toolbar icon
4. Click the current Community area in the Map window and agree to delete it
5. With Google Earth or Google Map active Zoom and Pan to the new Community area in the Map window
6. **Select** the **Draw Polygon** toolbar icon
7. **Left click each corner** of the new Community Area and **on the last corner double click**
8. To edit the new Community Area, **Select** on the **Edit shape** and then click inside the Community area. Drag any of the nodes (or mid nodes for new nodes) to their new positions
9. **Click** on the **Save** button
10. Finally select **Cache > Cache reset** menu item, to clear the old cached value.

6.4 Map Layers

Accessed through the **Map→Layers** menu option

If Community Server has been installed through the Docker reverse proxy approach or you may be accessing Community Server through computers other than directly on the host system, then

all the WMS URLs need to be changed to use the IP address of the host system and, where reverse proxy approach has been used, then the Port number needs to be dropped
For instance the WMS URL changes from
http://localhost:8085/geoserver/opentenure/wms **to http://*<<host-system-IP>>/geoserver/opentenure/wms***
For **each** layer:

1. Click on **pencil** icon for each layer
2. In the WMS tab and modify the WMS URL field according to the description above
3. Click on **Save** button
2. Finally select **Cache → Cache reset** menu item, to clear the old cached value.

6.5 Add New Satellite Imagery

To run Geoserver open the following link in your web browser on the host system http://localhost:8085/geoserver/web or https://domainname/geoserver/ or https://hostIPaddress/geoserver/

By default Geoserver Admin Console has the login credentials user **admin** and password **geoserver**.

6.5.1 General

The claim and administrative boundaries layers are defined when SOLA Community Server is first installed. These two layers should not need to be changed except possibly to modify the wms url. An initial imagery layer comes as part of the installation too but that will need to be replaced with appropriate imagery covering the new community's area of interest once the installation has been. The replacement imagery will depend on the availability of appropriate geospatial datasets such as orthophoto or satellite imagery. Any such new layers need to be delivered as a wms service using Geoserver and this web service included in SOLA Community Server as a layer through a SOLA Web Admin definition.

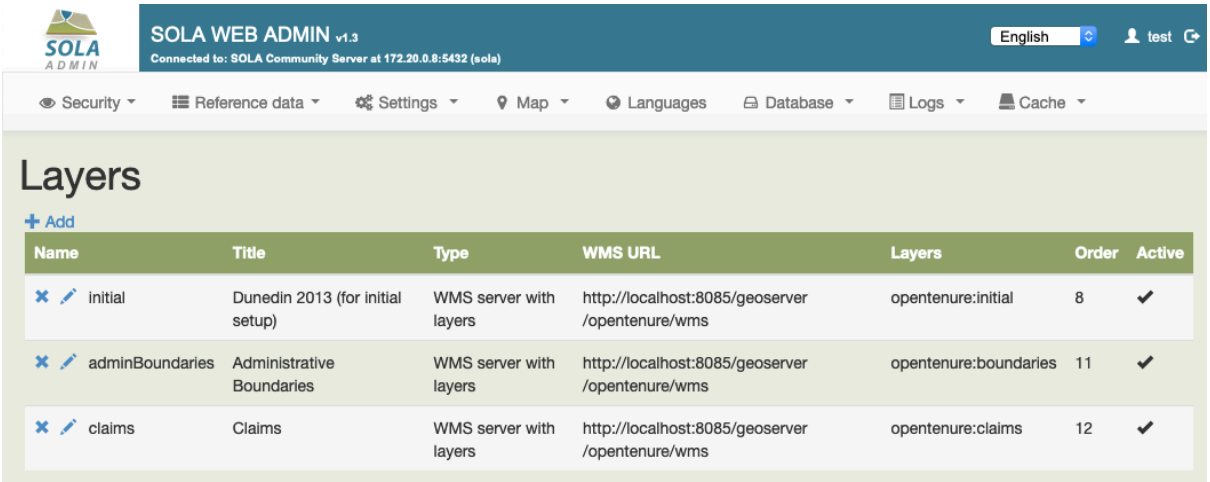


Figure 21 – Original Map Layer Definitions

6.5.2 In Geoserver replace the initial imagery (provided in the docker github repository)

Although other formats can be accommodated by Geoserver, the following steps assume the satellite (or orthophoto or drone) imagery has been supplied in GeoTIF format and ideally with a srid (map projection) of 4326. It is also assumed the supplied file is less than 2 Gigabytes[13].

1. Make a new folder **docker-master/docker/geoserver_data/imagery/community** and copy the new GeoTIF file into this folder
2. Logon to the Geoserver Admin console and go to Stores in the left side panel.
    - Click Add **New** Store.

[13] If the imagery GeoTIFF file is greater than 2 Gbytes, then it should be converted into a Mosaic file structure that is recognized by Geoserver. There are Geoserver plug-ins that perform this conversion

- Under category Raster data sources click **GeoTIFF** (if the imagery file is a geoTIFF file).
- For **Workspace** enter **opentenure**,
- For **Data Source Name** enter "community".
- For **URL** click the **file:coverages** button and use the Browse link to find the geoTIF file in the **../geoserver_data/imagery/community** folder
- Click **Save.** The screen **New Layer** will appear.

3. In the New Layer screen:
   - Click **Publish**.
   - For **Name** enter "community".
   - Make sure both **Enabled** and **Advertised** check boxes are ticked
   - For **Title** enter "Community area imagery".
   - In **Declared SRS** click **Find**. Search for **4326** which is the SRID of WGS84 (geographic coordinates) map projection.
   - If the srid value of the new imagery is **NOT** 4326 the **SRS handling** field has the value "**Force Native to Declared**".
   - If the srid value of new imagery is 4326 the **SRS handling** field has the value "**Keep Native**".
4. Click on **Compute** from data link and then **Compute from native bounds** link
5. Click **Save**.
6. Test the Layer by navigating to **Layer Preview** in the left side panel
   - Find "**Community area** imagery" in the list of layers
   - Under column Common Formats click on **OpenLayers**.
   - The community area imagery should display in a new web/tab page
7. Logout of Geoserver
8. Log into SOLA Web Admin (https://localhost:8181/admin) and select the **Map→Layers** menu option
   - Edit the **initial** layer by clicking on **Pencil** icon
   - In the **General** tab, change the **Title** field to something like "Community ABC Area imagery (2020)"
   - In the **WMS** tab change the **Layers** field to **opentenure:community**
   - Finally select **Cache > Cache reset** menu option, to clear the old cached value.

6.5.3 Definitions for the Claims and Community Areas map layers

In SOLA Web Admin, (through the **Settings→Open Tenure→Community area** menu option) check that the map legend options are suitable for this community implementation.

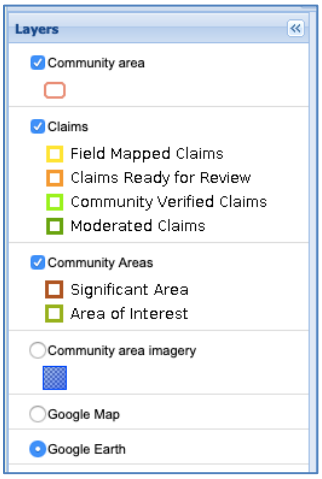


Figure 22 – Community Server (default) Map Legend

It may be necessary to change the symbology and the map legend labelling associated with different categories of Claims (and Community Areas – formerly known as "Boundaries"). The xml definition of these Claims (and Community Areas) layer details can be accessed in **Geoserver** through **Styles** (left hand panel) associated with each of these layers. The map legend definitions for the Community Areas layer, are the most likely **Styles** change. The definition of new types of Community Areas (see Section 8.6) will result in the need for **Styles** changes so that the new types of Community Areas are reflected in the map legend.

6.6 Language Selection

SOLA Community Server has been localized to work in a number of languages. As part of a new installation and configuration, the (localized) languages that are going to be used in this installation need to be identified as "**active**" and for one of those "active" languages to be defined as the **default** language.

To complete this language configuration:

In SOLA Web Admin

1. Click on the **Languages** menu option review

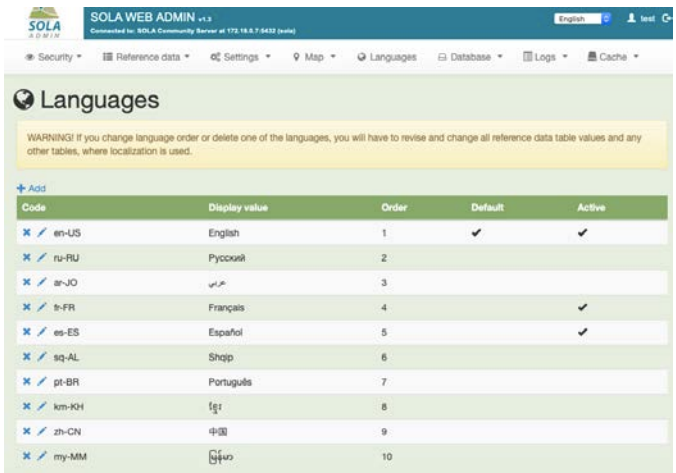


Figure 23 – Web Admin – Languages

2. Click on the Pencil icon for each language that is to be made active, made inactive or made the default language
3. Make the appropriate edits of the applicable checkbox items

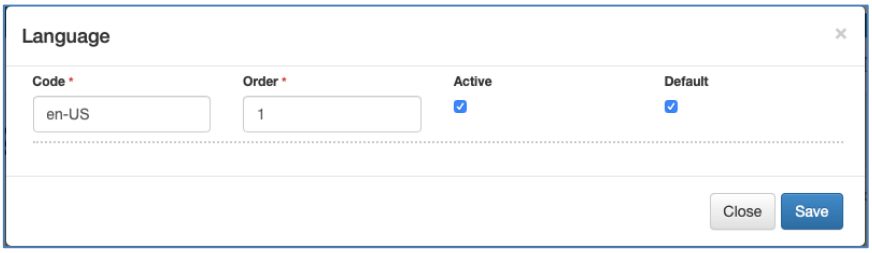


Figure 24 – Web Admin – Language Details

4. Click on **Save** button

6.7 Establish Mail Service

6.7.1 Enable Mail Service in Web Admin Settings

If the mail service is to be used (and you have configured the mail service as described in Section **9**) you must enable it with these steps:

1) Select on the main menu **Settings → System** settings.
2) Look for email-enable-email-service item and click the pencil icon.
3) Make sure **Active** checkbox is ticked and enter **1** as a value.
4) Click **Save** button.
5) Locate **email-admin-address** item and edit it. Provide system administrator's email address.
6) Locate **email-admin-name** and edit it. Provide system administrator's full name.

6.8 Web Admin Database Backup

You can also setup database backup settings. These values will be used if you want to do adhoc database backup and restore from SOLA Web Admin *but will only work if you have used a non-Docker installation.*

Adhoc database backups and restores can also be done using PGadmin or with a Docker command.

1) Select on the main menu Settings → System settings.
2) Look for **db-utilities-folder** item and click edit button on it (pencil icon).
3) Provide full path to the PostgreSQL bin folder (e.g. C:\Program Files\PostgreSQL\12\bin [Windows] or for Ubuntu /etc/postgresql/12/bin) and click **Save** button.

Please note that for Docker implementations, the database backup through SOLA Web Admin will not work.

In all implementations (including Docker implementations), the PGadmin application provides database backup and restore functionality.

7. User Management

Accessed through the **Security→Users** menu option

7.1 New Open Tenure & Community Server User

The first step you should take to configure the Community Server is to add a new admin user account and disable the test account.
In SOLA Web Admin

1) Select **Security → Users** from the main menu. You will see 5 user accounts, but only one account (test) will be active.
2) Click **+ Add** and add a new user account for yourself. On the **General tab,** remember to click **Active** to activate the user and then the select the **Admin** security Group to allow the user account to administer the SOLA Community Server.

   You can select other Security Groups (such as **Community Recorder, Reviewer, Moderator** or **Community** Member) for the user account depending on this user's role in the Community Server team for this community.

3) Click **Save** to save the new user account and verify the new user account is shown on the Users page and that it has a tick in the Active column.
4) On the Users page, you should do one of the following to effectively disable the test user account;
   a. Edit the test user account by clicking the pencil icon and deselecting the Active checkbox, or
   b. Edit the test user account and change the test user password, or
   c. Delete the test account completely.

   If you disable the test user account and forget to active your own account, you can edit the system.appuser table in the database and manually set active to true.

7.2 Roles

A key aspect of security is controlling which functions users are able to execute. All SOLA applications, including the SOLA applications support Role Based Access Control (RBAC). With RBAC, a role is used to represent a function (or group of functions) within the application that requires access control. For a user to execute that function, their user account must be assigned the role that corresponds to the function. As there can be a large number of functions within a system, there can also be a large number of roles. To simplify management of the roles, SOLA supports groups that combine multiple roles into logical groupings. Often job positions are used as the basis for groups as users can then be assigned the groups that correspond to their job position. To allow for people that have multiple roles within an organization, SOLA allows one or more groups to be assigned to a user. The user's access to system functions is then determined from all groups they are associated with.

To view the roles that have been configured for use in the SOLA application:
In SOLA Web Admin

1. Select **Security → Roles** page from the main menu. This gives a description for each role.

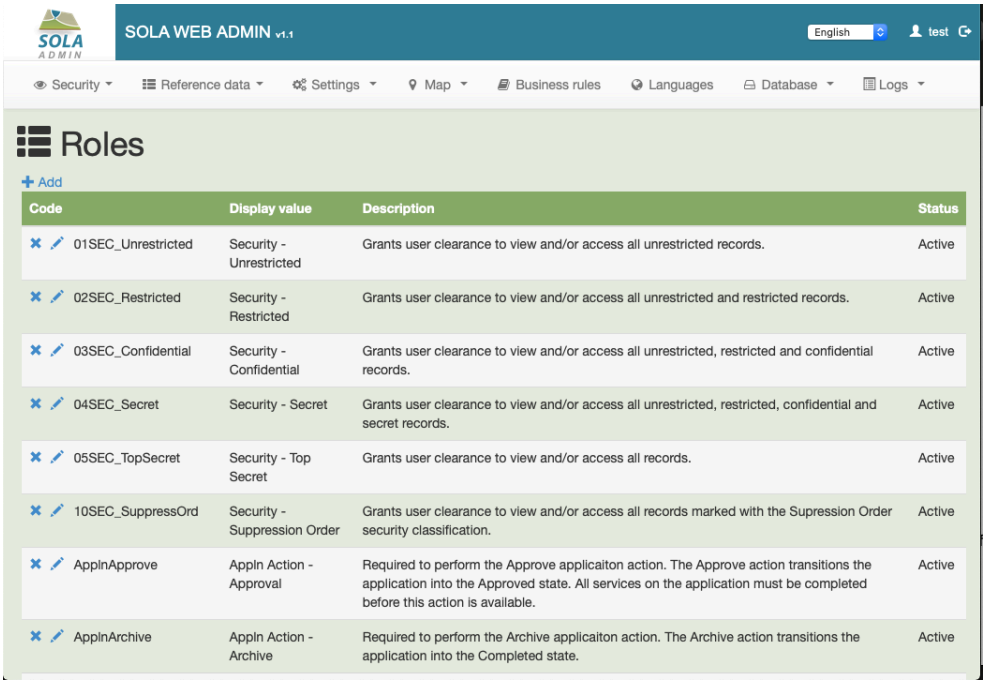


Figure 25 – Web Admin Roles

2. It is possible to add (using the Add link) and edit roles (using the **pencil icon** for the role to be changed), but *it is recommended that you do not modify these roles* unless a change to the SOLA software has caused a new role to be created or an existing role to be removed and you have been instructed by your software support specialist to make such a change.

7.2.1 Create User Groups

Groups in SOLA are groups of users who have the same roles. In SOLA Registry typically the Groups that are necessary are:

- Public Counter – receive applications & make enquiries
- Cadastral Processing – undertake cadastral processes
- Registration Processing – undertake registration processes
- Approvers - roles that approve registration & cadastral processes
- Statutory Approvals – roles that incorporate any statutory approval/clearance

Review the business processes that will be supported by the SOLA software and decide what Groups will be needed in your office and the Roles that are applicable to each group.

To create a new User Group
In SOLA Web Admin

1. Using SOLA Web Admin **Security → Groups** menu option,
2. Click on the **Add** link,
3. Enter the Groups name and tick roles that apply to that Group.
4. Then **Save**

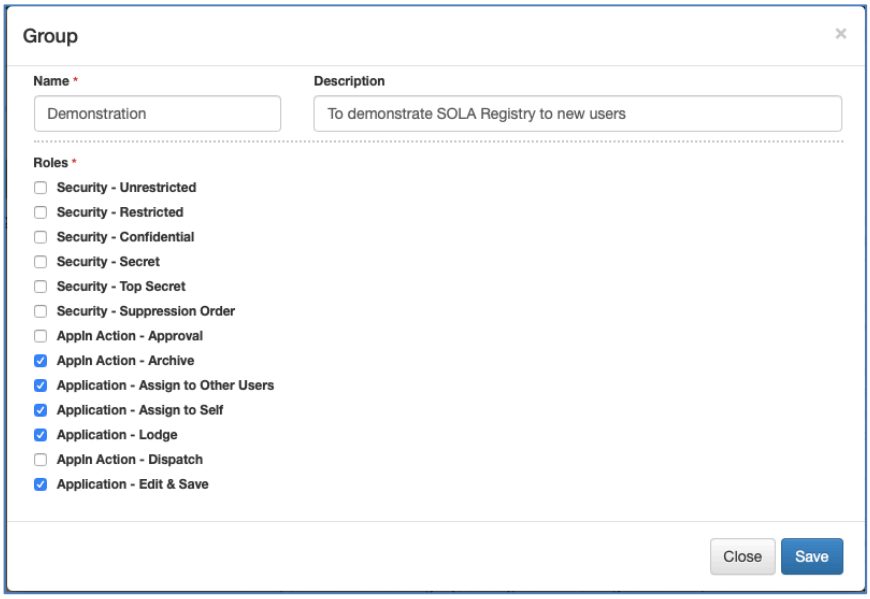


Figure 26 – Web Admin New Group

7.3 Enrol Users

User enrolment will be done by the system administrator using SOLA Web Admin.
User name and password should be selected with care to ensure:

- the user remembers the user name and password
- in countries where the language involves a unique font or there are some ambiguities in unicode definitions, user names and password in local font may not work. In such cases, consider using user names and passwords formed from Basic Latin numbers (eg mobile phone number, birthdays)

In SOLA Web Admin
The system administrator will :

1. Select **Security → Users** on the main menu
2. Click on the **Add** link (top left hand side of screen

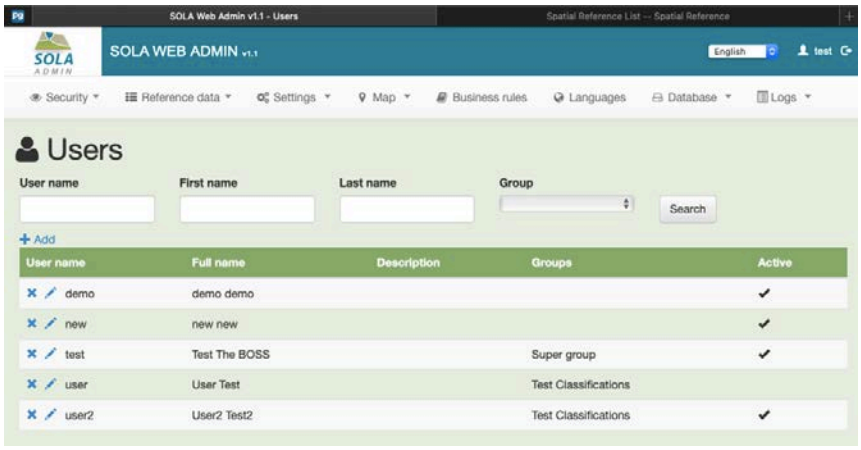


Figure 27 – Web Admin Add User to Group

3. Fill in the user details and tick the **Active** checkbox

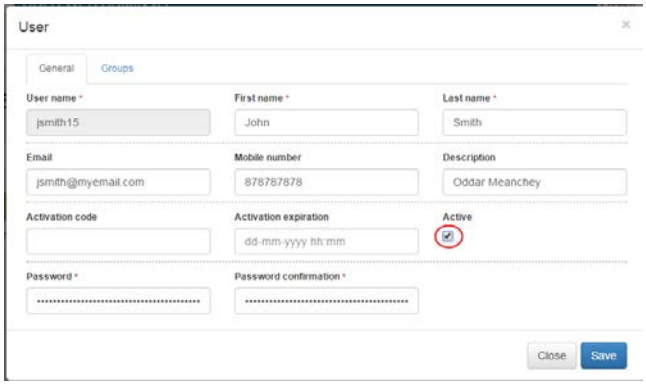


Figure 28 – Web Admin – Make User Active

4. Change to the **Groups tab** and identify which Groups the user will belong to
5. **Save** changes to enable the new user account.

Be sure to configure the appropriate security groups on the Groups tab otherwise the user will not be able to perform any actions in the SOLA application even if their user account has been activated.

7.4 Editing User Details

To edit user details, click the pencil icon beside the username in the Users page. If the Community Server has a large number of users, you can search for specific users by username, first name, last name or security group from the top section of the Users page.

7.5 User Passwords

There are three methods for resetting user passwords:

1. User Reset	Available in SOLA Community Server
2. Administrator Reset	The system administrator can reset a user's password from the User Details page (SOLA Web Admin). This approach should only be used if the user cannot use the Change Password functionality to reset their password such as when their email account is disabled or no longer accessible. After resetting a user's password, the Administrator will need to inform the user of their new password.

Passwords must be one or more characters in length. Users should be encouraged to create secure passwords of at least eight characters; however the SOLA servers do not enforce any password complexity constraints.

7.6 Disabling a User

Users can be disabled by unticking the Active checkbox on the User Details popup.
Users will also effectively become disabled if they do not have any security groups assigned to their user account. The User Details popup will not let you remove all security groups from a user, but it is possible to remove security groups from the Groups page which can lead to some users being deallocated from a security group.

7.7 Deleting a User

You should only delete a user when a mistake is made in the enrolment of a new user
To delete a user:

1. go to the Users page (SOLA Web Admin main menu select the **Security→ Users** menu option)
2. Click the **x icon** beside the username in the Users page.
3. A popup message will confirm the action before deleting the user account.

When you wish to withdraw access to a particular user:

1. go to the Users page (SOLA Web Admin main menu select the **Security→ Users** menu option)
2. Click the **Pencil icon** beside the username in the Users page
3. Disable the user account by unticking the **Active checkbox** in the User Details popup.
4. Click on **Save**

7.8 User Management details stored in SOLA Database

The SOLA database tables and views from System schema used to store user management details are:

system.appuser	The list of users that can have access to the SOLA application.
system.approle	Contains the list of application security roles used to restrict access to different parts of the application, both on the server and client side.
system.appgroup	Groups application security roles to simplify assignment of roles to individual system users.
system.approle_appgroup	Associates the application security roles to the groups. One role can exist in many groups.
system.appuser_appgroup	Associates users to groups. Each user can be assigned multiple groups.
system.active_users	View. Identifies the users currently active in the system. If the users password has expired, then they are treated as inactive users, unless they are System Administrators. This view is intended to replace the system.appuser table in the SolaRealm User Table property in Payara.
system.user_roles	View. Determines the application security roles assigned to each user. Referenced by the SolaRealm security configuration in Payara.
system.user_pword_expiry	View. Determines the number of days until the user's password expires. Once the number of days reaches 0, users will not be able to log into SOLA applications unless they have the ManageSecurity role (i.e. role to change manage user accounts) or the NoPasswordExpiry role. To configure the number of days before a password expires, set the pword-expiry-days setting in system.setting table. If this setting is not in place or is deactivated, then a password expiry does not apply.

8. Reference Data

8.1 General

The configuration of the SOLA reference data is an essential process in preparing SOLA Community Server for use. Reference data irrelevant to your implementation should be marked "inactive" and those elements of reference data deemed relevant should be described in terms of local terminology.
To review the SOLA codelists :
In SOLA Web Admin:

1. Select the **Reference Data** menu option and navigate through the sub-menu options to page corresponding to the reference data page
2. Click on the **Pencil** icon for each row (codelist value) and determine if it is relevant for your SOLA Community Server implementation. If it is click the ensure the appropriate **active checkbox** or leave this checkbox clear is not relevant
3. If the codelist value is relevant review the **Display Values** and amend to reflect the local land administration terminology.
4. Click on **Save** button

The following subsections identify the reference data tables that should be reviewed and modified as described above. It should be noted that all other reference data tables should be left unchanged (these tables areonly be modified by the software developers responsible for the updating of the SOLA software).

8.2 Administrative Reference Data

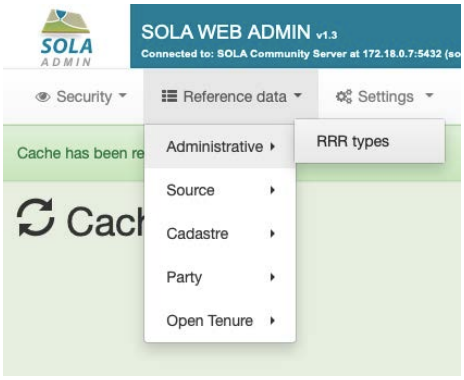


Figure 29 – Web Admin – Administrative Reference data

The Administrative reference data tables should be reviewed and modified as described in earlier in Section 8:

- RRR types – very important to review

8.3 Source Reference Data

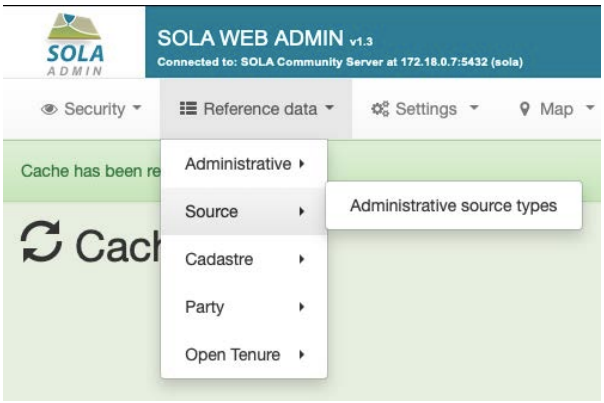


Figure 30 – Web Admin – Source Reference data

The Source reference data tables should be reviewed and modified as described in earlier in Section 8:

- Administrative source (Document) types – very important to review

8.4 Cadastre Reference Data

Figure 31 – Web Admin – Cadastre Reference data

The Cadastre reference data tables should be reviewed and modified as described in earlier in Section 8:

- Land use types

8.5 Party Reference Data

Figure 32 – Web Admin – Party Reference data

The Source reference data tables should be reviewed and modified as described in earlier in Section 8:

- Gender types
- ID types – very important to review

8.6 Open Tenure Reference Data

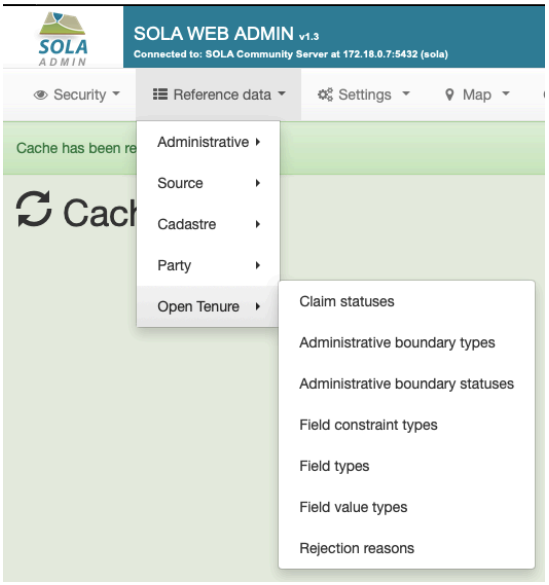


Figure 33 – Web Admin – Open Tenure

The Open Tenure reference data tables that should be reviewed and modified as described in earlier in Section **8** are:

- Community Area types (fomerly known as Administrative boundary types)[14]
- Rejection Reasons (for use in processing Claims)

These values can be edited using SOLA Web Admin:
In SOLA Web Admin

1. In the **Settings**✎✉🔔**System settings** menu option review each row
2. Where a change is required click on the **Pencil icon** and edit accordingly
3. Click on the **Save** button where an edit is made

[14] Any new Community Area types will probably need to be reflected with changes in the Geoserver Community Area styles definitions (Section 5.5.3)

9. Email Service

It is possible to configure SOLA Community Server to send system generated messages to users for certain events.

9.1 Email variables

In SOLA Web Admin
Click on the **Settings → System settings** menu option.

1. Then select each of these email setting variables and enter appropriate details and messages for your SOLA Community Server implementation
2. Make sure the **Active** checkbox is ticked.
3. Click on the Save button when each variable edit is complete

email-admin-address	Email address of server administrator. If empty, no notifications will be sent
email-admin-name	Name of server administrator
email-body-format	Message body format. text - for simple text format, html - for html format
email-enable-email-service	Enables or disables email service. 1 - enable, 0 - disable
email-mailer-jndi-name	Configured mailer service JNDI name
email-msg-claim-approve-moderation-body	Wording of claim moderation email
email-msg-claim-……..	Wording of various email messages associated with the claim
email-send-attempts1	Number of attempts to send email with first interval
email-send-attempts2	Number of attempts to send email with second interval
email-send-attempts3	Number of attempts to send email with third interval
email-send-interval1	Time interval in minutes for the first attempt to send email message.
email-send-interval2	Time interval in minutes for the second attempt to send email message.
email-send-interval3	Time interval in minutes for the third attempt to send email message.
email-service-interval	Time interval in seconds for email service to check and process scheduled messages.

Figure 34 – Email Settings in Web Admin Settings

If one of these email variables is not recorded in the Web Admin System Settings form, then the default value will apply. In those cases when you want to apply some other value, then you will need to add a new record and make sure the setting name is entered exactly as is given below.

9.2 Payara Server JNDI Mail Service

In addition to creating or editing email variables as system settings, it is also necessary to make certain changes to the Payara Server setup.
In the example below settings for Google mail server are given. If you intend to use a different email provider, modify the properties listed in step 5 to match your email provider details.

1. Open Payara Server administration console at http://localhost:4848 and go to **Resources → JavaMail Sessions**
2. Click "**New**" button to create new mail service.
3. Enter the following settings:
    a. JNDI Name = mail/sola
    b. Mail Host = smtp.gmail.com
    c. Default User = SOLA Mailer

   d. Default Sender Address = <*your_mailbox@gmail.com*>
4. Leave **Advanced** settings with default values
5. In the **Additional Properties** table add the following properties:
   a. mail-smtp-host = smtp.gmail.com
   b. mail-smtp-password = <*your_mailbox_password*>
   c. mail-smtp-socketFactory-class = javax.net.ssl.SSLSocketFactory
   d. mail-smtp-auth = true
   e. mail-smtp-socketFactory-port = 465
   f. mail-smtp-port = 465
   g. mail-smtp-starttls-enable = true
   h. mail.smtp.connectiontimeout = 60000
   i. mail.smtp.timeout = 180000
   j. mail-smtp-user = <your_mailbox@gmail.com>
   k. mail-smtp-socketFactory-fallback = false
6. Save settings by clicking "**Save**" button

9.3 Change security settings for Gmail account

Logon to the Google account being used as the gmail email account for the Community Server email service and under **Security** top menubar option click and turn **ON** the **Less secure app access**.

9.4 Email Log

In the SOLA Web Admin application, under the **Logs → Mail queue** menu option it is possible to view a table of email messages generated by the server with details regarding the recipients, the subject line and body of the message and whether the email had been successfully sent.
If users report that they are not receiving SOLA Community Server emails, this log is a good starting point in trouble-shooting such problems.

9.5 SOLA Database Table for Email

The SOLA database table used with system generated emails is:

system.email	Contains all emails that have been logged, but are yet to be sent by the server. Emails that are sent successfully by server are deleted from this table. To view the history of all emails that have been sent by the server, see the Sent folder of the email account that the server uses for the system generated email.

10. Reports

SOLA Community Server provides for the generation of a certificate noting a recorded tenure right (claim). If it is generated before the community has validated the claim (typically through a community based claim moderation process), the certificate prints with a watermark indicating its "unmoderated" status.

In addition to the Community Server certificate, Open Tenure also generates a "Claim Summary" manually during the recording of the claim or automatically at the time the claim is uploaded to the Community Server (as a supporting document to the claim).

Further report templates (primarily reporting on tenure claims that have been or are being processed) are also included with the Community Server software. These are report **templates** that have been created using the open source JasperReports; a report design and generation tool. A community may choose to use these report templates as a starting point for their own reports. If the JasperReports Server link is activated (through a configuration setting in SOLA Web Admin), this allows any report templates defined and generated in JasperReports Studio and stored in the **Reports** folder (definable in SOLA Web Admin), to be made available to users through the Reports menu in the Community Server.

10.1 Configuring JasperReports Server

Before defining (and potentially publishing) report templates through JasperReports server, you need to create a data source resource and language input control. The data source is required to query the Community Server database on the server side and language input control is used for reports to pass language code for localization purposes.

To create a new JasperReports data resource following the next steps:

1) In **Payara Admin Console**, make sure you have configured JDBC resources on your Paraya server to connect target database (Resources→JDBC menu option – check both **Connection Pools** (that **jasperserver_pool** can **Ping**) and in **JDBC Resources** (that there is an entry for **jdbc/jasperserver**) sub-menu options)
2) In **JasperReports Admin Console**, right click on the folder where you want to create data source and select **Add Resource → Data Source** from the context menu. It is advised to create it in the **Data Sources** folder to have a more logical structure.

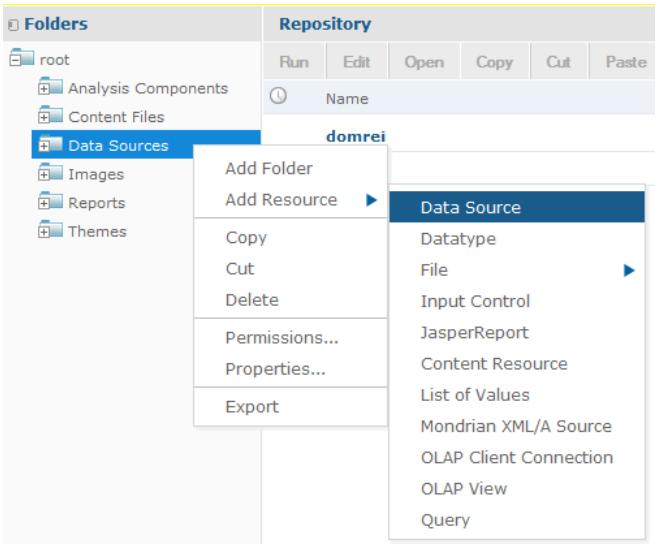


Figure 35 – JasperReports – Add New Data Source

3) On the Data Source Type and Properties page select "**JDBC Data Source**" and enter details as indicated in the following figure. Please note that while the illustrated details

are suitable for docker installations, with non-Docker installations the host IP address should be added to Host field (or alternatively use the **JNDI Data Source** option to reference "jdbc/sola")

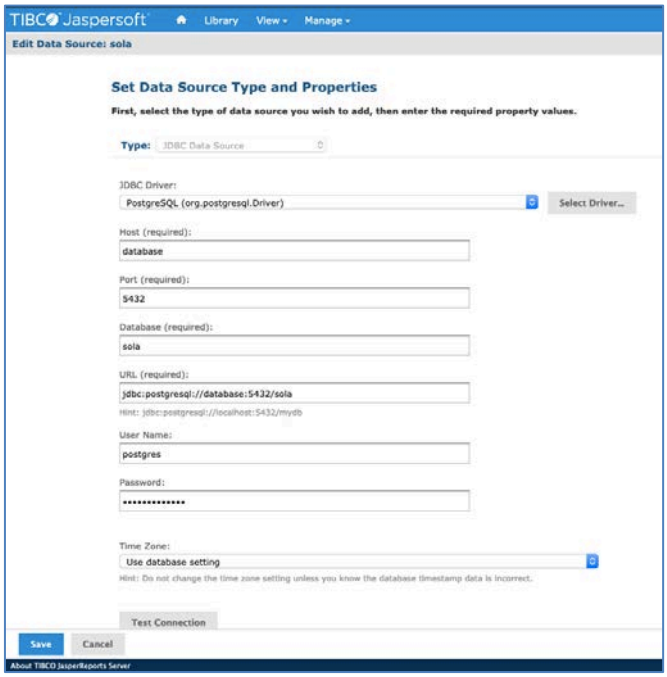


Figure 36 – JasperReports – New Data Source Details

4) Click **Test Connection** button to check connectivity.
5) If connection is successful, click on **Save**.
6) On the popup window provide Data Source Name (e.g. sola), which will be displayed in the folder as a resource name.
7) Click Save button to finalize data source creation.

Next stage is to create **input control**, which will be used later to bind it to reports.

1) Create **Input controls** folder in the **Reports** folder.
2) Select created folder and right click on it.
3) Select **Add Resource → Input Control** from the context menu.
4) On the Create Input Control page provide the following values:
    a. Type = **Single Value**`
    b. Prompt Text = **Language**
    c. Parameter Name = **lang**
    d. Remove all checkboxes from "**Mandatory**", "**Read-only**", "**Visible**"
5) Click **Next** button to proceed.
6) On the **Locate Datatypes** page, leave "**Define a Datatype in the next step**" and click **Next** button.
7) On the **Set the Datatype Kind and Properties** page provide the following values:
    a. Type = Text
    b. Name = text
    c. Resource ID = text
8) Click **Save** button to finish creation of input control.

10.1.1 Publish Report Templates on JasperReports Server

Finally, we need to add a new folder under the JasperReports folder, name it "Community Server" and upload the existing Community Server report templates into this folder. The

existing Community Server report files and associated language resource bundles can be downloaded from github (https://github.com/OpenTenure/reports.git).

Continuing within JasperReports Server:

1) On the Reports node, right mouse click and **Add Folder**, "community_server"
2) On the new community_server folder, right mouse click **Add Resource→JasperReport** and complete the following details in the Setup the Report form:
   - In the **Locate the JRXML File** field click on the **Browse** button and identify a report jrxml file that has been downloaded from the OpenTenure/reports github repository (e.g. ClaimSummary.jrxml) and click on **Open**
   - Enter a **Name** for the report (eg Claim Summary)
   - Enter a **Resource ID** for the report (eg ClaimSummary)
   - Click on **Submit**

   Each of the existing Community Server reports has an input parameter used for the language localization of the reports. Unfortunately JasperReports Server does not automatically pickup up these input parameters during this report publishing stage. This means we have to add the input parameters manually within the JasperReports Server Admin Console continuing with the following steps:
3) Locate the report (added in the last step) and select it (do not click on the report name, but click on the row itself
4) On the toolbar click **Edit** column heading.
5) Click on the **Controls & Resources** menu on the left and wait for the link "**Add Resource…**" to appear under **Controls &Resources** header. Default Community Server reports have **bundle.properties** file as well as other similarly named files used for localization and they have to be published together with report (eg bundle_ru_RU.properties file). There may also be image resource files. For each of these resource files
   a. Browse your resource (eg Bundle.properties) in the "**Upload a Local File**" option and click **Next** button.
   b. Provide resource name on the next page. It's highly advised to enter the same name as resource file name without extension i.e. Bundle.
   c. Click **Next** to finalize adding of resource.
   d. Repeat steps a-c to add other resources.
6) On the "**Set Up the Report**" page click on the **Controls&Resources** menu item on the left.
7) Click "**Add Input Control…**" link under "**Input Controls**" header.
8) On the "**Locate Input Control**" page, select option "**Select an Input Control from the repository**" and click **Browse…** button.
9) Locate the language input control and **Select** it.
10) Click **Next** to finish adding of input control.
11) Click on the **"Data Source"** menu item on the left.
12) On the "**Link a Data Source to the Report**" page select option "**Select data source from repository**" and click **Browse…** button.
13) Locate the **sola** data source (previously created) and **select** it.
14) Click **Submit** button at the bottom to apply changes.

10.1.2 Publish Certificate Template on JasperReports Server

Finally, we need to add a new folder under the root node, name it "Certificate" and upload the existing Claim Certificate template and the associated resources and input controls into this folder. The existing Claim Certificate jrxml report file definitions and associated language resource bundles can be downloaded from github (https://github.com/OpenTenure/reports/tree/master/ClaimCertificate).

Continuing within JasperReports Server:

1) On the **root** node, right mouse click and **Add Folder**, "Certificate"
2) On the new certificate folder, right mouse click **Add Resource>JasperReport** and complete the following details in the Setup the Report form:
    - In the **Locate the JRXML File** field click on the **Browse** button and identify ClaimCertificate jrxml file that has been downloaded from the OpenTenure/reports github repository and click on **Open**
    - Enter a **Name** for the report (eg Claim Certificate)
    - Enter a **Resource ID** for the report (eg ClaimCertificate)
    - Click on **Submit**

    Like the other existing Community Server reports have resources and input parameters. Unfortunately JasperReports Server does not automatically pickup up these input parameters during this report publishing stage. This means we have to add these resources and parameters manually within the JasperReports Server Admin Console continuing with the following steps:
3) Locate the report (added in the last step) and select it (do not click on the report name, but click on the row itself
4) On the toolbar click **Edit** column heading.
5) Click on the **Controls & Resources** menu on the left and wait for the link "**Add Resource…**" to appear under **Controls &Resources** header. Default Community Server reports have **Bundle.properties** file as well as other similarly named files used for localization and they have to be published together with report bundle (eg bundle_ru_RU.properties) file(s)). Including the **Shares.**jrxml and the **logo.png** files. For each of these resource files
    a. Browse for each of the resource files provided in the downloaded Reports/ClaimCertificate sub folder one by onein the "**Upload a Local File**" option and once a file has been highlighted, click the **Next** button.
    b. Provide resource name on the next page. It's advised to enter the same name as resource file name with extension ie Bundle.properties.
    c. Click **Next** to finalize adding of resource.
    d. Repeat steps a-c to add other resources.
6) On the "**Set Up the Report**" page click on the **Controls&Resources** menu item on the left.
7) Click "**Add Input Control…**" link under "**Input Controls**" header.
8) On the "**Locate Input Control**" page, select option "**Select an Input Control from the repository**" and click **Browse…** button.
9) Locate the language input control and **Select** it.
10) Click **Next** to finish adding of input control.
11) With the "**Add Input Control…**" link create further input controls for each of these parameters – **id**, **imageUrl**, **CommunityName** by using the **Define an Input Control in next step** option, click on **Next** then populate with the **Single Value** list item, ID for Prompt value, id for Parameter name and check **Visible** checkbox. Click on **Next** and then **Define Datatype in next step. Click Next.** Populate with **Text** list item, **myDataType** for both Name and Resource ID. Click on **Save** and following the **ID** input control then repeat for **imageUrl** and **CommunityName**
12) Click on the **"Data Source"** menu item on the left.
13) On the "**Link a Data Source to the Report**" page select option "**Select data source from repository**" and click **Browse…** button.
14) Locate the **sola** data source (previously created) and **select** it.
15) Click **Submit** button at the bottom to apply changes.

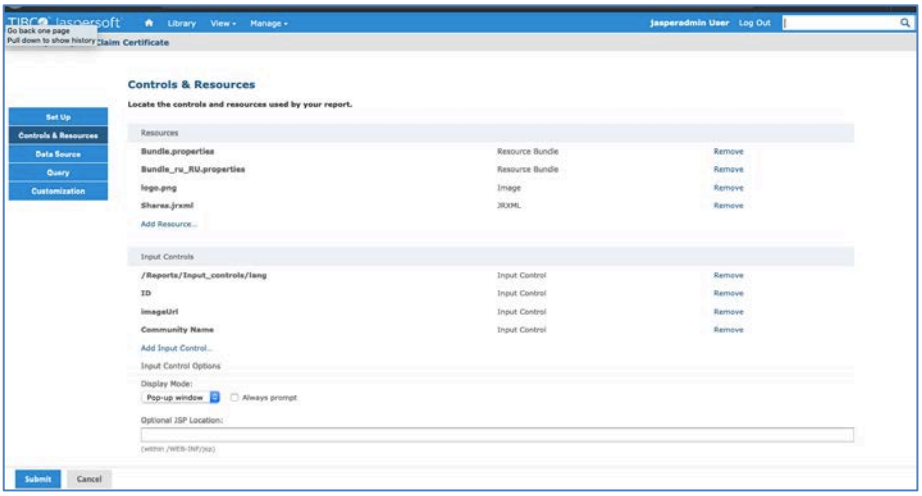


Figure 37 – JasperReports – Controls & Resources

16) Test in Jasper Admin Console (use PGadmin to obtain ID value of a claim from opentenure.claim table; use http://*<<hostIPaddress>>*/claim/GetMapImage.xhtml for imageUrl value; and "Test" for CommunityName)

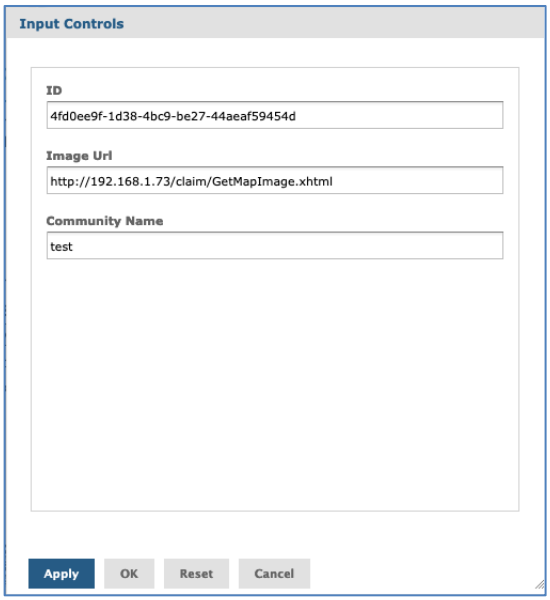


Figure 38 – JasperReports – Input for test display of Claim Certificate

10.2 Confirm Community Server Configuration

After finishing setup of JasperReports Server and publishing reports you have to configure Community Server in order to link it with reports server. Follow these steps complete configuration:

1) Open SOLA Web Admin application and login into the system.
2) Go to the **Settings → System Settings** menu.
3) Scroll down and locate **enable_report** setting and change value to **1**.
4) Scroll down and locate **report_server_pass** setting.
5) Make changes according to your JasperReports Server configuration:
    a. **claim_certificate_report_url** - /Certificate/certificate/ClaimCertificate
    b. **claim_certificate_map_print_url** – This URL will vary according to the how Community Server and JasperReports are installed, on the version of Jasper Reports and the ports used. If Payara and JasperReports Server are installed on

the same server, you should be able to use "**localhost**" in the URL but if this does not work use the IP address of host system. For non-docker installations this is http://localhost:8080/claim/GetMapImage.xhtml. For the standard CS docker installation it is currently http://<<*hostIPaddress*>>/claim/GetMapImage.xhtml

c. **enable_reports** – set to 1
d. **report_server_pass** – User password used to access reports server (jasperadmin if password has not been changed)
e. **report_server_url** – URL of reports server. This URL will vary according to the how Community Server and JasperReports are installed, on the version of Jasper Reports and the ports used. If Payara and JasperReports Server are installed on the same server, you should be able to use "**localhost**" in the URL but if this does not work use the IP address of host system. For non-docker installations this is http://localhost:8080/jasperserver (Jasper Reports version 6) or http://localhost:8080 (Jasper Reports version 7). For the standard CS docker installation it is currently http://hostIPaddress:8090
f. **report_server_user** – User name used to access reports server (jasperadmin).
g. **reports_folder_url** – Folder path with reports on the JasperReports Server. You have to provide "**/**" at the beginning of the path. If you have followed these installation instructions and placed your folder or reports in the **Reports** folder the value will be **/Reports/community_server**. To check this value open JasperServer (http://localhost:8090) and logon jasperadmin/jasperadmin), select the Reports folder, right mouse click and select Properties.

6) Finally select **Cache → Cache reset** menu item, to clear the old cached value.
7) Open main Community Server Web-site and check reports accessibility. If you cannot see **Reports** menu, check that user name you are using to login belongs to the group(s) with "**View Community Server reports**" role. Otherwise, go to the **Security → Groups** to add this role to the group.

10.3 Modify Existing or Create New Reports

If you need to create a new JasperReport report definition or modify an existing report definition for use within the Community Server, the Jaspersoft Studio software can be used for this purpose.

10.3.1 Install Jaspersoft Studio

1 Jaspersoft Studio can be downloaded from http://community.jaspersoft.com/project/jaspersoft-studio/releases (SOLA uses version 6.5.1).
Select the version applicable to the operating system:
Ubuntu: TIB_js-studiocomm_6.5.1.final_linux_amd64.deb
Windows: TIB_js-studiocomm_6.5.1.final_windows_x86_64.exe/download

2 To install

Windows	**Ubuntu**

In File Manager, select the downloaded .exe file and **Run as administrator** the downloaded file	In a Terminal window make the following commands:
	$ sudo dpkg –i ~/Downloads/ TIB_js-studiocomm_6.5.1.final_linux_amd64.deb
	$ sudo apt install -f

10.3.2 Configure Jaspersoft Studio

1) Run Jaspersoft Studio and create a new connection:
   a. Right click on the **Servers** in the **Repository Explorer** window and select **Create JasperReports Server Connection**.
   b. On the popup window enter server name (JasperReports Server), URL(http://localhost:8080/jasperserver), user (jasperadmin, *unless modified*) and password (jasperadmin, *unless modified*).
   c. Click **Test Connection** to make sure everything ok.

d. Click **Finish** button to save connection.

10.3.3 Using Jaspersoft Studio

Comprehensive Help and start-up tutorials can be found within Jaspersoft Studio.

Once a report definition has been modified or created within Jaspersoft Studio, this definition needs to be published to JasperReport Server. One method of publishing has been described previously in this guide – refer to sections 10.1.1 and 10.1.2.10.1.1

11. Configuring Dynamic Forms

11.1 The need to use Dynamic Forms

Dynamic Forms provide a community the means to record additional details relevant to that community.

Typically a Community Server is implemented without any "dynamic forms". If Dynamic Forms are defined, they are reflected in both the Community Server and those Open Tenure tablets that are associated with a Community Server and these additional data fields can be organised to appear on one or more forms/tabs.

Although the Dynamic Form definitions can be made at any time, it is best to ensure all data collected with one Dynamic Form definition is uploaded before the definition is changed.

With the initial Dynamic Form definition, the Community Technologist may decide to look at what Dynamic Form definitions have been used by other communities using Open Tenure to get ideas on what it is relevant for their community. If another example of Dynamic Form definition is found, that definition could be exported and then imported into the new community's Web Admin to save time.

It should also be noted that the Dynamic Form definition allows for a definition to be made in any single or in multiple languages. As a minimum this definition should be in the community's language of choice although if the community wants technical support from an external specialist, it would be useful to populate the Display value field understood by the technical support specialist.

Keep your initial dynamic form definitions simple and check each dynamic form before defining the next dynamic tab.

11.2 Establish a Dynamic Form

The first step you need to do is to design your form.

To illustrate the steps to follow to create dynamic forms a new tab (University Assets) will be described. Figure 39 illustrates this example and includes

- A single tab (University Assets)
- 6 fields (see Figure 39)
    - Asset ID (including constraint that it must be an integer)
    - Field Inspection Check (including constraints that it is Boolean (0 or 1) and also a mandatory field)
    - Street Address (text field unconstrained)
    - University Asset Type (a drop-down list)
    - Number of carparks (including constraint that it must be an integer
    - Date of inspection (including constraints that it is in DATE format and mandatory)

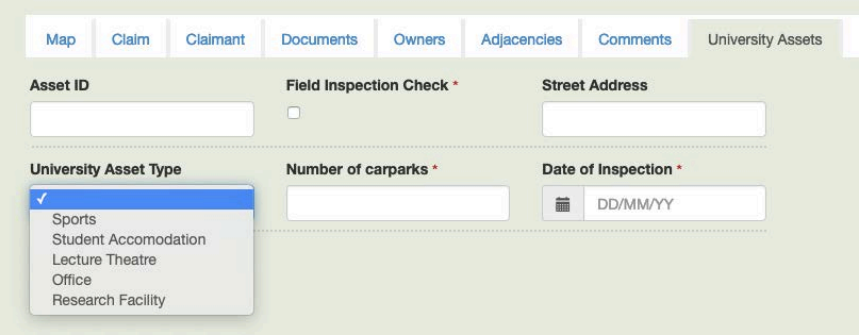


Figure 39 – Example Dynamic Tab (Community Server)

Dynamic tab(s) for both Community Server and Open Tenure are defined through the Web Admin **Settings – Open Tenure – Dynamic forms** menu option

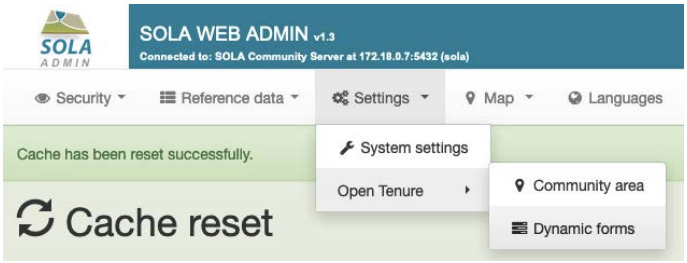


Figure 40 – Dynamic Form Defintion in Web Admin

If you are importing a Dynamic Form definition from another implementation of Community Server, click on the **Import** link, and upload the Dynamic Form export file from this other implementation. A new Dynamic Form will now be available in your implementation of Community Server.

A form definition (including one that has been imported) can be edited through this same menu option by identifying the form to be edited and clicking on the icon.

To export a form definition, identify the form to be exported and click on the icon.

Figure 41 – Dynamic Form options (New, Edit, Import, Export)

If you are creating a new Dynamic Form definition, click on the **New form** link and follow the following steps to create the example form from Figure 39 :

1. Click on the **Edit** icon and add a name for the new form in the **Display name** field
2. Click on the **+ Add Section** and complete the details for the fields in all 3 tabs

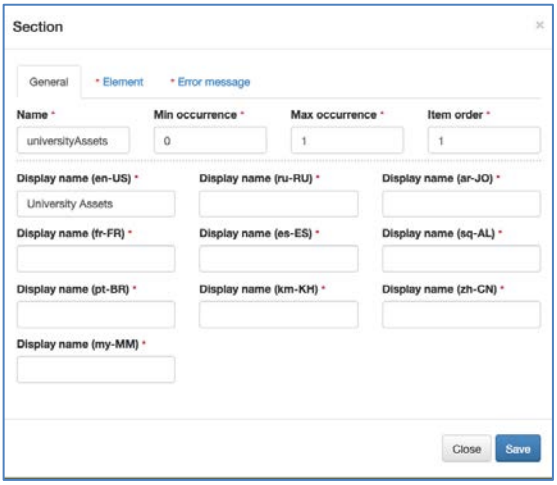


Figure 42 –Section Form General Tab details

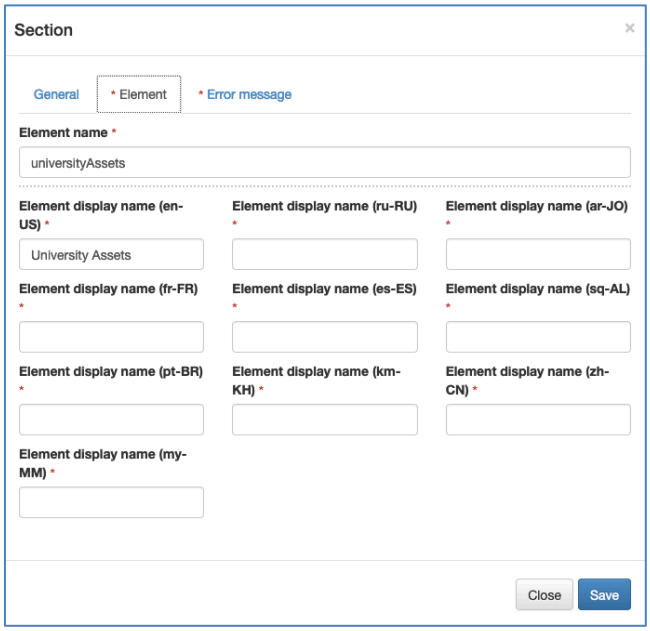


Figure 43 – Section Form Element tab

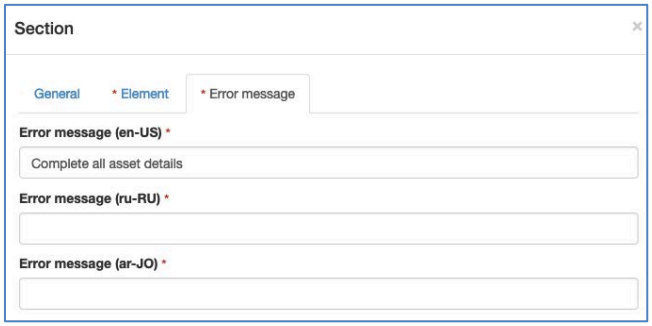


Figure 44 – Section Form Error message tab

3. Click on the **+ Add Field** and complete the (assetID) field details and then click on **Save** button

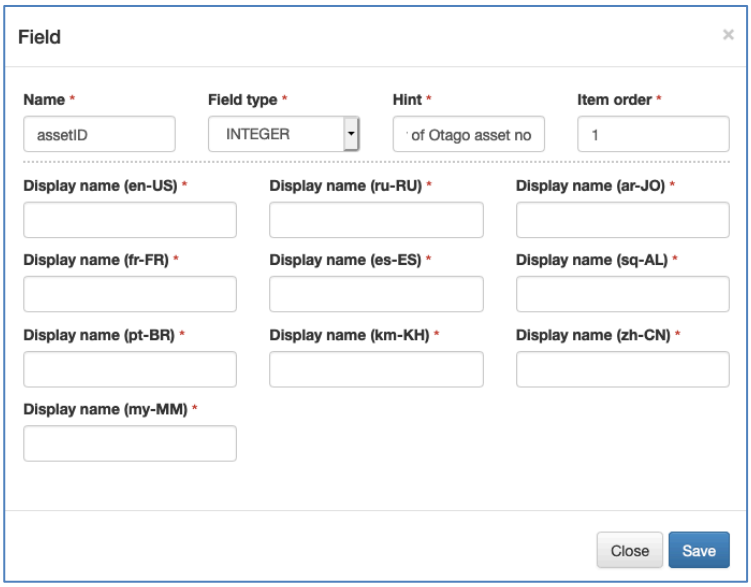


Figure 45 – Field Details

4. Click on the **+ Add Constraint** and complete the (assetIDtypeConstraint) constraint details, complete the **Error message** tab and then click on **Save** button

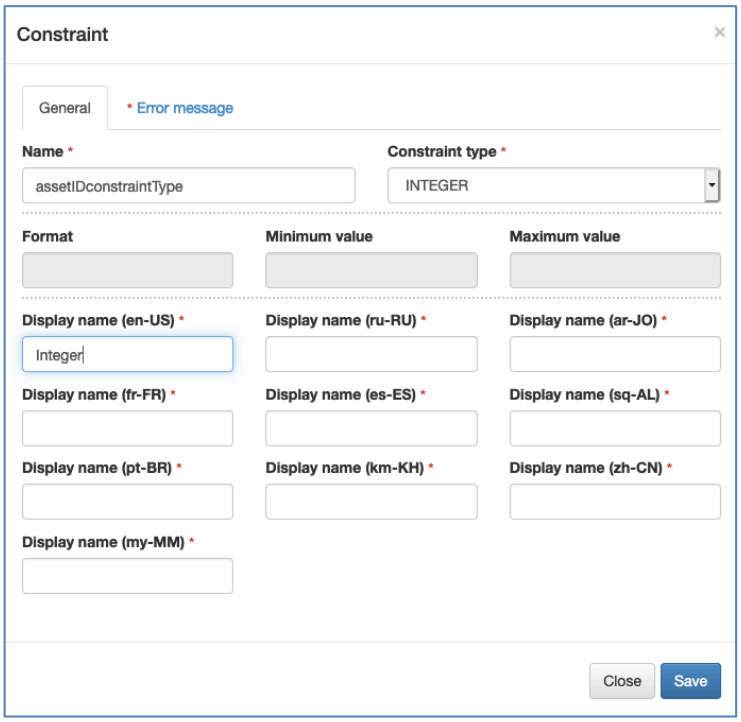


Figure 46 – Constraint Details General Tab

5. Click on the **+ Add Field** and complete the **Field Inspection Check** details and the two associated constraints **Field Check Mandatory Constraint** and **Check either 0 or** 1 including **Error message** tab details and **Save**.

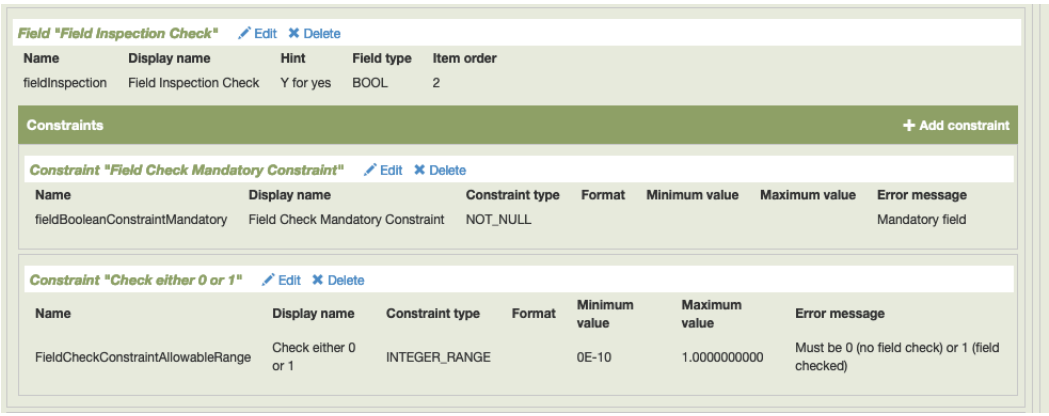


Figure 47 – Field with mandatory & range constraints

6. Click on the **+ Add Field** and complete the **Street Address** details (no associated constraints) and **Save**.

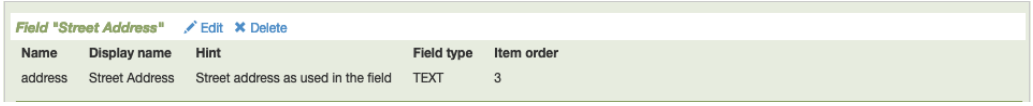


Figure 48 – Field with no constraints

7. Click on the **+ Add Field** and complete the **University Asset Type** details and the associated constraint **Asset Type List Constraint** including **Error message** tab details and **Save**.

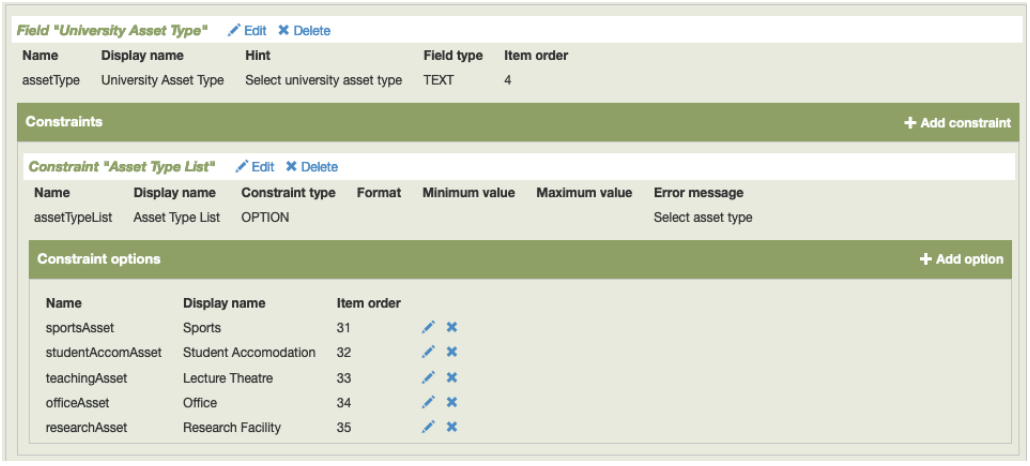


Figure 49 – Field with drop-down list

8. Click on the **+ Add option** and complete the (sportsAsset) details, and then click on **Save** button

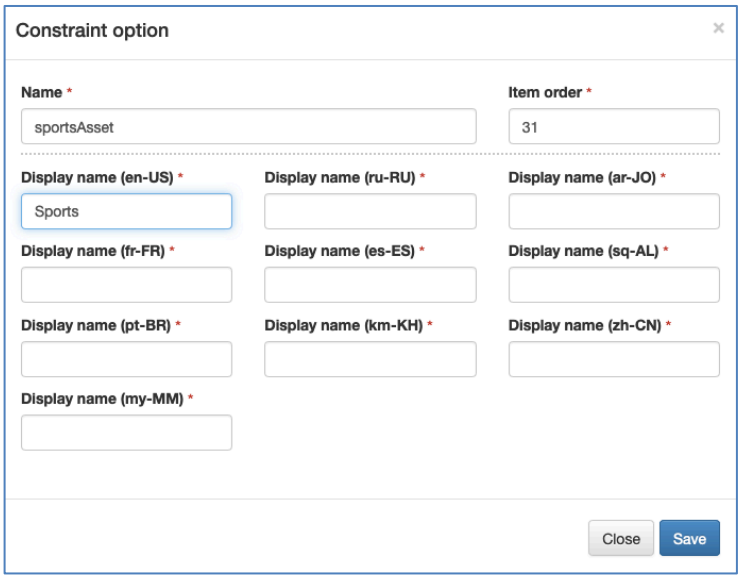


Figure 50 – Constraint option details (drop-down list)

9. **Add option** for the remaining constraint options (studentAccomAsset, teachingAsset, officeAsset and researchAsset) – see details in Figure 50
10. Click on the **+ Add Field** and complete the **Number of carparks** details and the two associated constraints **Integer Constraint for Carparks** and **Carpark mandatory constraint** including **Error message** tab details and **Save**.

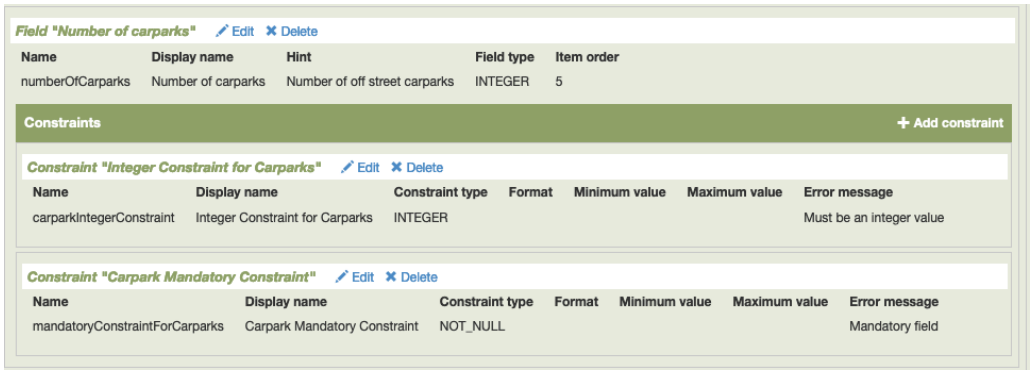


Figure 51 – Field with Integer and Mandatory constraints

11. Click on **Save** button and then the **Cache – Cache reset** menu option

12. Log onto both Community Server (New Claim) and Open Tenure (New Claim) and check that the new form displays and performs correctly.

12. Open Tenure (Client) Setup

12.1 Installation

12.1.1 Android

From the tablet, enter the Play Store, search for "Open Tenure", select and install.

In Android Settings ensure **Location** is set to **ON** and change the **Locating method** (on some devices as "Mode") from default "High Accuracy" to "**GPS only" (or "Device only")**.

If you are likely to be using the devices camera to include photos or scanning documents as supporting evidence for Open Tenure claims, you are advised to open the device's **Camera** app, click on its Setting toolbar icon and set the photo **Resolution** to **Low.**

It is strongly recommended that you also search for and install another two apps.

1. Tinyscanner Pro which allows the tablet to produce excellent multi-page pdf scans which can be incorporated into Open Tenure. However, please note that *Tinyscan Pro is not a free app and will cost about $6*.
2. SwiftKey Keyboard which provides a series of keyboards for languages not based on Latin alphabet. You must download both the app and the keyboards you will be using (SwiftKey app is free)

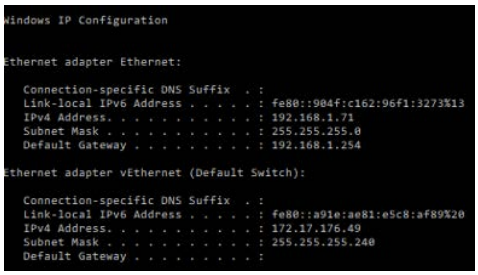
12.1.2 iOS

From the iPhone or iPad, enter the App Store, search for "OpenTenure", select and install.
Similarly Tinyscan can downloaded (upon payment) from the App Store.
Also SwiftKey Keyboard can be downloaded free from the App Store.

12.2 Preparation for Configuration

Tablets and mobile devices (running Android and iOS operating systems) do not generally recognise the computer name of the Community Server. For this reason in Open Tenure settings it is best to identify the Community Server by way of its IP address and to specifically identify the port and similar for the Geoserver URL setting. To determine the IP address of the Community Server open a Terminal window on the Community Server and make the following command:

Windows (as a Command prompt)
ipconfig
And this will display



Ubuntu
$ hostname -I

192.168.1.71 172.17.0.1 172.18.0.1

Figure 52 – Windows – IP address of host

The IP address of the Community Server in the examples above is **192.168.1.71**

12.3 Reinitialization

If a tablet or mobile device has used Open Tenure previously, you will need to "re-initialize" Open Tenure by:

1. clearing the (Open Tenure) data and settings through:

Android
In Android Settings navigate Apps-Open Tenure-Clear Data

iOS
In Open Tenure Settings page, click on the Garbage icon

2. Check through the Google Play Store (My Apps) that there are no outstanding Open Tenure updates waiting to be installed. Update if neccessary
3. Restart Open Tenure Restart Open Tenure

12.4 Configuration

On both Android and Apple devices, click on the Settings icon in the top right corner of the main screen to open the Settings form and enter the following settings:

Community Server URL: Mandatory. Examples:

- **https://demo.opentenure.org** – where host system has current SSL certificate
- **http://nossl.opentenure.org:8080** – where host system has **NO** current SSL certificate
- **http://192.168.1.71:8080** – for IP address access (including wifi connections)

Tiles Provider : select WTMS, TMS or Geoserver (WMS). Usually **Geoserver** is selected where the Community Server uses Geoserver to publish satellite imagery for Open Tenure and Community Server by way of a WMS service

Geoserver URL: Optional

Example: **http://imagery.opentenure.org:8085/geoserver/opentenure/wms**
or **http://192.168.1.71:8085/geoserver/opentenure/wms**

Geoserver layer: Optional

Example **opentenure:initial** (<<Geoserver workspace>>:<<Geoserver layer>>)

TMS URL : Optional (only completed when Open Tenure and Community Server will use TMS sourced imagery)

WTMS URL : Optional (only completed when Open Tenure and Community Server will use WTMS sourced imagery)

Form URL : Only completed if the Open Tenure dynamic form definitions are not stored on the Community Server

Language : Only completed if the language to be used for Open Tenure is a language not recognised by Android and Open Tenure has been modified to accept such as Cambodian (Khmer) and Albanian (Shqip).
For all other languages the **Default** setting is used.

Dynamic fields validation on submit : Only change this from **ON** if instructed by Open Tenure software support specialist

12.5 Initialisation

Once the community's Community Server URL setting has been entered the device must be initialised to link the device's Open Tenure software to the community's Community Server.

The device must be linked to the Community server by internet (if the Community Server is cloud based) or wirelessly (if the Community Server is local). The initialisation process is started by clicking on the white triangle toolbar icon and then confirming that you wish to proceed with the initialisation. When the initialisation is completed a message will inform you that initialisation has been completed successfully. The white triangle toolbar icon will have disappeared.

If the Community Server URL setting is incorrect, or if the internet or wireless connection is weak, a different message will display suggesting that you check the Community Server URL setting. Be aware that in addition to checking the URL setting you also need to consider an alternative means of connecting to the internet if you are using Open Tenure in an area with unreliable internet coverage.

13. Language Localization

13.1 Prerequisites

13.1.1 Arrangements

Before beginning a language localization, you need to make contact with a SOLA developer who is a member of the developer team for the SOLA github repositories that you wish to modify by adding a new language localization. Although, anyone can "pull" the software source code from the github repositories only the software developer team can "push" the files incorporating the language localization back into the SOLA github repositories so that others can benefit from the language localization effort.

Secondly, you need to be a software publisher on the Google Play Store and/or the Apple App Store or make contact and arrange for someone who is to publish the new version of the Open Tenure software and make it available to others.

Finally, you need to find a translator, ideally from English to the target language, who also has some familiarity with mapping/geospatial terminology.

13.1.2 Tools

For the software developer assisting with the new software localization, the essential tools to add a new language to SOLA Community Server and Open Tenure are:

- Netbeans IDE 8.2 Java SE bundle
  downloadable from https://netbeans.org/downloads/8.2/ for Windows, Linux (Ubuntu) and MacOS
- Android Studio
  downloadable from https://developer.android.com/studio#downloads
  for Windows, Linux (Ubuntu) and MacOS

Both these tools have the ability to make a copy (clone) of the github source code repositories and to post back the localization changes to the source code to github. However, there are other tools that manage the github processes including:

- **git** available for Windows, MacOS & Linux/Ubuntu) downloadable from https://git-scm.com/download
- **gitslave** available for Windows, MacOS and maybe Linux/Ubuntu downloadable from https://sourceforge.net/projects/gitslave/files/ - not maintained since 2010. Gitslave has the advantage of being able to manage all 10 SOLA github repositories with one series of commands, whereas the other alternatives work only on one github repository at a time. Refer to Section 0.
- **Github Desktop** for Windows & MacOS downloadable from https://desktop.github.com

The translator involved in the software localization will need to have access to:

- **Microsoft Excel**
- A text editor (eg in Windows, Notepad++)

13.2 Add new language into SOLA database language table

Currently the following languages are incorporated into some (but not all) of the SOLA software applications:

code	Display Value	Active	Default	Order	Left to Right
en-US	English	true	true	1	true
ru-RU	Русский	true	false	2	true
ar-JO	عربي	true	false	3	false
fr-FR	Français	true	false	4	true
es-ES	Español	true	false	5	true
pt-BR	Português	true	false	6	true

code	Display Value	Active	Default	Order	Left to Right
zh-CN	中国	true	false	7	true
km-KH	ខ្មែរ	false	false	8	true
sq-AL*	Shqip	false	false	9	true
my-MM	မြန်မာ	false	false	10	true
sm-WS	Samoa - English	false	false	11	true
en-AU	Australia English	false	false	12	true

Figure 53 – Current SOLA Localization Languages

1. When a new language localization is required for Open Tenure and SOLA Community Server, check the table above and see if this language has already been used in SOLA software.
2. Confirm this by launching SOLA **Web Admin** and click on the **Language** menu option and review the online Language form to see if the language is included.
3. If you confirm that the new language is not yet used by SOLA, click on the Add link and completing the following fields:
    a. **code** – language and country code separated by dash symbol "**-**" [ie <ISO 639 2 character code for language>-<ISO 3166 2 character code for country> e.g. en-US for English language, USA country). You can find supported locales by the following link: ftp://ftp.fu-berlin.de/doc/iso/iso3166-countrycodes.txt .
    b. **display_value** – language name, which will be shown on the languages list for selection. Follow languages item order when entering localized language name.
    c. **active** – boolean value, indicating whether to show language in the selection list or not. Set True value.
    d. **is_default** – boolean value indicating whether the language is default or not. True value must be set only to one language. Set **False** value.
    e. **item_order** – integer value to define the order of localized strings in the reference data tables. Set **next available number**.
    f. **ltr** - If the language is not a "left-to-right" language (such as English) but is a "right-to-left" languages (such as Arabic), change the new language record ltr field to **false**[15]

If the new language record is also the default language, the display value fields for each of the other language records must be reviewed and modified to reflect the default language.

Warning! Enter next sequential number in item order field for the new language. Do not change existing order. It will result into wrong localization of existing reference data values.

13.3 Download source code

Using your preferred means of obtaining a copy of a github repository (eg Netbeans IDE, Android Studio or Git Bash[16] (and gitslave commands)), download the following SOLA software source code repositories:

- OpenTenure/code
- OpenTenure/services
- OpenTenure/boundary
- OpenTenure/common
- OpenTenure/common
- OpenTenure/messaging

[15] The ltr field is missing in SOLA Web Admin version at the time of writing this Admin Guide. As an interim measure, use PGadmin4 to set the ltr attribute in the system.language table as TRUE.
[16] Refer to Section 0

- OpenTenure/database
- OpenTenure/reports
- OpenTenure/ot-android-studio
- OpenTenure/ot-ios
- SOLA-WA-FAO/code
- SOLA-WA-FAO/clients
- SOLA-WA-FAO/services
- SOLA-WA-FAO/common
- SOLA-WA-FAO/messaging
- SOLA-FAO/localization

13.4 Preparing Resource Files for Translation

To complete a language localization the files displaying text need to be replicated for the new localization language, translated and then reincorporated into the SOLA software which in this case involves Community Server, Web Admin, Open Tenure Android and Open Tenure iOS software applications.

13.4.1 SOLA Localizer tool

The SOLA Localizer tool extracts (and after translation, reincorporates) all error, messages and database reference table attributes into a spreadsheet (**resources.xls**) which can be given to the translator.
Required input for SOLA Localizer is:

- an instance of PostgreSQL with the SOLA database for Community Server and
- the SOLA (**Open Tenure** & **Web Admin**) source code for only the following Github repositorities:
    - OpenTenure/code
    - OpenTenure/services
    - OpenTenure/boundary
    - OpenTenure/common
    - OpenTenure/messaging

  in the standard SOLA Github repository (& file directory) structure. This structure is created automatically if you use the Gitbash and gitslave approach to download source code (described in Annex 2 - gitslave routine to download all Open Tenure repositories). Please note that the OpenTenure (Community Server) Github repository setup has only 7 repositories (and does not have a clients, help or rules Github repositories, as other SOLA applications do).

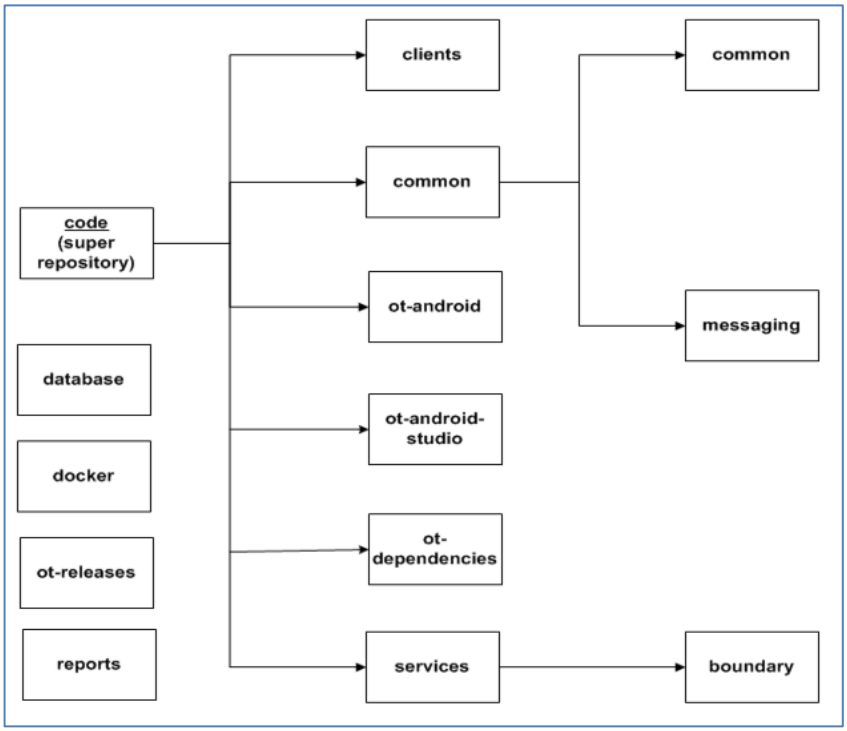


Figure 54 – Open Tenure (incl. SOLA Community Server) Github Repository Structure

OpenTenure/ot-android is a superseded Github repository. All other Open Tenure related Github repositories are stand-alone repositories. (eg OpenTenure/ot-ios, OpenTenure/latest-release, OpenTenure/reports and OpenTenure/docker). These standalone repositories are not required by the SOLA Localizer tool to create the resources.xls file of strings to be translated.

To run the SOLA Localizer tool:

1. Create a new PostgreSQL database called **sola** in pgAdmin. Run the **create_database script** downloaded in the OpenTenure/database Github repository. (**Y**es for create database and **N**o for fill with sample data)
2. Using pgAdmin, confirm that the SOLA Community Server database is available (can you see an opentenure schema in the SOLA database ?).
3. Confirm that both the Community Server and Web Admin source code in the **OpenTenure/code** and **SOLA-WA-FAO/code** conforms to the SOLA super Github repository structure (Figure 54)
4. Unzip the Localizer.zip file to be found in the **SOLA-FAO/localization** Github repository download
5. Run SOLA localizer tool

Windows	**Ubuntu**
In File Manager navigate to the Localizer folder containing the Localizer.jar file Select, right mouse click and **Run as Administrator** the **run.bat** file	$ cd ~/Localizer $ java -jar Localizer.jar

6. Complete the details for the SOLA Community Server PostgreSQL database, **en-**US as the language code and the location of the OpenTenure/code folder and then click on the **Export** button.

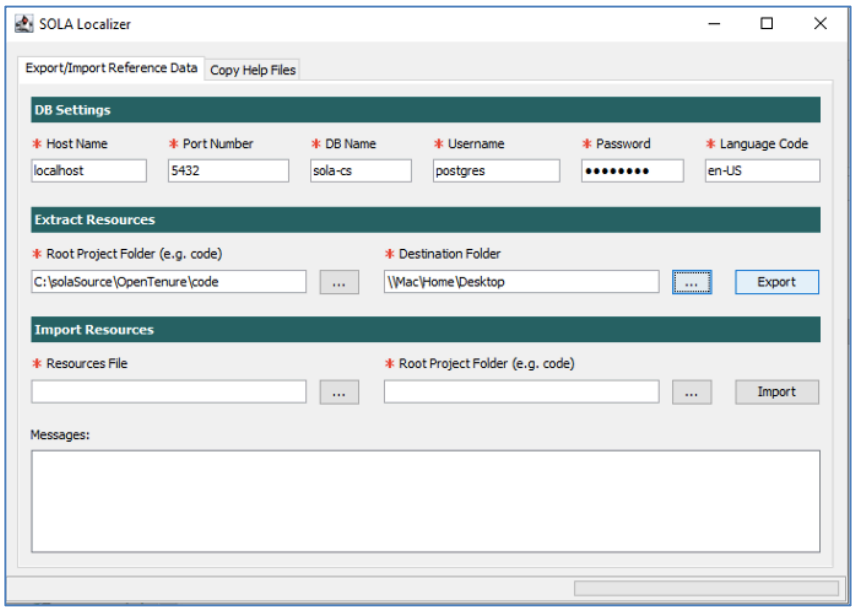


Figure 55 – SOLA Localizer tool Export

7. Rename the generated resources.xls file to **resources-cs.xls**
8. Complete the details for the location of the SOLA-WA-FAO/code folder (retain same database details and language code)
9. Rename the generate resources.xls to **resources-wa.xls**
10. Open **resources-wa.xls** and delete the Database tab (leaving just the Bundles tab) and **Save**

13.4.2 Make copies of specific source code files

1. From the **OpenTenure/boundary** Github repository download, take a copy of the following file for the translator:
..\OpenTenure\code\services\boundary\web\src\main\webapp\content**welcome.xhtml**
Rename copied file to **welcome*_nn_NN*** where *nn_NN* is the language code for the new localization language where *nn* is the ISO 631 2 character code for language name and *NN* is the ISO 3166-1 2 character country code.
2. From the **OpenTenure/ot-android-studio** Github repository download, take a copy of the following files for the translator:
..\ot-android-studio\openTenure\src\main\res\values**strings.xml**
..\ot-android-studio\openTenure\src\main\res\values**strings_activity_login.xml**
..\ot-android-studio\openTenure\src\main\res\values**strings_showcaseview.xml**
3. From the **OpenTenure/ot-ios** Github repository download, take a copy of the **..\ot-ios\Open Tenure\en.lproj** folder, **rename** the copied folder to ***nn.*lproj** where nn is the ISO 631 2 character code for the the language name. This folder contains these files for the translator:
..\ot-ios\OpenTenure\en.lproj**ActivityLogin.strings**
..\ot-ios\OpenTenure\en.lproj**Additional.strings**
..\ot-ios\OpenTenure\en.lproj**Localizable.strings**
..\ot-ios\OpenTenure\enlproj**Main_iPad.strings**
..\ot-ios\OpenTenure\en.lproj**Showcase.strings**

13.5 Instructions for Translator

The files to be translated are:

1. **Resources-cs.xls** – open in Excel and for both the **Database** and the **Bundles** tabs translate the strings in the left (green) column titled “String to translate” to the new language;
2. **Resources-wa.xls** – open in Excel and for the **Bundles** tab translate the strings in the left (green) column titled “String to translate” to the new language
3. **welcome_*nn_NN*.xhtml** – open in a text editor such as Notepad++ and translate the strings (not xhtml headers)
4. **strings.xml** – open in a text editor and translate the strings (not xml headers)
5. **strings_activity_login.xml** – open in a text editor and translate the strings (not xml headers)
6. **strings_showcaseview.xml** – open in a text editor and translate the strings (not xml headers)
7. **ActivityLogin.strings** – open in a text editor and translate the second string after the “=” character (not the first string)
8. **Additional.strings** – open in a text editor and translate the second string after the “=” character (not the first string)
9. **Localizable.strings** – open in a text editor and translate the second string after the “=” character (not the first string)
10. **Main_iPad.strings** – open in a text editor and translate the second string after the “=” character (not the first string)
11. **Showcase.strings** – open in a text editor and translate the second string after the “=” character (not the first string)

The translator must follow these rules:

Resources-cs.xls & Resources-wa.xls

1 Do not remove or add new rows or columns to the spreadsheets.
2 Do not translate values in red color (e.g. Default value column). On the Bundles tab you can find such values in the "String to translate" column as well. Just skip it.
3 If you meet in the text combinations like “{0}” or “%1”, do not remove them. They used to insert parameters. Place them in the sentence according to the sentence semantic.
4 Skip rubbish translation like “JText.Label”.
5 If string is broken into multiple lines with "\" symbol at the end, don't remove it.
6 Cells in turquoise color depict values which are the same as default language (English). This means translation is needed/missing.

welcome_*nn_NN*.xhtml

1 Do not remove any tags (e.g. <p>, <body>, etc.)
2 Translate text between tags only, don’t translate in the tag itself (e.g. don’t translate <ui:composition>).
3 <p> tags are used to split text, creating paragraphs.

strings.xml, **strings_activity_login.xml** and **strings_showcaseview.xml**

These XML documents use tags similar to **welcome_*nn_NN*.xhtml** file. So, translate only the text between tags (e.g. <string name="**action_search**">Search</string>, only string in bold has to be translated)

1 Do not remove characters such as “**%d**”, “**%s**”, “**%1$d**”, “**%2$d**”, etc., they are used to insert parameters. Place them according to the sentence logic.
2 Single quote displayed as “ **\'** ”. Keep this rule in mind if you need to use single quotes

ActivityLogin.strings, **Additional.strings**, **Localizable.strings**, **Main_iPad.strings**, **Showcase.strings**

/* Strings related to login */
"action_forgot_password"="**Recover lost password**";

1 Do not translate any line that begins with “/*

2 Do **not** translate the first string on a line (eg "action_forgot_password"=" from example above)

3 **Only** translate the second string (eg in bold in example above - **Recover lost password**)

13.6 Incorporating translated files into SOLA software

13.6.1 SOLA Localizer tool Import

To integrate the translated strings from **resource-cs.xls** and **resource-wa.xls**:

1. Using pgAdmin, confirm that the SOLA Community Server database is available (can you see an opentenure schema in the SOLA database). Note the name of the database.
2. Open both resource-cs.xls and resource-wa.sls in Excel and copy the **Database** tab from **resource-cs.xls** to **resource-wa.xls** (so that there is a copy of same translated Database tab in both xls files)
3. Run Localizer (see Section 13.4.1) enter the database details, the new localization language code and load the **resources-cs.xls** by clicking on the **Import** button
4. Repeat Localizer import for the **resources-wa.xls** spreadsheet file

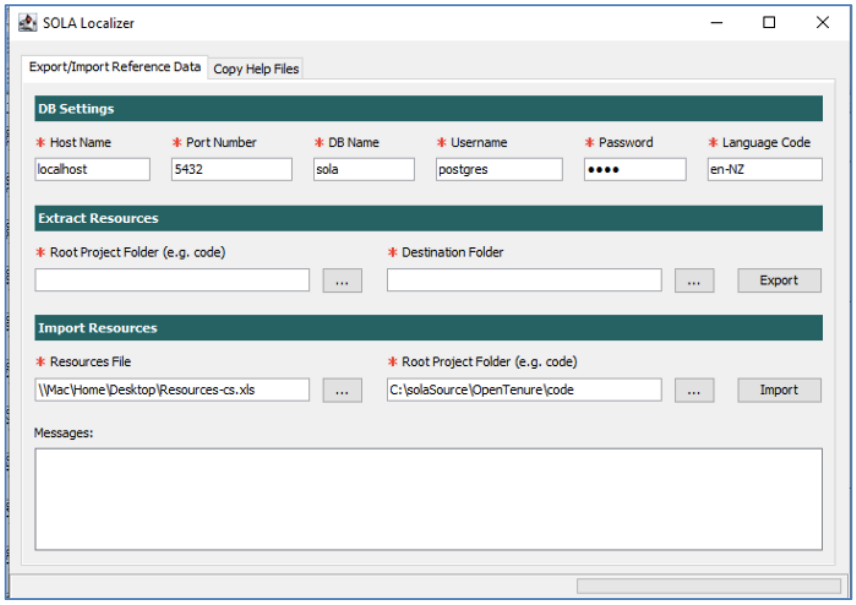


Figure 56 – SOLA Localizer tool Import

5. Using pgAdmin, make a database backup of the SOLA Community Server database.

13.6.2 Copy translated files into source code

When the translated files are received back from the translator, Open Tenure files (android and iOS) plus one of the Community Server files need to be separately incorporated into the source code repositories as they are not dealt with by the SOLA Localizer tool.

The translated Open Tenure files need to be incorporated by this process:

<u>General</u>

1. Check the language and country codes for the new localization language (for illustration purposes in this guide given as *nn-NN* but you must substitute the values of the new localization language and country whereever *nn-NN* is referred to)

<u>Community Server</u>

2. Copy **welcome_*nn_NN*.xhtml** into the **..\OpenTenure\code\services\boundary\web\src\main\webapp\content** folder of your downloaded source code

Open Tenure Android

3. Make a new folder ..\ot-android-studio\src\main\res**\values-*nn*** in your downloaded source code.
4. Copy **strings.xml**, **strings_activity_login.xml** and **strings_showcaseview.xml** into this new folder **..\ot-android-studio\src\main\res\values-*nn***

Open Tenure iOS

5. Make a new folder ..\ot-ios\Open Tenure***nn*.lpg** in your downloaded source code.
6. Copy **ActivityLogin.strings**, **Additional.strings**, **Localizable.strings**, **Main_iPad.strings**, **Showcase.strings** into this new folder **..\ot-ios\Open Tenure*nn*.lpg**

13.6.3 Open Tenure Android Software Language Extension

As part of the localization process, a check needs to be made on the Android devices most likely to be used with Open Tenure in the new community where the new language is required to see what languages and locales (including keyboard) are supported on those Android devices. This check can be done by through the Android settings (Language & input) where you can check the list of languages and also the options available for Current Keyboard.

Where the new language is not one of the languages (and locale) recognised by Android, then the Open Tenure software needs to also be extended to recognise the new language following these steps:

Open Tenure Android

Open Tenure has already been localised for Arabic, Spanish, French, Italian, Portuguese, Russian, Khmer/Cambodia, Albanian and Myanmar/Burmese languages.

Open Tenure needed to be extended for the Khmer, Albanian, Myanmar and Samoan language localizations because those languages were not recognized by Android at the time of the Open Tenure localizations.

These software changes to ensure Open Tenure picks up the new language affect 5 ot-android-studio files and the necessary changes (additions) are outlined in the following table:

File	Folder	Approx Line No	Content	Note
strings.xml	..\ot-android-studio\src\main\res**values-*nn***	29	<string name="burmese_language">Myanmar</string> **<string name="*new*_language">*new_country*</string>**	Add
strings.xml	..\ot-android-studio\src\main\res**values**	29	<string name="burmese_language">Myanmar</string> **<string name="*new*_language">*new_country*</string>**	Add
strings.xml	..\ot-android-studio\src\main\res**values-*nn***	420	<string-array name="language_keys"> <item >Khmer</item> <item >Albania</item> <item >Myanmar</item> **<item >*newCountry*</item>** <item >Default</item> </string-array>	Add (in new language)

File	Folder	Approx Line No	Content	Note
strings.xml	..\ot-android-studio\src\main\res**values**	420	<string-array name="language_keys"> <item >Khmer</item> <item >Albania</item> <item >Myanmar</item> **<item >*newCountry*</item>** <item >Default</item> </string-array>	Add (in new language)
strings.xml	..\ot-android-studio\src\main\res**values-*nn***	427	<string-array name="language_values"> <item >khmer_language</item> <item >albanian_language</item> <item >burmese_language</item> **<item >*newLanguage*_language</item>** <item >default_language</item> </string-array>	Add
strings.xml	..\ot-android-studio\src\main\res**values**	427	<string-array name="language_values"> <item >khmer_language</item> <item >albanian_language</item> <item >burmese_language</item> **<item >*newLanguage*_language</item>** <item >default_language</item> </string-array>	Add
OpenTenure.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	90	public static final String burmese_language = "burmese_language"; **public static final String *new*_language = "*new*_language";** public static final String tiles_provider = "tiles_provider";	Add
OpenTenure.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	126	OpenTenureApplication.*getInstance*().setBurmese(language.equals(OpenTenure.***burmese_language***)); **OpenTenureApplication.*getInstance*().set*newLanguage*(language.equals(OpenTenure.*newLanguage_language*));**	Add

OpenTenure.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	133	} else if (OpenTenureApplication.getInstance().isBurmese()) {
			setLocale(context, new Locale("my"));
			} else if (OpenTenureApplication.getInstance().is*NewLanguage*()) {
			setLocale(context, new Locale("*nn*"));
			} else {
			setLocale(context, Locale.getDefault());
				Add

OpenTenureApplication.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	85	private static final String _BURMESE_LOCALIZATION = "my-MM"; **private static final String *_NEWLANGUAGE*_LOCALIZATION = "*nn-NN*";** private String localization;	Add
OpenTenureApplication.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	91	private boolean burmese = false; **private boolean *newLanguage* = false;** private static boolean loggedin;	Add

File	Folder	Approx Line No	Content	Note

OpenTenureApplication.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	559	public boolean isBurmese() {
			return burmese;
			}
			public void setBurmese(boolean burmese) {
			this.burmese = burmese;
			}
			public boolean is*NewLanguage*() {
			return *newLanguage*;
			}
			public void set*NewLanguage*(boolean *newLanguage*) {
			this.*newLanguage* = *newLanguage*;
			}
			public void setLocalization(Locale locale) {
				Add

OpenTenureApplication.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	583	} else if (isBurmese()) {
			localization = OpenTenureApplication._BURMESE_LOCALIZATION;
			} else if (is*NewLanguage*()) {
			localization = OpenTenureApplication.*_NEWLANGUAGE*_LOCALIZATION;
			} else if (locale.getLanguage().toLowerCase(locale).equals("ar")) {
			localization = _ARABIC_LOCALIZATION;
				Add

OpenTenure Preferences Migrator.java	..\ot-android-studio\opentenure\src\main\java\org\fao\sola\clients\android\opentenure	60	if(preferences.getBoolean(OpenTenure.khmer_language, false)){
			language = OpenTenure.khmer_language;
			}
			if(preferences.getBoolean(OpenTenure.*newLanguage*_language, false)){
			language = OpenTenure.*newLanguage*_language;
			}
			SharedPreferences.Editor editor = preferences.edit();
				Add

Figure 57 – Open Tenure Localization code changes

13.7 Re-compile Software

Once the translated material has been incorporated into the source code of all the affected software application packages (Community Server, Web Admin, Open Tenure Android and Open Tenure iOS), then each of these applications needs to be re-built (compiled), deployed and tested, before they are made available to users.

13.7.1 SOLA Community Server & Web Admin

1. Use Netbeans IDE to **Clean&Build** the (OpenTenure-code) Main CS POM project
   Collect the newly generated deployable file so that you can do a fresh installation and thoroughly test this new version that includes the new language.
   This deployable file is:
   - ..\code\services\services-ear\target**sola-services-ear.ear**

13.7.2 Open Tenure Android Studio

1. Run **Android Studio** and click on the link to "open an existing Android project" link and then navigate to the ..\ot-android-studio\openTenure folder and select the **ot-android-studio.iml**
2. Click on menu option Build-Clean followed by Build-Rebuild
3. Find the generate **openTenure-debug.apk** file in the ..\ot-android-studio\openTenure\build\outputs\debug folder and transfer this file to an android device for testing.

13.7.3 Open Tenure iOS Xcode

1. Run **Xcode** (on an Apple Mac computer) and click on the File – Open menu option and then navigate to the ..\ot-ios folder and select the **Open Tenure.xcodeproj** file
2. Click on **Product – Build for – Testing** menu option and then **Product – Build** to build.

13.7.4 Publishing Open Tenure Android (& iOS)

The final stages of localization, following successful testing of the new localized versions of all associated SOLA software packages (Community Server, Web Admin, Open Tenure Android & (maybe) Open Tenure iOS) are:

1. to "push" the software changes back to the SOLA Github repositories;
2. to compile a "signed" version of Open Tenure Android (& iOS);
3. to publish the "signed" localized version of Open Tenure Android on Google Play Store (and likewise for Open Tenure iOS on the Apple App Store). In both cases this requires the person publishing software to have a developer account with the publishing store and in the case of the Apple App Store for the software to be moderated before it is published.
   For these reasons, if it is practical, contact should be made with the original publishing developers (contact details on Play Store / App Store) to see if they are willing to assist.

14. Software Changes

Although the SOLA Community Server and Open Tenure software applications are open source software it is strongly recommended that only experienced Java programmers attempt making any changes to the SOLA software.
The IDE used are:

1. Netbeans IDE 8.2 for the SOLA Community Server & Web Admin;
2. Android Studio for Open Tenure Android; and
3. Xcode for Open Tenure iOS.

Source code for the SOLA Community Server and associated applications is available from a series of Github repositories. The super github repositories for each of the SOLA applications are:

1. OpenTenure/code [SOLA Community Server & Open Tenure Android]
2. SOLA-WA-FAO/code [SOLA Web Admin]
3. Open Tenure/ot-ios [Open Tenure iOS]

14.1 Software Release Management Process

When you make software changes that need to be incorporated back into github repositories it is important that you follow this process:

1. Releases are identified by a 5 character code - YYMM<sequential alpha> eg first release in November 2019 would be 1911a, the second release 1911b
2. Check the Github super repository website of the SOLA application you have made changes to (eg https://github.com/SOLA-FAO/code) to see if there have been any other software releases in this month. Based on that, determine what your release code will be based on the rule provided above.
3. While at the SOLA super repository website, update the Issues log to note that the issue was resolved and reference the resolution of the issue with the new release code. Also take note of the IDs of resolved issues as recorded in the Issues Log
4. As part of the git “push” process (by way of a git or gits command or using the Git Desktop app) identify the release code and the issues resolved and/or a brief description of the changes (eg “Albanian language localization”).

14.2 Hardening Payara Server

For a production deployment it is important to ensure that Payara Server is adequately hardened to reduce the potential avenues an attacker could use to compromise the system. This page references some basic hardening tasks that should be performed on Payara Server. Note that Payara Server is just one piece of the security puzzle. For a secure system, a Threat Assessment should be undertaken to identify the most likely areas of attack and the appropriate counter measures put in place. Areas of security risk will likely encompass the network, hardware and operating systems, backup procedures as well as non-technical aspects such as physical server access control.
Things you should do

1. Run Payara Server under a non root (admin) user account. If you have installed Payara Server as root, you can change the owner to a new account by stopping Payara Server and granting the new user account ownership rights to the payara5 and all child folders. On Linux it will also be necessary to ensure all HTTP port listeners are configured for ports above 1024 (e.g. 8080 instead of 80 and 8181 instead of 443).
2. Change the admin password for the Payara Server Admin Console to ensure this is a strong password (min 12 characters with letters, numbers and symbols)
3. Secure the Administration Console by adding a password (current logon credentials are admin/<blank>)
   $ cd /payara5/bin
   $ sudo ./asadmin change-admin-password
   $ sudo enable-secure-admin

$ sudo ./asadmin restart-domain domain1

4. Confirm Payara Server is using the **-server** JVM option instead of the **-client** JVM with appropriate memory settings
5. Clear the product name setting
6. Obfuscate Payara Server **Powered By** details.
7. Use Password Aliasing to encrypt the database user and password used to connect to the SOLA Database
8. Update the Payara Server Keystores

References that explain the items above in more detail include;

Securing your Payara Hardening Guide

Java Key and Certificate Management Tool

14.2.1 Updating the Payara Server Keystores

Another very important element of hardening Payara Server and SOLA Community Server is to update the keystores used to secure the SOLA web service communications. Payara has two password encrypted keystores (keystore.jks and cacerts.jks) to manage the digital certificates it uses for SSL, encryption, authentication and related security services. The source code for SOLA includes the default Payara Server keystores updated with the Development Default certificates used by WSIT and Metro. These keystores along with the passwords used to encrypt them are public information and should be treated as compromised. **They are not suitable for production deployments of SOLA**. The following sections describe how to update the keystores with unique certificates and change the passwords protecting the keystores. Note that care should be taken to ensure the updated keystores and passwords generated by this process do not become public information.

Modifying the SOLA Source and Build

The first step is to update the SOLA source code to make it easier to hide private details such as the updated keystores and the keystore passwords.

NOTE: *The changes described in this section were made to the SOLA configuration in 2013. They are listed here for reference only. If you are creating new WIST digital certificates, proceed to the Change the Payara Master Password section below.*

- Add ***.jks** to the .../code/services/boundary/.gitignore file to ensure Git does not push the updated keystores to a public repository. There is no need to remove the existing keystore.jks and cacerts.jks - they are already compromised.
- Add the following resource filtering section to the pom.xml of the Boundary Web Services project

```
<resources>
<!-- Only filter the WSIT XML files. This will ensure the binary jks
 file is not corrupted by the filtering process. -->
  <resource>
    <!-- Sets filtering to true on the WSIT XML files -->
    <directory>src/main/java/META-INF</directory>
    <filtering>true</filtering>
    <includes>
      <include>**/*.xml</include>
    </includes>
  </resource>
  <resource>
    <!-- Sets filtering to false on all other files -->
    <directory>src/main/java/META-INF</directory>
    <filtering>false</filtering>
    <excludes>
      <exclude>**/*.xml</exclude>
    </excludes>
  </resource>
</resources>
```

- Add or update the properties section in the pom.xml of the Boundary Web Services project with the following. These properties relate to the default keystores but they can be overridden using the Maven settings.xml file.

```
<properties>
  <wsit.keystore.storepass>changeit</wsit.keystore.storepass>
  <wsit.keystore.peeralias>xws-security-server</wsit.keystore.peeralias>
  <wsit.keystore.location>keystore.jks</wsit.keystore.location>
</properties>
```

- Replace the sc:Keystore element in each WSIT configuration XML file in the META-INF source code package of the Boundary Web Services project to read as follows. Note the FileStreaming file does not require an update.

```
<sc:KeyStore wspp:visibility="private" location="${wsit.keystore.location}" type="JKS" storepass="${wsit.keystore.storepass}"
alias="${wsit.keystore.peeralias}"/>
```

- Add the following resource filtering section to the pom.xml of the Boundary Web Services Clients project

```
<resources>
<!-- Only filter the WSIT XML files. This will ensure the binary jks
 file is not corrupted by the filtering process. -->
  <resource>
    <!-- Sets filtering to true on the WSIT XML files -->
    <directory>src/main/resources</directory>
    <filtering>true</filtering>
    <includes>
      <include>**/*.xml</include>
    </includes>
  </resource>
  <resource>
    <!-- Sets filtering to false on all other files -->
    <directory>src/main/java/META-INF</directory>
    <filtering>false</filtering>
    <excludes>
      <exclude>**/*.xml</exclude>
    </excludes>
  </resource>
</resources>
```

- Add or update the properties section in the pom.xml of the Boundary Web Service Clients project with the following.

```
<properties>
  <wsit.truststore.storepass>changeit</wsit.truststore.storepass>
  <wsit.truststore.peeralias>xws-security-server</wsit.truststore.peeralias>
  <wsit.truststore.location>cacerts.jks</wsit.truststore.location>
</properties>
```

- Replace the sc:TrustStore element in each WSIT configuration XML file in the META-INF resources package of the Boundary Web Service Clients project to read as follows.

```
 <sc:TrustStore wspp:visibility="private" type="JKS" storepass="${wsit.truststore.storepass}"
peeralias="${wsit.truststore.peeralias}" location="${wsit.truststore.location}"/>
```

Change the Payara Server Master Password

The Payara Server Master Password is used to decrypt the Payara Server keystores allowing Payara Server to access the digital certificates they contain. By default, this password is *changeit* and you should do exactly that. Be aware that if you complete this process on your development instance of Payara Server so that you can create and test the build package for deployment, you will also need to synchronize the Payara Server Master Password on the Payara Server production instance to match the password used to generate the updated keystores.

To update the Payara Server Master Password, stop your instance of Payara Server and use the change-master-password subcommand of asadmin from a command prompt. e.g.

```
<Payara Install Dir>\bin\asadmin change-master-password --savemasterpassword=true --domaindir <SOLA Payara Domain Dir>
```

You may need to run the change-master-password command as an Administrator. You should also explicitly set the location of the domain (domaindir) to ensure the change-master-password updates the passwords on the default Payara Server keystores.

Configure your Maven settings.xml

settings.xml is used by Maven to provide override property values and User Settings (located in the users .m2 folder) are often used to limit exposure of passwords and other secure information. Add or update the settings.xml file in your .m2 folder (will be in your Windows user profile or your Linux home directory) to be similar to the following. Replace YourMasterPassword with the Payara Server Master Password set in the following section.

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <profiles>
    <profile>
      <id>prod</id>
      <activation>
        <activeByDefault>false</activeByDefault>
      </activation>
      <properties>
        <wsit.truststore.storepass><!--YourMasterPassword--></wsit.truststore.storepass>
        <wsit.truststore.peeralias>sola-wsit-cert</wsit.truststore.peeralias>
        <wsit.truststore.location>sola-truststore.jks</wsit.truststore.location>
        <wsit.keystore.storepass><!--YourMasterPassword--></wsit.keystore.storepass>
        <wsit.keystore.peeralias>sola-wsit-cert</wsit.keystore.peeralias>
        <wsit.keystore.location>sola-keystore.jks</wsit.keystore.location>
      </properties>
    </profile>
  </profiles>
</settings>
```

Generate new Certificates

To generate new certificates for Payara requires using the Java Keytool. The following Windows CMD script can be used to update the default Payara Server keystores and generate two new keystores suitable for SOLA (sola-keystore.jks and sola-truststore.jks). Simply update the **keytool** and **Payara_config_dir** variables to match your locations. If the script fails with an error indicating the password for the keystores is incorrect, make sure you have correctly set the domaindir in the change-master-password command above. For example, if the domain config dir is C:\Work\source\Payara\sola\config\, the domain dir will be C:\Work\source\Payara

```
@echo off
REM 27 FEB 2013
REM Windows CMD script that generates new self-signed certificates using the
REM Java 7 Keytool and configures the default keystores in Payara so
REM the WSIT certificate can be used for SOLA Web Services authentication.
REM
REM This script should be used to generate a new WSIT certificate for
REM production deployments of SOLA. The keystores and certificates
REM packaged with the generic SOLA are publicly available and should
REM be treated as compromised. THEY ARE NOT SUITABLE FOR PRODUCTION
REM DEPLOYMENTS OF SOLA. Care should be taken to ensure the keystores
REM and passwords generated for a production deployment do not become
REM public information.
REM
REM This script relies on the Payara Master Password. The default master
REM password for Payara is changeit. You should change this password
REM on the production instance of Payara using the Payara asadmin
REM change-master-password sub-command.
REM
REM When running this script on a developer instance of Payara,
REM ensure the master password matches the Production master password
REM otherwise the keystores created will not be accessible by the
REM Production Payara instance.

REM Location of the Java Keytool. Note that the Java 7(+) Keytool must be
```

```
REM used otherwise the certificate generated will not be suitable for use
REM by Metro/WSIT. Update as required.
set keytool="C:\Program Files\Java\jre7\bin\keytool"
REM Location of the SOLA domain config folder. Update as required.
set Payara_config_dir=C:\Work\source\Payara\sola\config\
set master_password=?

set /p master_password= Enter your Payara Master Password [%master_password%] :

echo
echo Updating the Payara keystores. Please wait...
echo

REM Remove the default s1as and Payara-instance certificates from the
REM Payara keystore.jks and the cacerts.jks keystore. As with the default
REM SOLA WSIT certs, these certs are public information and should be
REM considered compromised for any production deployment.
%keytool% -delete -alias s1as -keystore %Payara_config_dir%keystore.jks -storepass %master_password%
%keytool% -delete -alias Payara-instance -keystore %Payara_config_dir%keystore.jks -storepass %master_password%
%keytool% -delete -alias sola-wsit-cert -keystore %Payara_config_dir%keystore.jks -storepass %master_password%
%keytool% -delete -alias s1as -keystore %Payara_config_dir%cacerts.jks -storepass %master_password%
%keytool% -delete -alias Payara-instance -keystore %Payara_config_dir%cacerts.jks -storepass %master_password%

REM Generate three new certificates in the default Payara keystore. Two
REM that replace the s1as and Payara-instance certificates and a new one
REM to be used by SOLA WSIT authentication. Note that only one X509v3
REM certificate is required for Username Authentication with Symmetric Key -
REM the default WSIT authentication mechanism used by SOLA. If you are using
REM a different authentication mechanism such as Mutual Authentication, you
REM may need to generate more certificates.
REM
REM Notes:
REM 1) The certificates generated are valid for 10 years from the day of
REM    generation
REM 2) The keypass and storepass must match the Payara Master Password
REM    otherwise Payara will not be able to load the certificates from
REM    the keystore
REM
%keytool% -genkeypair -keysize 2048 -alias sola-wsit-cert -keyalg RSA -validity 3650 -dname "CN=sola-wsit-cert, O=SOLA" -
keypass %master_password% -storepass %master_password% -keystore %Payara_config_dir%keystore.jks
%keytool% -genkeypair -keysize 2048 -alias s1as -keyalg RSA -validity 3650 -dname "CN=s1as, O=SOLA" -keypass
%master_password% -storepass %master_password% -keystore %Payara_config_dir%keystore.jks
%keytool% -genkeypair -keysize 2048 -alias Payara-instance -keyalg RSA -validity 3650 -dname "CN=Payara-instance, O=SOLA" -
keypass %master_password% -storepass %master_password% -keystore %Payara_config_dir%keystore.jks


REM Export the three new cerificates so they can be loaded into other
REM keystores.
%keytool% -exportcert -alias sola-wsit-cert -keystore %Payara_config_dir%keystore.jks -storepass %master_password% -rfc -file
%Payara_config_dir%sola-wsit-cert.cer
%keytool% -exportcert -alias s1as -keystore %Payara_config_dir%keystore.jks -storepass %master_password% -rfc -file
%Payara_config_dir%s1as.cer
%keytool% -exportcert -alias Payara-instance -keystore %Payara_config_dir%keystore.jks -storepass %master_password% -rfc -file
%Payara_config_dir%Payara-instance.cer


REM Tidy up any sola keystores created from a previous run of this script.
DEL /Q  %Payara_config_dir%sola-truststore.jks
DEL /Q  %Payara_config_dir%sola-keystore.jks


REM Import the new certificates into the default Payara cacerts.jks
REM keystore and create a new keystore (sola-truststore.jks) to hold the
REM public key for sola-wsit-cert
echo
echo Enter Y to confirm the import of the certificates into the keystores
echo
%keytool% -importcert -alias sola-wsit-cert -keystore %Payara_config_dir%sola-truststore.jks -storepass %master_password% -file
%Payara_config_dir%sola-wsit-cert.cer
%keytool% -importcert -alias s1as -keystore %Payara_config_dir%cacerts.jks -storepass %master_password%  -file
%Payara_config_dir%s1as.cer
%keytool% -importcert -alias Payara-instance -keystore %Payara_config_dir%cacerts.jks -storepass %master_password% -file
%Payara_config_dir%Payara-instance.cer
```

```
REM Delete the certificate files - they are no longer required.
DEL /Q %Payara_config_dir%sola-wsit-cert.cer
DEL /Q %Payara_config_dir%s1as.cer
DEL /Q %Payara_config_dir%Payara-instance.cer


REM Create a copy of the Payara keystore.jks and remove any certs that
REM are not required. The sola-keystore.jks will be used by the SOLA Web
REM Services and does not need the additional certificate information.
COPY /Y /B %Payara_config_dir%keystore.jks %Payara_config_dir%sola-keystore.jks
%keytool% -delete -alias s1as -keystore %Payara_config_dir%sola-keystore.jks -storepass %master_password%
%keytool% -delete -alias Payara-instance -keystore %Payara_config_dir%sola-keystore.jks -storepass %master_password%
%keytool% -delete -alias wssip -keystore %Payara_config_dir%sola-keystore.jks -storepass %master_password%
%keytool% -delete -alias xws-security-client -keystore %Payara_config_dir%sola-keystore.jks -storepass %master_password%
%keytool% -delete -alias xws-security-server -keystore %Payara_config_dir%sola-keystore.jks -storepass %master_password%

echo
echo All Done!
echo
pause
```

Build and Deploy with the Updated Keystores

You will now have 4 keystores in the Payara Server config directory - keystore.jks, cacerts.jks, sola-keystore.jks and sola-truststore.jks

- Copy the sola-keystore.jks to the Boundary Web Services META-INF source code package
- Copy the sola-truststore.jks to the Boundary Web Service Clients META-INFO resource package
- Build the Main POM using the prod configuration setup in settings.xml. You may need to include some extra build parameters for this configuration such as skip tests and Maven java memory settings. See the default build configuration for details.
- Deploy and test the deployment packages as per usual.

When you are ready to deploy the updated build to the Production Payara Server instance

- Stop the production instance
- Update the Payara Server Master Password on the production instance to match the password used to encrypt the keystores
- Copy the updated keystore.jks and cacerts.jks to the Domain config directory on the Production Payara Server instance. Be careful to ensure these keystores are not exposed more than is absolutely necessary. Also be sure to use the correct config directory - the keystore.jks and cacerts.jks should replace files that already exist. If they do not exist its because you are in the Payara Server config directory instead of the Domain config directory.
- Deploy the new SOLA Services EAR and Web Start WAR

If you want to return to using the development keystores, revert the Payara Server Master password on your developer Payara instance to *changeit* and rebuild SOLA using the default build configuration. Be aware that once the Production Payara Server instance has been updated, you will need to use the prod build configuration whenever you create a build for production. You also need to ensure the sola-keystore.jks and sola-truststore.jks are in the appropriate directories as they will be excluded from source control.

15. Community Server Deployment Scenarios

15.1 Single Community Server hosted on a Cloud Server

In situations where Open Tenure recording is beginning as a small scale, potentially pilot implementation but with the possibility of this initial implementation growing into a very comprehensive implementation, a cloud based hosting arrangement could be an excellent option. This is particularly the case if there is no capital expenditure budget, there are no or only limited trained staff to maintain a local computer system or there is no suitably secured office space or stable infrastructure (including power supply).

In such a situation, a monthly fee of less than USD 100 will give the community access to a cloud based server that can host the Community Server. Many cloud servers provide a VNC Tunnel & VNC Link as part of the cloud server admin console and so no extra software is required by the Community Technologist and the installation of Community Server is virtually the same as for a locally based Community Server assuming there is good internet connectivity.

A cloud based server provides maximum flexibility to upgrade the server (or close down) the cloud-based server as the community's needs change over time.

Typically the method of payment is by monthly payments made using a credit card.

15.2 Local Computer with limited internet access

This scenario becomes relevant where :

- the internet access is weak or unaffordable
- the community could have difficulties paying the monthly cloud server fees and
- the community has a desktop or laptop computer and the skills to keep it running and secure.

15.2.1 Required Equipment

The absolute requirement for this scenario is a relatively new computer (ideally a laptop to cope with any interruptions in power supply) with a minimum specification of :

- 4Gb of RAM
- 150 GB of free HDD
- a modern version of Operating System (eg Windows 7 or later)
- 802.11 wireless network connectivity that is compatible with the Wireless Access Point (defined below)
- at least one USB port
- a mouse
- connected to Power Surge protection device and ideally a UPS
- Loaded with:
    - antivirus software and a current update subscription
    - a versatile text editor (ie in Windows Notepad++)
    - MS Office or Open Office

15.2.2 Associated Equipment

1. A Wireless Access Point (802.11 abgn wireless network connectivity depending on which wireless standard is used by the mobile devices that will connect) and that allows for more connections than the likely maximum number of Open Tenure tablets being used in conjunction with the Community Server at any one time. A Wireless Access Point device powered by a battery or through USB connection would be an advantage if power continuity is an issue.
2. A UPS to provide surge protection and at least 30 minutes operation of the server.
3. Printer – if certificates are to be printed or copies of records will be supplied to community members (do not connect a laser printer to the UPS).

15.2.3 Community Server Setup

In this scenario, the most common situation is that the host computer will have a Windows operating system. If that is the case, a regular Windows Community Setup setup (such as is described in Section 2) should be undertaken.

The following additional steps are needed to incorporate the Wireless Access Point and a fixed IP address for the Community Server:

1) Set a high value IP address for the community server on the Wireless Access Point LAN.
    - From the Community Server computer, ensure the wireless connectivity on your computer is turned on and connect to the Access Point.
    - Through your Wireless Access Point Admin Console (usually accessible through your web browser using a url of http://192.168.0.1 or http://192.168.1.1), go to the LAN settings form and note the range of IP values that will be automatically assigned to computers and devices connecting to this wireless LAN (eg 192.168.0.50 to 192.168.0.199)
    - find the DHCP "Fixed Mapping" settings, identify the Community Server computer (usually using the MAC Address) and set a higher IP address that is easy to remember (eg 192.168.0.100)
    - restart the Access Point and re-connect Community Server computer to the Access Point and through the Admin Console verify this IP address has been assigned
2) On the Community Server computer open a Command window and confirm the IP address with the **ipconfig** command (or **hostName -I** command if computer is running Ubuntu)
3) In SOLA Web Admin confirm or modify the following as "localhost" url setting values:
    - **report_server_url** – http://localhost:8090/ (**System settings**)
    - **WMS URL** (WMS tab) http://localhost:8085/geoserver/opentenure/wms (**Map – Layers** menu option) assuming this layer has been generated from Geoserver domain of local Payara 5

15.3 Local Area Network with Internet Access

This scenario may arise when an organisation allied to a community offers to host Community Server on its existing server and to give the community access to their network for Open Tenure tenure recording work.

The community would gain access to such an instance of Community Server in one of two ways:

1) By way of **wireless connectivity to the organisation's network** and hence to the instance of Community Server hosted on that network (as in the Laptop/Desktop Scenario described in Section 15.2) with the possible added benefits, if the organisation's network included an internet service of being able to utilise web based map imagery (to augment the Community Server map display and to do Open Tenure imagery downloads) if the organisation's network is connected to the internet.
2) If the organisation's **network was web facing** and the organisation is willing for the Community Server to be accessible from the web this would provide a similar setup as to when Community Server is hosted on a Cloud Server. An implication of this option Open Tenure mobile devices would need to be fitted with SIM cards to allow for an internet connection to the Community Server, unless the hosting organisation had a local office within the community's area and were prepared to also provide wireless connectivity to the community

Community Server installation would follow the same steps as has been described previously (in Sections 2 and possibly 3) with the only differences being that in the case of Windows based servers, there would be a different operating system (Windows Server 2019 or 2016). As well, and more significantly, the installation would need to be completed within the existing system

and network roles and privileges. Knowledge of, and the ability to modify, those settings including activating certain system features (such as Windows .NET Framework) will be with the organisation's system and network administrator. It is essential that this person undertake the Community Server installation.

15.4 Multiple Community Server Deployments on Single Server

It is possible for one physical server to host multiple implementations of Community Server.

The most obvious approach to this situation is to create a new Payara domain and deploy the new Community Server on it. This new Payara domain would need to be configured such as has been described in previous sections of this guide. The disadvantages of this approach of multiple Payara domains is that more system resources are consumed and importantly it is not possible for all the different Payara domains to use the same HTTP port on the same server.

There is another approach involving the creation of virtual servers running under the same Payara domain allowing existing settings such as JDBC resources, security realms and other configuration settings to be re-used for every virtual Community Server without the need to create them again. These virtual servers use less resources, can re-use (or vary) the HTTP and HTTPS ports and allow the referencing of different host names (internet domains) thus making this the preferred approach for the rapid deployment of multiple Community Server instances.

Although the virtual server approach is preferred, it has two limitations that need to be considered when planning the establishment of new virtual server setup.

1. there is a need for a single wildcard domain certificate so that you are able to name your multiple Community Servers under the same domain (e.g. cs1.**exampledomain.org**, cs2**.exampledomain.org**, cs3.**exampledomain.org**, etc).
2. for each Community Server, their Payara JDCB pool and security realm definitions need to be configured so that they reference their own database. This requires that each Community Server deployment package package is separately compiled with the specific settings for the JDBC connection and security realm

The steps to implement multiple instances of Community Server using the virtual server approach are:

15.4.1 Get and install domain (wildcard) SSL certificate

Install and configure wildcard SSL certificate with the following steps:

16) On your Internet Service Provider (ISP) website, log on to your account and register your domain name (if you do not have any) and setup DNS records to point to your server IP address.
17) Open command line and navigate to the bin folder of Java JDK (e.g. C:\Program Files\Java\jdk1.8.0_231\bin).
18) Generate new entry in **keystore.jks** with information of your domain:

    keytool -keysize 2048 -genkey -alias *www.yourdomain.com* -keyalg RSA –dname "CN=*www.domain.com*,O=*company*,L=*city*,S=*State*,C=*Country*" -keystore keystore.jks
    Enter password of your keystore when asked. Please note that the GoDaddy domain registry requires at least 2048 bits keysize. **CN** is your sites domain name, **O** is your company name, **L** is the city, **C** is the 2 character country code. There are more options you could specify if you want. But these are enough. **alias** is the key you will use to refer this certificate.

19) Create the request file for submitting to Godaddy:
    keytool -certreq -alias *www.yourdomain.com* -keystore keystore.jks -file cert_req.csr
    Enter password of you keystore when asked.
20) Open **cert_req.csr** file in the text editor and copy all content between and including the following text:
    —–BEGIN NEW CERTIFICATE REQUEST—–

—–END NEW CERTIFICATE REQUEST—–

21) Go to GoDaddy domain registry website and enter copied content into CRS text area, when requesting certificate:

Figure 58 – SSL Certificate Installation

22) After finishing with certificate request you will receive approval email to your address.
23) After approval you need to download a zip file which contains all certificates you need. During this step you will be asked for which server you are downloading certificates. You could select **Other** because Payara is not listed. Your download will contain 4 files:
    a. gd_bundle.crt
    b. gd_cross_intermediate.crt
    c. gd_intermediate.crt
    d. *yourdomain.com*.crt

    First 3 of them are certificates belonging to your domain registry (such as godaddy.com). They are used to verify your domain's certificate. They may already be contained in your **cacerts.jks** but there is no harm importing them in your **keystore.jks**
24) Import these certificates to your **keystore.jsk** using following steps:
    keytool -import -alias root -keystore keystore.jks -trustcacerts -file gd_bundle.crt
    keytool -import -alias cross -keystore keystore.jks -trustcacerts -file gd_cross_intermediate.crt
    keytool -import -alias intermed -keystore keystore.jks -trustcacerts -file gd_intermed.crt
    keytool -import -alias www.yourdomain.com -keystore keystore.jks -trustcacerts -file yourdomain.com.crt
25) Restart Payara
26) Open Payara administration console and go to **Configurations** => **server-config** => **HTTP Service** => **Http Listeners** and select **http-listener-2**
27) On the right side of the page select **SSL** tab on top and enter your certificate alias (used in the previous steps) into the **Certificate NickName** field.
28) If you want your server to use standard SSL port (443), select **General** tab and enter **443** into the Port field.
29) Finally save settings by pressing **Save** button.
30) Restart Payara server and test https connection by going to https://your.domain.com. Check for certificate info in the address bar.

Note – Sometimes the Payara Admin Console gives an error when creating a new http listener and assigning a certificate alias to it. In such case, try the asadmin command line as follows:
asadmin> create-ssl --type http-listener --certname sampleCert http-listener-1

15.4.2 Create DNS record for new server

A DNS record for the new server is required so that you can create host names for the Community Server instances in terms of your main domain name. These host names are created under the main domain. For instance if your domain name is **mydomain.org**, and you have created and deployed a SSL certificate for this domain, you can create any number of host names for your

Community Servers, which will be deployed on the same Payara instance (e.g. **community.mydomain.org**, **newcommunity.mydomain.org**, etc).

DNS record creation depends on the specific Internet Service Provider (ISP) but in general it is a simple task. For instance with GoDaddy ISP you logon to your account and click on Domains – Manage

1) Click the domain name you want to use.
2) Click the **DNS Zone File** tab.
3) Click **Add Record**.
4) From the **Record type** list, select **A (Host)**.
5) Complete the following fields:
   Host Name — Enter the host name the A record links to. Type **@** to point the record directly to your domain name, including the **www**.
   Points to IP Address — Enter the IP address your domain name uses for this host record.
   TTL — Select how long the server should cache the information.

Click **Finish**, then click **Save Changes**. The new A record displays in the A (Host) section.

15.4.3 Create new Community Server database

New database can be created using scripts for generating the database giving it a name sola_<<community name>> (refer to Section 3.3.4).

Alternatively you can restore a database backup of another Community Server and then run the following sql script to clean the database of any claim data that is specific to the original community.

Using the PG Admin tool, select the **sola** database and then click on the **SQL** toolbar icon, copy the script in Annex 3 - SQL script to revert sola database to "new installation" state into the panel and click on the **Run** icon (arrow pointing to the right).

If you want to preserve the dynamic forms definitions from the existing Community Server database, do not execute the delete commands after "Dynamic forms structure" comment.

15.4.4 Create new JDBC connection pool and resource

Since every new Community Server has its own database (created in Section 15.4.3) (using the Payara Admin Console) create a JDBC connection to reference the newly created SOLA database (Refer to Section 5.1) In configuring the JDBC connection you give new names to the resource (Pool Name – sola_***community name*** and JNDI name – jdbc/sola_***community name*** and likewise reference the new database name (eg sola_***community name***).

15.4.5 Create new Security Realm

In Payara Admin Console create a new security realm (refer to Section 5.2) and make sure to give a new name to the realm (eg Sola_***community name***) and to reference the JDBC connection, created in the previous step.

15.4.6 Create new JavaMail Service

If the Community Server is going to use the email functionality create a new JavaMail Service using Payara Admin Console (refer to Section 9). Make sure you use a new name when creating this JavaMail service (eg mail/***sola_community name***.

15.4.7 Create new Virtual Server

To create a new virtual server and bind the DNS name (created and configured in Section 15.4.2) to it, follow these steps:

1) Open Payara administration console at http://localhost:4848 (or http://domainname.org:4848 or http://ipaddress:4848) and go to **Configurations => server-config => Virtual Servers.**
2) Click "**New**" button.
3) Enter short server name into "**Id**" field (e.g. name of community)
4) Enter full domain name into "**Hosts**" field (host name created before at step 2).
5) Select "**http-listner1**" and "**http-listner2**" in the "**Network Listeners**" field (use CTRL key to select both listeners).
6) Leave all other fields unchanged and click "**OK**" button to create the virtual server.

15.4.8 Deploy new Community Server

Although this deployment is similar to the standard deployment of the application, there are some are some differences so please follow these steps closely:

1) Open Payara administration console at http://localhost:4848 (or http://domainname.org:4848 or http://ipaddress:4848) and go to "**Applications"** and click **"Deploy"** button**.**
2) Browse and select Community Server deployment package (EAR file).
3) Enter a new application name to distinguish it from other Community Server instances eg SOLA Community Server - <<community name>>-<<version number>>.
4) In the "Virtual Servers" list select the virtual server you created in the previous step.
5) Click "**OK**" button to finish deployment.

Although the new Community Server is now deployed, further configuration is required.

You also need to change the configurations on the original instance of Community Server if it was deployed using the regular Community Server package.

15.4.9 Make changes to configuration files

Where the regular Community Server package has been deployed, two xml configuration files of the deployed application need to be modified to reference the JDBC connection and security realm created for use with the virtual server.

Assuming your Payara is deployed on "C:\" drive, the folder where you will find the file **mybatisConnectionConfig.xml** is
"C:\Payara5\glassfish\domains\domain1\applications*your_application*\META-INF".
Open this file from **META-INF** folder in a text editor and find the following line:

```
<property name='data_source' value='jdbc/sola'/>
```

Change value key from "**jdbc/sola**" to JDBC resource name, you created before for this application (eg jdbc/sola_***community name***). Save the file.

Now open **sun-application.xml** file from the same **META-INF** folder in a text editor and find the following lines:

```
<realm>
        SolaRealm
</realm>
```

Change "**SolaRealm**" value to the security realm name, you created for this application (eg Sola_***community name***Realm) and save changes to the file.

Please note that these files will be overwritten every time you re-deploy your Community Server application and you will need to make these changes again. One way of avoiding this is to make copies of the modified files and copy them over next time you re-deploy the Community Server application.

15.4.10 Disable and enable new Community Server application

After changing application configuration files, you have to restart application for changes to take effect. There is no need to restart the whole Payara server, it's enough to disable and enable your application. Follow these steps to do this:

1) Open Payara administration console at http://localhost:4848 (or http://domainname.org:4848 or http://ipaddress:4848) and go to "**Applications".**
2) Select checkbox on your application and click "**Disable**" button on the toolbar.
3) Wait for disable action to finish and select you application again. Click "**Enable**" button this time.

After you restart the Community Server application, you should be able to access this instance of Community Server through your web browser using the new url for this Community Server eg https://***community name.exampledomain.org***. Check that it is empty database with no claim records. A further check is to logon to the SOLA Web Admin application associated with the new instance of Community Server (eg https://***community name.exampledomain.org/***admin) and check the header of the page. It must contain IP address and database name.

Figure 59 – Community Server header details

15.4.11 Configure Community Server settings

After new Community Server has been deployed, further changes are required to prepare it for its new community:

1) If the database was created from a backup of another Community Server database, use SOLA Web Admin to remove users or to change passwords and roles.
2) Enrol new users (refer to Section 7
3) Define an area of interest for the new community (refer to Section 6.3).
4) Make a copy of JasperReport templates and adjust their settings. Community Server comes with a set of pre-defined report templates, which can be deployed through JasperReports Server. You can deploy the report templates as described in Section 4). Alternatively, you can copy over the existing set of report templates and adjust their data source to point to the database for the new Community Server. Follow the steps below to do it:
   (i) Login to JasperReports server admin console
   (ii) Go to **Data Sources** folder and create a new data source. Make sure to reference the JDBC connection created for this instance of Community Server (eg. **jdbc/*sola_community name***)
   (iii) Go to **Reports** folder, right click on it and select "**Add Folder**" submenu. Enter new folder name for your reports eg. "community_server/***community name***".
   (iv) Go to the existing folder containing reports you want to copy, select all reports inside (use SHIFT or CTRL key to select multiple) and click "**Copy**" button on the toolbar.
   (v) Go back to the folder you created and click on the "**Paste**" button to insert copied reports.
   (vi) Select first report in the list and click "**Edit**" button.

(vii) On the report settings select "**Data Source**" menu on the left and click "**Browse**" button under "**Select data source from repository**" field.
(viii) Select data source, you defined in step b (eg. **jdbc/*sola_community name***).
(ix) Click "**Submit**" button at the bottom to save changes.
(x) Repeat steps (vi)-(viii) for the other report templates.

5) Change reports settings. After you created or copied reports for your new CS installation, you have to adjust reports settings through SOLA Web Admin. Go to **Settings => System settings,** review and adjust the following settings, according to your setup:
   - claim_certificate_report_url (make sure to reference your folder)
   - enable-reports
   - report_server_pass
   - report_server_url
   - report_server_user
   - reports_folder_url (make sure to reference your folder)
6) Configure GeoServer for the new Community Server. In order to expose claims layer on the map, you have to create the claims layer in GeoServer that references the SOLA database (eg. sola_***community name***), for this new instance of Community Server (refer to Section **Error! Reference source not found.**). If you have other layers to publish on the map, add them in GeoServer as well (refer to Section 5.6.6).
7) Modify map layers configuration. Using SOLA Web Admin go to **Map => Layers** menu. Modify the "**claims-orthophoto**" layer to reference claims layer for the new Community Server published on GeoServer. Usually this will only require a changed layer name on the "**WMS**" tab of the layer's properties. Refer to Section 5.6.5 for publishing an imagery layer (if available).
8) Modify mailer settings. If you configured a new JavaMail session for your Community Server for a regular Community Server deployment, modifications will need to be made using SOLA Web Admin **Settings => System settings** menu option. Review and modify the following settings:
   - email-admin-address
   - email-admin-name
   - email-body-format
   - email-enable-email-service
   - email-mailer-jndi-name (make sure to reference your JavaMail session)

   You may also need to adjust message templates, used to send various emails to reflect the name of the new community to be served by this new instance of Community Server. Check those templates start with "**email-msg-**".
9) Adjust other system settings. You may need to modify other system settings as well such as the "**community-name**" setting. Use SOLA Web Admin to review and make these changes.

As a final step to these changes, remember to use SOLA Web Admin to reset the cache using the **Cache => Cache reset** menu option.

15.4.12 Deploying new versions of Community Server

When a new version of the Community Server software needs to be deployed follow the following steps:

1) Check to see if there are any associated database scripts to run. If there are scripts, run PG Admin, select the community's sola database and open a SQL window and run the script
2) Run the Payara Admin Console on the server (http://localhost:4848 or http://domainname.org:4848 or http://ipaddress:4848), select Applications, and the

community's implementation and then redeploy with the new Community Server ear file

3) Create a suitably modified .sh file (eg update.sh) and save it in a folder (say home/ubuntu/temp

```
echo "Updating bago..."
sudo cp $TMP/bago/mybatisConnectionConfig.xml $GF/bago-cs/META-INF/mybatisConnectionConfig.xml
sudo cp $TMP/bago/sun-application.xml $GF/bago-cs/META-INF/sun-application.xml
echo "DONE"
```

4) From a command line, type **~/tmp/update.sh** to run the update.sh file.
5) Return to the Payara Admin Console and the Applications page. Select redeployed Community Server (tick checkbox) and click "Disable" button.
6) Select redeployed Community Server (tick checkbox) and click "Enable" button

If you want to preserve the dynamic forms definitions from the existing Community Server database, do not execute the delete commands after "Dynamic forms structure" comment.

1) In Payara Admin Console (http://localhost:4848), restart server. When server has been restarted, from the Payara Admin Console open the Resources – JDBC – JDBC Connection Pools – sola form and click on Ping button to confirm sola database is connected. Also Open the Applications form and confirm that Community Server has been deployed and Geoserver.
2) Launch Geoserver (http://localhost:8080/geoserver/web (or http://domainname.org:8080/geoserver/web or http://ipaddress:8080/geoserver/web)) and in the Preview Layers confirm that all required layers (including Claims and imagery published on local (hardware) server are viewable.
3) If required, install and configure JasperReports (refer to Section 10)
4) If required, complete Email configuration (refer to Section 9)

15.5 Intel ConnectStick (computer on a dongle) with limited internet access

This scenario becomes relevant where :

- the internet access is weak or unaffordable
- the community could have difficulties paying the monthly cloud server fees
- the community does not have a desktop or lapto or the right conditions to operate such computers (eg no mains power).

15.5.1 Required Equipment

The absolute requirement for this scenario is an Intel Connect Stick STK2M364CC (without OS) (Available from Amazon : https://www.amazon.com/Intel-Corp-BLKSTK2M364CC-Compute-STK2m364CC/dp/B01DQ3RLPO). This device has built-in wifi connectivity which can be configured as a hotspot from which mobile devices running Open Tenure can connect.

To operate SOLA Community Server on a Connect Stick ideally you need the following additional items of equipment:

- 1 or 2 Quick Charge 3.0 Powerbank(s) with capacity of at least 12000 mAH using QC USB Output DC 5V-3A
- Quick Charge cable from Powerbank USB to USB-C
- Portable Solar Charger with USB-A output port. 21 Watt or better and ideally with meter to measure current being generated (to charge powerbank(s))
- microSD card to receive backups (8 – 128 Gb)

To setup SOLA Community Server and its pre-requisites initially you need:

- access to a TV or monitor which takes HDMI input
- USB hub

- Mouse with USB connector
- Keyboard with USB connector

Subsequent setup steps can be done with a VNC connection.

15.5.2 Associated Equipment

- Printer – if certificates are to be printed or copies of records will be supplied to community members.

15.5.3 Community Server Installation

Because the recommended model (STK2M364CC) of the Intel Connect Stick is supplied without any operating system, it is necessary to install Ubuntu 18.04 from an iso file first. Then the standard Linux installation described in Section 3 can be followed. The Linux Docker Desktop installation approach, described in Section **Error! Reference source not found.**, is part well suited in so far as simplicity and time taken.

++++++++++++++++++++++++++++++++

15.5.4 Downloads to install Ubuntu 18.04 LTS

There are two downloads needed before installation:

i. The latest Ubuntu release 64-bit desktop ISO from http://releases.ubuntu.com (ubuntu-18.04-desktop-amd64.iso was the current version in May 2018). Approximately 2Gb and dated 28 April 2018

ii. The latest Linuxium ISO respin scripts from https://drive.google.com/file/d/0B99O3A0dDe67S053UE8zN3NwM2c/edit Please note that these scripts are being frequently refined and some of the syntax may change. The following description of how to use these scripts are based on the April 2018 Linuxium blog

15.5.5 Respin an ISO file for Ubuntu 18.04 LTS

The Linuxium scripts need to be run on a Ubuntu platform (confirmed as working on version 18.04). To download the Ubuntu ISO file and the Linuxium respin scripts:

1 open a Terminal window and perform the following commands:

```
$ sudo apt update
$ sudo apt install squashfs-tools
$ sudo apt install xorriso
$ cd ~/Downloads
$ sudo isorespin.sh -i ubuntu-18.04-desktop-amd64.iso --atom
```

2 the respun ISO file will have been created in the Downloads folder and be called **linuxium-atom-ubuntu-18.04-desktop-amd64.iso.**

15.5.6 Create a Bootable microSD card/USB for Ubuntu 18.04 installation

1 insert a (ideally 8Gb or more) microSD (probably using a SD or USB adaptor) into the same (Ubuntu OS) computer as used to create respun Ubuntu ISO file.

2 Determine device name of microSD (**NOT the partion label**)

```
Disk /dev/sda: 3.9 GiB, 4194304000 bytes, 8192000 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x14a883b1

Device     Boot Start     End Sectors Size Id Type
/dev/sda1  *        0 4071999 4072000   2G  0 Empty
/dev/sda2         484    6557    6074   3M ef EFI (FAT-12/16/32)


Disk /dev/mmcblk1: 14.9 GiB, 16012804096 bytes, 31275008 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x14a883b1

Device         Boot Start     End Sectors Size Id Type
/dev/mmcblk1p1 *        0 4071999 4072000   2G  0 Empty
/dev/mmcblk1p2        484    6557    6074   3M ef EFI (FAT-12/16/32)
```

$ sudo fdisk –l

In the example (above) the two attached devices are a USB disk (named **/dev/sda)** and a microSD card directly inserted into the slot of an Intel Connect Stick (named **/dev/mmcblk1).**

IGNORE AND DO NOT USE THE PARTITION LABELS /dev/sda1 /dev/sda2 /dev/mmcblkp1 and /dev/mmcblkp2

3 Based on the device name determined in the previous step perform the following command in a Terminal window. **Please take care as this command can corrupt the host platform if not applied correctly**

$ sudo dd if=~/Downloads/linuxium-atom-ubuntu-18.04-desktop-amd64.iso of=/dev/**mmcblk1**

(if you are using a microSD card in the microSD slot on the Intel Connect Stick)

15.5.7 Ubuntu Installation on Intel Connect Stick

For this series of steps you will need to have the Intel Connect Stick connected to a screen (via HDMI) , keyboard and mouse (via USB junction box). You will need wireless internet and either mains power or power from a rapid-charge powerbank (at least 5volts at 2 amps) with a connecting cable from the powerbank to a USB-C plug (which connects to the Intel Connect Stick)

1 Connect keyboard and mouse to the Intel Connect Stick using a USB junction box
2 Connect Intel Connect Stick to a screen via a HDMI port and the cable supplied with the Connect Stick
3 Insert bootable Ubuntu 18.04 microSD card into dedicated slot on Intel Connect Stick (or the bootable Ubuntu USB disk into the USB junction box)
4 Connect the USB-C power supply to the Intel Connect Stick and watch the screen for when the Connect Stick splash-screen display diskplays a series of options as to how the Connect Stick should proceed.
5 (quickly) press F10 key on keyboard to enter Boot Menu
6 Use the Down Arrow key to select "**UEFI : SDHC: PART 0 : Bootloader**" row entry (and not the "UEFI : Ubuntu : PART 0 : Bootloader" row entry if you are replacing a previous Ubuntu installation).
7 Press Enter key to initiate the booting of a trial version of Ubuntu 18.04
8 Select "Try Ubuntu without Installing" option
9 Once the home page trial version of Ubuntu 18.04 is displaying, double click on the "Install Ubuntu 18.04 LTS" desktop icon to create a permanent version of Ubuntu 18.04 on its main built-in 64 Gb disk

10 Accept appropriate settings for language (eg English US) and keyboard and connect to your wireless internet
11 Updates and other software wizard page – click on Normal installation and both "Other options" (Download updates while installing Ubuntu AND Install third party software for graphics and wifi hardware and additional media formats)
12 If you are doing a fresh installation click on Yes to overwrite previous disk partitions
13 Installation type (if it displays), select "Erase disk and install Ubuntu"
14 Select Time Zone Intel Connect Stick will be used in
15 Complete user login details and Click on "login automatically"
16 Click on **Settings** icon and open the **Power** settings panel. Turn the (Power Saving) **Blank Screen** to **Never** and ensure (Suspend & Power Button) **Automatic suspend** is **OFF**
17 Click on **Settings** icon and open the **Details – Users** panel. **Unlock** and switch the automatic login switch on

15.5.8 Configure Wifi for SOLA Community Server on Intel Connect Stick

With the Intel Connect Stick, the standard wifi functionality that comes with the Ubuntu 18.04 provides **eithe**r a wifi connection **to** a Wifi Access Point (Hotspot) **or** a Wifi Access Point (Hotspot) to **other devices and computers** wanting to connect to the Intel Connect Stick.

While the Hotspot only (with no internet access) option is fine for SOLA Community Server and Open Tenure field recording and the processing of tenure claims, it significantly complicates system administration and means the wifi arrangements need to be reconfigured for this purpose.

The configuration of the "Hotspot only" approach is described in Section 0. However, it is recommended that the following alternative approach is installed **unless** internet or other (not the Connect Stick Hotspot) wireless connectivity is slow and/or unreliable.

<u>Configure the Wifi Connection (possibly with internet)</u>

1. In **Settings – Wi-Fi** confirm the Wifi button (top in panel heading) is **ON**
2. Select your wifi connection from the **Visible Networks** and connect to it
3. Click on the Settings icon to the left of your wifi connection and in the Details tab ensure the **Connect automatically** is ticked. Click on **Apply**

<u>Configure the Intel Connect Stick Wifi Hotspot</u>

1. In **Settings – Wi-Fi** turn the Wifi button (top in panel heading) **OFF**
2. Run **Network Connections** (by a click on its Desktop icon)
3. Click on "**+**" button at bottom of Network Connections panel
4. Select **Wifi** and the **Create** button
5. In the **Connection Name** field **AND** in the **Wifi** tab, enter a name for the hotspot in the **SSID** field (eg CS1 Connect Stick)
6. In the **Wifi** tab and **Device** field, select the only value (the device's MAC address)
7. In the **Wifi Security** tab and **Security** field select **WPA & WPA2 Personal** and a password (eg sola2018)
8. In the **IP4 Settings** tab in the **Methods** field select **Shared to other computers**
9. **Save** and close Network panel
10. Using **Files** locate the new connection file in the /etc/NetworkManager/system-connections folder, select this new connection file, right click and **Edit as Administrator**
11. On line 11 find "mode=infrastructure" and change to "**mode=ap**". **Save** changes to this file
12. In **Settings – Wi-Fi** turn the Wifi button (top in panel heading) **ON**

13. Click on Options icon (to right of Wifi ON-OFF button) and select on **Connect to Hidden Network** and then from the **Connection** field/list select the newly created Hotspot connection – this will disconnect you from any internet connection you might have had.
14. Please note that when you are connected to the Intel Connect Stick using this approach its IP address will be **10.42.0.1**

15.5.9 Install VNC Server (Gnome Vino)

In order to simplify the remaining installations (and eventually user support), you need to install and configure a VNC server. We recommend (and will describe the installation of) the Ubuntu Gnome Vino VNC server package. Please note that this installation description involves minimal system security so as to simplify installation and connection to the Intel Connect Stick

1 open a Terminal window and perform the following commands:
$ sudo apt update
$ sudo apt install net-tools
$ sudo apt install vino dconf-tools

2 Click on Activities (Ubuntu home page) and type in "Sharing" and then click on Settings – Sharing

3 In the Settings – Sharing screen, click **ON** (top toolbar) and click on Screen Sharing panel

4 In the Screen Sharing screen click "ON" (top toolbar) and "Require a password". Enter Password (eg. sola2018). Close this screen and then the Settings – Sharing screen

5 Click on Activities and type in "editor" and then click on the dconf-editor icon

6 Navigate settings tree : org – gnome – desktop – remote-access and make the following changes to the default settings:
alternative-port to **5901** (turn default OFF and increment Custom Value to 5901)
authentication-method to '**none'** (make Custom Value none)
network-interface to '**„**' (turn default OFF and make Custom Value „)
require-encryption to **OFF**
use-alternative-port to **ON**
vnc-password to **sola2018** (enter **sola2018** in Custom Value field)

7 In a Terminal window make the following commands to identify the IP address of the Intel Connect Stick (while connected to the current Wireless Access Point.
$ ifconfig
eg 192.168.1.79

8 Keeping the monitor, keyboard and mouse connections connected, from a VNC Client session on another computer connected to the same Wireless Access Point attempt a VNC connection to the IP address determined in the last step. If your VNC Client requires the port to be specified remember it is 5901 eg 192.168.1.79:5901 or this sometimes abbreviated to 1 eg 192.168.1.79:1

9 Assuming your test VNC connection is successful you can now proceed to create and configure a "Headless" Xorg X11 profile for the Intel Connect Stick so that you can access it without the current peripheral devices (monitor, keyboard and mouse). Without the Headless Xorg X11 profile the Connect Stick screen will not display in your VNC Client session.
Open a Terminal window and make the following commands to create a "Headless" xorg.conf file / profile:
$ sudo apt install xserver-xorg-video-dummy
$ sudo touch /usr/share/doc/xserver-xorg-video-dummy/xorg.conf
$ sudo chmod -R 777 /usr/share/doc/xserver-xorg-video-dummy
$ sudo gedit /usr/share/doc/xserver-xorg-video-dummy/xorg.conf

Add the following to xorg.conf open in the editor gedit

```
Section "Device"
    Identifier  "Dummy"
    Driver      "dummy"
    VideoRam    256000
    Option      "IgnoreEDID"    "true"
    Option      "NoDDC" "true"
EndSection

Section "Monitor"
    Identifier  "Monitor"
    HorizSync   15.0-100.0
    VertRefresh 15.0-200.0
EndSection

Section "Screen"
    Identifier  "Screen"
    Monitor     "Monitor"
    Device      "Dummy"
    DefaultDepth    24
    SubSection  "Display"
        Depth   24
        Modes   "1920x1080" "1280x1024"
    EndSubSection
```

Then make the following commands in a Terminal window:

$ sudo chmod -R 777 /etc/X11

$ sudo cp /usr/share/doc/xserver-xorg-video-dummy/xorg.conf /etc/X11/xorg.conf

Restart the Intel Connect Stick **without any peripheral devices** (monitor, keyboard, mouse) and test the VNC connection from another computer connected to the same Wireless Access Point as the Intel Connect Stick device.

You are now ready to do the Linux Docker Desktop/Engine installation for Community Server and associated software as described in Section *Error! Reference source not found.**.***

16. Troubleshooting

16.1 Community Server

16.1.1 How to describe a software issue

When a problem is encountered it is important to record the following:

1. Any error message details
2. What form, record and field were being accessed and what was the user action immediately before the error message displayed or software problem became evident
3. Was the error able to be repeated ?
4. What is the version of the software (note header on Home page)

The other source of information is the Paraya (raw) log which is viewable from the Paraya Admin console. Under the Common Tasks panel (left side of screen), click on **server** and then within the General Information Screen, click on the **View Raw Log** button. Scroll to the bottom (most recent log entries) and then scroll up until you find reference to the software incident. These log entries will often provide the reason for the software issue. A copy of an extract of the raw log file from before the incident when the system was operating OK until after the incident with all the log entries resulting from the incident should be made – highlight those entries, CTRL-C and paste into a text editor.

If you need technical support from someone else, a copy of all of these items should be sent to that person, together with (if it is practical) a backup of the SOLA Community Server database.

16.1.2 Paraya Deployment Problems

If you experience Paraya deployment problems that are not solved by restarting the domain1 domain (either through the Paraya Admin console or by the command line ./**asadmin stop-domain domain1** and then ./**asadmin start-domain domain1** (run from the C:\payara5\bin or /etc/paraya5/bin folder), then stop the domain1 domain and delete the **generated** sub-folder (C:\ paraya5\glassfish\domains\domain1**generated – for Windows (**or /etc/paraya5/glassfish/domains/domain1/**generated – for Ubuntu).** Then restart the domain1 Paraya domain

16.1.3 Paraya Starts but Community Server does not run

When you type in http://localhost:8080 or http://localhost:8080/admin for Web Admin ((or http://domainname.org:8080 or http://ipaddress:8080 etc) into your web browser and you do not get the Community Server Home Page then you should follow the following steps:

1. Open Paraya Admin console (http://localhost:4848 or http://domainname.org:4848 or http://ipaddress:4848) and click on **Applications** in the left hand panel
2. Open the Applications, and click on the **Re-deploy** link for the Community Server and Web Admin application
3. Under **Server** (near the top of the page) click on the **Restart** button

16.1.4 Changes to Dynamic Tab Definitions or GeoServer Settings are not reflected in Community Server and/or Open Tenure

In SOLA Web Admin select Cache – Cache Reset menu option **AND**
in a Command window (run as Administrator) type the following commands:

Windows	**Ubuntu**
sc stop PayaraSOLA sc start PayaraSOLA	$ sudo service paraya_sola restart

16.2 Open Tenure

As for the Community Server, the same items should be reported if a problem is encountered with Open Tenure:

1. Any error message details

2. What form, record and field were being accessed and what was the user action immediately before the error message displayed or software problem became evident
3. Was the error able to be repeated ?
4. What is the version of the software (check Settings from Main (Home) form)

Useful items to be included with any software incident report are:

- An Export of the Open Tenure log
- A Backup of the local data

Both these items can be generated from the **Settings** menu in the **List of Claims** form. These items can be retrieved from the Open Tenure folder within the device's storage

Another useful item is an export file of the Claim that was being processed when the software incident occurred. **Claim export file** (as used in manual upload process) is generated from selecting the claim in the **List of Claim** form and clicking on the export icon (arrow icon pointing upwards) and leaving the password field blank

Upload issues most commonly occur when there is a weak internet connection. The easiest problem in such situations is to wait until you have a stronger connection. Such problems are aggravated when there are numerous or large attachments and can be overcome by reducing the number of attachments and/or reducing the resolution of the camera of your device to the lowest setting.

Another solution is to do a **manual upload** by using the Claim Export and then connecting your device directly to the Community Server by USB cable or by transferring the Claim export file to a computer which has a good internet connection. Logon to Community Server and go through the Claim upload process.

Annex 1 - SOLA Community Server Database Data Dictionary

Version: Release 1807a

Each of the SOLA applications has a distinct database schema although they have many common elements including the schema structure.

The SOLA databases have been developed using the PostgreSQL open source database and is an implementation of an extended version of the Land Administration Domain Model (LADM). LADM is published as ISO 19152 by the International Organization for Standardization (ISO). In the design of the SOLA databases, it has been necessary to extend ISO 19152 because of the operational needs of land administration agencies to incorporate case management and other features into any system that supports the processing of client service requests (for land information, registration and cadastre change requests and others) and the maintaining and updating of the record of rights and restrictions, ownership and property boundaries.

The SOLA Data Dictionaries are a series of linked html pages. There is an index for each of the SOLA database and then html pages organized according to the the SOLA database schemas. To navigate the data dictionaries, follow the hyperlinks available on each page. Each schema section includes a list of all tables within the schema along with table descriptions, column list, constraints and relationship details. Database views and functions are also described.

Tags are used to summarise common table features. The tags used in this dictionary are

- Business Rules - The table provides configuration or implementation details for business rules.
- Change History - All changes to the table are tracked in a duplicate _historic table for audit and recovery purposes.
- FLOSS SOLA Extension - The table or column is an extension to the LADM standard.
- LADM Reference Object - The table is an implementation of a object from the LADM standard.
- Map Configuration - The table provides configuration details for the Map Viewer.
- Not Used - The table has been included in the Community Server database schema as it represents an object from the LADM, however it is not used by the Community Server application.
- Reference Table - The table contains code values used in other Community Server tables.
- System Configuration - The table provides Community Server system configuration details.
- User Admin - The table is part of the user management system in Community Server.

Using the script found in the Github database repository (../database/scripts/winos/**gen_data_dictionary.bat**) and the Community Server database current as of 17 May 2020, a data dictionary for the Community Server sola database was generated to accompany this Administration Guide (31 May 2020).

Annex 2 - gitslave routine to download all Open Tenure repositories

This routine assumes the setup is on a Windows workstation.

1. Create an Account on GitHub (first time use only)
   Go to www.github.com and click on the **Signup and Pricing** tab and then click on **Create a Free Account** button. Enter user name, email and password (this is NOT your github passphrase) and click on the **Create an Account** button.
2. Setup **git** (first time use only)
   - Download Git for Windows from https://git-scm.com/download/win
   - Install by **Run as Administrator** downloaded installation file changing default editor to Notepad++ and selecting the "Use Git from Git Bash only" option
   - Perform all of these steps to setup up SSH keys and your details
     - from https://help.github.com/articles/generating-ssh-keys
     i. Run Git Bash **cd ~/.ssh** ("No such file or directory" should display)
     ii. Use the email registered with www.github.com
     ssh-keygen -t rsa -C your_email@youremail.com
     iii. Enter file in which to save the key
     (/c/Users/*yourWindowsUserName*.ssh/id_rsa): ***[Press enter]***
     iv. Enter passphrase (empty for no passphrase): ***[Type a passphrase]***
     v. Enter same passphrase again: ***[Type passphrase again]***
     vi. Run the following code to copy the key to your clipboard.
     clip < ~/.ssh/id_rsa.pub
     vii. Log onto www.github.com logon and click on the icon for your Account Settings
     viii. Click **"SSH Keys"** in the left sidebar
     ix. Click "**Add SSH key**" button
     x. Paste your key into the "**Key**" field
     xi. Click "**Add key**"
     xii. Confirm the action by entering your GitHub password
     xiii. Back to GitBash type in **ssh -T git@github.com**
     xiv. Verify that the fingerprint matches the one generated and copied into your githube ssh Keys and type "**yes**"
     xv. If that username is correct, you've successfully set up your SSH key. Do not worry about the shell access comment.
3. Setup **gitslave** (first time use only)
   - Download gitslave from https://sourceforge.net/projects/gitslave/files/
   - UnTar the download file and copy resulting folder to C:\ (eg C:\gitslave-2.0.2)
   - Add C:\gitslave-2.0.2 to the Environment variable **Path**

   Right click **This PC – Properties – Advance System Settings – Environment**
4. Complete the following **git and gitslave** processes:
   - Create the following directory structure on C:\drive
     - C:\SOLAsource\OpenTenure
   - Run Git Bash
     - enter your github passphrase when prompted
     - **cd C:/solaSource/OpenTenure** to navigate to the super repository (code) root folder. If this folder does not exist create this folder.
     - **git clone git://github.com/OpenTenure/code** to create the main repository for Open Tenure
     - **cd code**
     - **./gits –v populate** if you lose your connection during this action repeat the gits populate command until you are successful for all 10 git repositories.

Annex 3 - SQL script to revert sola database to “new installation” state

```
-- Claim & Boundaries (Community Features) forms data
delete from opentenure.claim_comment;
delete from opentenure.claim_comment_historic;
delete from opentenure.claim_uses_attachment;
delete from opentenure.claim_uses_attachment_historic;
delete from opentenure.attachment_chunk;
delete from opentenure.attachment;
delete from opentenure.attachment_historic;
delete from opentenure.claim_location;
delete from opentenure.claim_location_historic;
delete from opentenure.party_for_claim_share;
delete from opentenure.party_for_claim_share_historic;
delete from opentenure.claim_share;
delete from opentenure.claim_share_historic;
delete from opentenure.claim;
delete from opentenure.claim_historic;
delete from opentenure.party;
delete from opentenure.party_historic;
alter sequence opentenure.claim_nr_seq restart with 1;
delete from opentenure.administrative_boundary;
delete from opentenure. administrative_boundary_historic;

-- Dynamic forms data
delete from opentenure.field_payload;
delete from opentenure.field_payload_historic;
delete from opentenure.section_element_payload;
delete from opentenure.section_element_payload_historic;
delete from opentenure.section_payload;
delete from opentenure.section_payload_historic;
delete from opentenure.form_payload;
delete from opentenure.form_payload_historic;

-- Dynamic forms structure
delete from opentenure.field_constraint_option;
delete from opentenure.field_constraint_option_historic;
delete from opentenure.field_constraint;
delete from opentenure.field_constraint_historic;
DROP TRIGGER __track_history ON opentenure.field_template;
delete from opentenure.field_template;
CREATE TRIGGER __track_history
    AFTER DELETE OR UPDATE
    ON opentenure.field_template
    FOR EACH ROW
    EXECUTE    PROCEDURE    public.f_for_trg_track_history();delete    from
opentenure.field_template_historic;
delete from opentenure.section_template;
delete from opentenure.section_template_historic;
delete from opentenure.form_template;
delete from opentenure.form_template_historic;
```