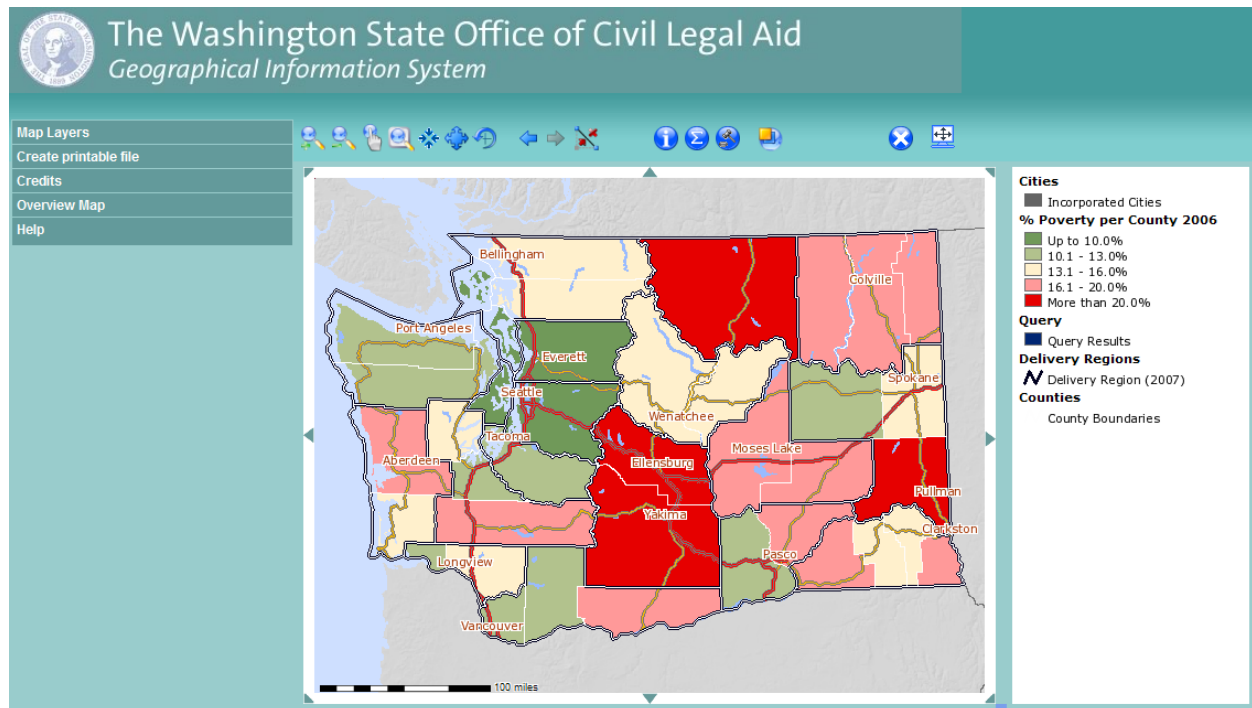


The Washington State Office of Civil Legal Aid *Geographic Information System*

The OCLA GIS Technical Guide



Technical Guide Version June 6, 2011
Prepared by Karsten Vennemann, Terra GIS, Seattle, WA

Contents

1	Introduction to this Guide	6
2	Overview of the OCLA System Architecture	8
2.1	Operating System and Server Software	8
2.2	Programming Languages	8
2.3	The GIS Software Stack	9
2.3.1	Data and Database	9
2.3.2	MapServer - the Map Rendering Engine	11
2.3.3	Mapbender - the Web Mapping Framework	11
3	How to install and configure OCLA Software Components	13
3.1	HostGIS Linux	13
3.2	Installation of GIS, Database and supporting components	15
3.3	Mapbender Installation	18
3.3.1	Prerequisites	18
3.3.2	Main Installation	19
3.3.3	Mapbender Database	23
3.3.4	References	23
4	OCLA GIS System Configuration and Administration	26
4.1	File System Structure	26
4.2	Loading Data into PostGIS	27
4.3	Database backup	30
4.4	Cartography and Map Configuration with MapServer	30
4.5	Configuring WMS Services with MapServer	31
4.6	Loading and Configuring WMS services with Mapbender	31
4.7	Adding Data to the Mapping Application	31
4.8	Enabling Attribute Identification for Map Layers	32
4.9	Query Engine	32
4.10	Managing the Mapbender Installation and OCLA System Users	32
4.11	Work flow for installing OCLA GIS	34
5	Data used in OCLA GIS	36
6	Appendix	37
6.1	Software	37
6.2	Documentation	39
6.3	Support	39
6.4	Resources	39

6.5	Examples	40
6.5.1	MapServer map file layer example	40
6.5.2	Query Template reference in a MapServer map file	41
6.6	Data and GIS Data Layers	42

List of Tables

1	Content of file system folders	26
2	Data sub folders	26
3	MapServer Facts	37
4	Mapbender Facts	38
5	PostGIS Facts	38
6	OS Software Documentation	39
7	OS GIS Reference Books	39
8	GIS Layers - Census Counties Year 2000	42
9	GIS Layers - Census Tracts Year 2000	43
10	GIS Layers - Census Counties Year 2009	44
11	GIS Layers - Census Counties Year 2010	44
12	GIS Layers - Base Layers	45
13	GIS Layers - Boundaries	45
14	GIS Layers - Client Service Resources	45
15	GIS Layers - Background	46
16	GIS Layers - Other Info	46
17	GIS Layers - Comparative Demographics	46
18	Conflicting GIS data layers in the OCLA GIS	47
19	Sources of GIS Data Layers	49

List of Figures

1	The OCLA GIS Architecture	10
2	File system structure	27
3	OCLA and Mapbender Database Tables	29
4	Mapbender Web based Administration GUI	33

1 Introduction to this Guide

This technical guide is part two of the documentation of the Washington State Office of Civil Legal Aid (OCLA) Geographic Information System (GIS). OCLA GIS is an Internet based application that allows the display, query and analysis of demographic data, information regarding Civil Legal Aid providers, and other relevant information presented spatially on maps and in reports. It was built primarily for the use of the Washington's Alliance for Equal Justice and it's partners. The information system requires Internet access and can be accessed using a web browser. User access is restricted and requires a user name and password. The following URLs ¹ are valid Internet sites to log into the system: <http://alliance.terragis.net> and <http://ocla.terragis.net>.

Part one is the *user guide* that describes what OCLA GIS is, what its goals and functionalities are, which data are included in the application and documents how to use them. Part two (this document) focuses on the technical issues of the system, such as installation procedures of the software components for OCLA, the system architecture, data load procedures and software configurations that are needed for the set-up of the OCLA system. One of the primary goals of this technical documentation is to enable a technically knowledgeable person to re-store or re-build the OCLA system on the same or a different server computer in case that will be necessary in the future. Because the OCLA GIS is a complex system this technical documentation can only include a short de-

scription of each of the components in OCLA GIS, and a brief description on how the single components can be configured to work together as far as it directly applies to the OCLA GIS. For additional and more in depth documentation and general information on each of the software components in OCLA links to resources and additional documentation are included throughout this document. Skills needed by anyone who will want to successfully install and configure the OCLA system on a server include the following:

- system administration knowledge (Linux): deal with file permissions and access rights and software installations in a Linux environment
- database administration (PostgreSQL and PostGIS): install and set-up users and access rights, load tables and (spatial) data into a couple of databases
- web server administration: installation and configuration of web server software (Apache2), setting up permissions and access rights on the file system for the web user

Back-ups of the file system and databases are highly recommended to ensure the longevity of the system. The ability to restore corrupted files, lost or damaged files due to malicious or accidental causes is invaluable to maintain a healthy system over time ². In case of the OCLA system only part of these tasks fall into the domain of the GIS consultant *Terra GIS LTD*. Terra GIS is responsible to make sure that the GIS system itself is running smoothly and that the

¹Uniform Resource Locator, in simpler terms a web site address

²The fact that the question is not *if* but *when* any computer hard disk drive eventually will fail illustrates this necessity

system functionalities are working correctly. The server administration tasks such as *data back-up* and *security monitoring* (firewall, automatic security checks etc.) that are good practice on any web server have been provided by a third party hosting service *HostGIS* www.hostgis.com that has done an excellent job over the last years working together with Terra GIS Ltd.

2 Overview of the OCLA System Architecture

2.1 Operating System and Server Software

Operating System

The current OCLA system is running on a shared virtual server with administrative access for the OCLA system administrator (Terra GIS LTD). The server is hosted by HostGIS and running on a 64 bit server with a HostGIS Linux operating system. While any other Linux system can be used for restoring OCLA GIS this document will describe the procedure as it applies to the original system. HostGIS Linux <http://www.hostgis.com/linux> is a *Slackware*¹ based Linux flavor that has one significant advantage over using other Linux distributions: it comes pre-installed with the majority of GIS software component needed for running OCLA GIS and thus is a significant benefit over manually installing many components in addition to the operating system². The components that HostGIS Linux provides and that OCLA Web GIS directly takes advantage of are the following:

- **PostgreSQL**
- **PostGIS**
- **MapServer**
- **GDAL/OGR**
- **Proj4**
- **PHP 5**

¹<http://en.wikipedia.org/wiki/Slackware>

²Another viable alternative to the existing operating system would be to use Ubuntu Linux and the pre-packaged software packages from UbuntuGIS <https://wiki.ubuntu.com/UbuntuGIS>

The components above will be discussed in more detail in the following chapters. In short using the HostGIS Linux distribution to install the server with an operating system and the GIS software suite will save a lot of potential troubleshooting and configuration issues because all of its components have been tested to work with each other. Many other operating system could be used instead of HostGIS Linux (such as Ubuntu for example <http://www.ubuntu.com>), however in the following chapters we will focus on the original set-up and configuration of the OCLA system.

Web Server

The Apache 2 HTTP web server software is used for serving web pages. Apache including the PHP scripting language is belongs to the HostGIS Linux distribution and does not need to be installed separately. In case one is using a different Linux distribution manual installation of PHP is required. For additional documentation on the Apache HTTP server consider the Apache documentation <http://www.apache.org> for the PHP programming language see <http://www.php.net>.

2.2 Programming Languages

PHP

PHP is a programming language that is used for server side tasks in OCLA GIS. More information about PHP can be found here <http://en.wikipedia.org/wiki/PHP>. In other

words components of the OCLA system are written in PHP and that software runs directly on the server computer and not in the users web browser.

JavaScript

Java Script is a scripting language widely used in web based applications <http://en.wikipedia.org/wiki/JavaScript>. One its primary uses is as client side scripts that enable dynamic functionality in web pages. In OCLA GIS multiple Java scripts libraries are used to enable interactive functionality of the web application. In addition to the Java Script Libraries that are included in Mapbender such as JQuery the functionality OCLA GIS query builder is based on the ExtJS library [http://en.wikipedia.org/wiki/Ext_\(JavaScript_library\)](http://en.wikipedia.org/wiki/Ext_(JavaScript_library)).

2.3 The GIS Software Stack

To run OCLA GIS a variety of different software packages are needed to run on the web server. Those software packages are sometimes called the *software stack*. The following will briefly discuss which packages are needed to run OCLA GIS and lists the functionalities which each is adding to the system. The software components that are used in OCLA are part of an inter-operative system that in their entirety make up the framework for the GIS application. Figure 1 on page 10 shows an overview of the system architecture that shows the high level components of the OCLA system. OCLA GIS system encompasses an HTTP server (*Apache 2*), a data repository (*GIS data files* and GIS data in a *PostGIS spatial database*), a map rendering engine (*MapServer*), a web mapping framework that supports client and server side (*Mapbender*), and some additional components such as several supporting software li-

braries³. The following sections will describe the roles that each of the components is fulfilling in the inter-operative system.

2.3.1 Data and Database

The data that are used in OCLA are not all in just one single format. The GIS or spatial data and related attribute data such as the GIS layers (geometries) and information about service providers and demographic data are partially stored in GIS files such as ESRI shape files and partially loaded in the OCLA database as spatial features. Especially the data that can be queried are all stored in the database to provide easy access to the attributes via the query builder and other tools such as the identify tool. The database used in the OCLA GIS houses tables for the administration and configuration of the Web mapping framework. In fact the positions of tool button and general elements on the GUI of the web mapping tool is stored in the *mapbender* database. Attribute and Geographical features of GIS layers are stored in the *OCLA* database⁴. PostgreSQL and the PostGIS extension is used for all database needs in OCLA GIS.

PostgreSQL and PostGIS

The functionality includes data import/export, storage and retrieval of spatial geometries, and spatial indexing (R-tree) of spatial objects in PostgreSQL by adding functions, casts, and storage types. It enables spatial operations and analysis by simple means of running (spatial) SQL queries in the database. In short PostGIS is a GIS of its own

³For example the ExtJS Java script library <http://www.sencha.com/>

⁴Both databases, the one called *mapbender* housing a large part of the system configuration and the one called *ocla* containing GIS data and related data tables are databases within the PostgreSQL/PostGIS system

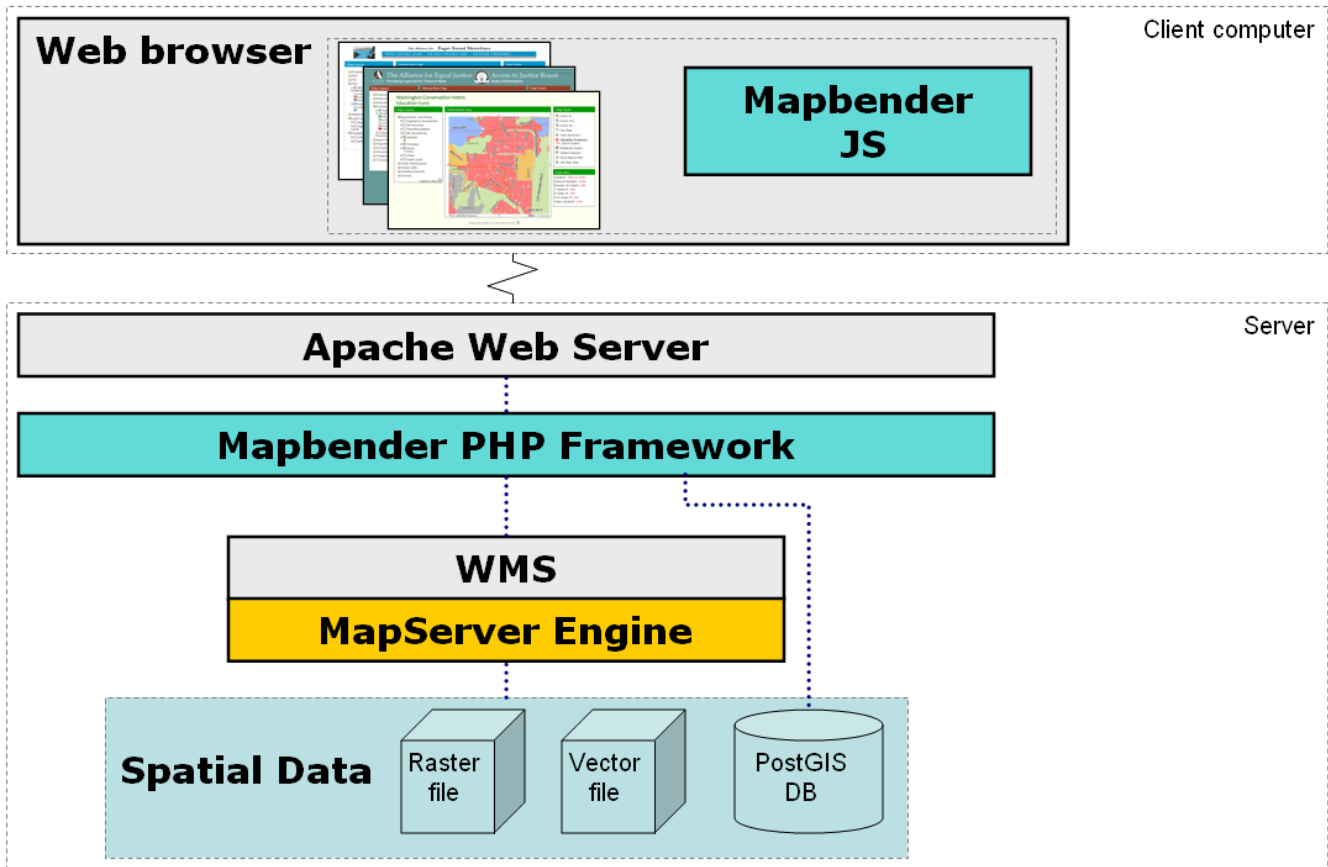


Figure 1: The OCLA GIS Architecture



PostGIS is an extension for PostgreSQL and adds support for geographic objects to PostgreSQL. It enables PostgreSQL server to be used as a back end spatial database for GIS.

right and can greatly enhance web GIS applications with additional functionality. In a sense it provides similar functions as ArcSDE but operates differently. It is not a middle ware sitting on top of a database, rather it is an internal extension of the database written in C. There is no difference in the use of spatial and non-spatial objects in PostGIS and all

objects are accessible via SQL queries to a GIS application.

When the first version of PostGIS was released in 2001 only MapServer accepted PostGIS as an input format, however over time it has become the standard spatial database back end for all the other open source GIS tools (compare Table 5). Starting with version 9.3 ArcGIS has read and write support for PostGIS but requires an ArcSDE license to connect. Prior to version 9.3 ArcGIS was lacking native support for PostGIS, however a third party ArcGIS extension "**zigGIS**" (available from <http://pub.obtusesoft.com>) and the Data Interoperability extension can enable earlier versions of ArcGIS to use PostGIS layers.

2.3.2 MapServer - the Map Rendering Engine

A map rendering engine is a software component on the Server that can create map images (e.g. in such formats such as jpg, png or gif) from raw geographic datasets (GIS layers) as input. The images that are generated on request e.g. then a user zoom into the map for switches on a GIS layer are the delivered via HTTP protocol to the web browser on the client computer for viewing. All the logistics of these requests is handled by another component the web mapping framework ⁵. Input data for the MapServer engine include raster and vector data and come in a variety of formats such shape files, data in spatial databases. MapServer as the map rendering engine needs some kind of configuration that defines how the real world features should appear on the map, a simple text file called the *map file*. The configuration in the MapServer map file defines if rivers are blue, have labels and which colors all the other features on a map should be shown in. For more information about map file and the map file syntax consult the comprehensive MapServer documentation available at <http://mapserver.org/documentation.html> and the PDF-version of the documentation <http://mapserver.org/MapServer.pdf>.



MapServer is a map rendering engine that is used in OCLA GIS and was developed at the University of Minnesota (UMN). It is one of the most mature and most successful open source GIS projects and is implemented in the C programming language. The main focus is on rendering spatial data and in providing a de-

⁵in the case of OCLA the Mapbender web mapping framework

velopment environment for spatially-enabled Internet applications. Some of the highlights about MapServer are that it supports more input data sources than most of the proprietary products, has higher performance, and the pre-compiled versions are simpler to install and to set up (for details see table 3). There are two main ways that one can interact with MapServer:

- using it as a CGI⁶ application
- using one of several MapScript APIs to write your custom application

The MapScript API⁷ and is available for the following programming languages: PHP, Python, Perl, Ruby, Java, and C#. MapServer uses a configuration text file that commonly has the .map extension. This file configures how MapServer will output (render) a map. The configuration includes general information about the maps such as *map projection*, *size of the map* in pixels, *map image file* output format etc. Individual layer tags are used to determined the classification of a layer. Examples for input formats are *Map-Info* files, *ESRI shape* files, *PostGIS*, *Oracle Spatial*, *ArcSDE*, *WMS*, and all other *GDAL* and *OGR* formats ⁸. Output formats include GIF, JPG, PNG, all GDAL formats, WFS and WMS (compare table 3).

2.3.3 Mapbender - the Web Mapping Framework

The web mapping framework is the software component that is used to run and manage the overall OCLA application. The web map-

⁶Common Gateway Interface is a standard protocol for interfacing external application software with web servers.

⁷An application programming interface (API) is a set of functions, procedures, methods, classes or protocols.

⁸For OGR input formats see http://www.gdal.org/ogr/ogr_formats.html and for GDAL input formats http://www.gdal.org/formats_list.html

ping framework includes components that are stored and run on the server (server side) as well as components that are downloaded from the server to the client's web browser (client side) and are executed in the users computer (e.g. HTML pages and Java scripts code embedded into this). The combination of server and client side code enables the OCLA application to dynamically serve maps, enable secure access to map functionality and enables advanced options such as printing maps, run database queries and to generate reports.

Mapbender

Mapbender is a comprehensive client and server side mapping framework on which OCLA GIS is based. It is implemented in PHP, Java Script and XML and based on an administration database that can either be housed in either MySQL or PostgreSQL. Some of its functionality such as measuring distances on a map is only available if PostgreSQL/PostGIS is used as an administration database. It features a web based administration interface to manage applications, application layout, user and groups access rights, mapping services, available tool functionality and security for WMS and WFS via OWS proxy functionality (compare Table 4). Functionality for each of the housed web mapping applications include displaying, navigating, editing and querying spatial data and maps. Input sources are solely OGC WMS compliant map services and WFS. In the near future it is envisioned to embed OpenLayers as a map viewing option in addition to the intrinsic Mapbender viewer. For more information about *Mapbender* see <http://www.mapbender.org/>.

3 How to install and configure OCLA Software Components

The OCLA web GIS system is an interoperable solution. What this means is that it is not a single monolithic software application but instead there are multiple building blocs as laid out in section 2.3 starting on page 9. Note that most components for the OCLA system are pre-installed in case of using HostGIS Linux, and that if OCLA will continue to use the services of HostGIS as a hosting provider HostGIS Linux itself is pre-installed on the server so that no installation for the components is required. The main exception from this is that in order to run OCLA the Mapbender framework needs to be installed and configured to work with the existing MapServer, PostGIS and PHP installations. The sections provided in the following pages are intended as a reference in case in the future another service provider is chosen in order to make sure that the needed software building blocks can be installed and configured from scratch if ever needed.

Another viable alternative to the original HostGIS Linux installation is to choose the newest stable Ubuntu distribution as the server operating system. The GIS components can be easily added using the *apt-get* command line utility on the server and the *Ubuntu GIS* repositories that provide pre-compiled packages for all of the required components for OCLA except for **Mapbender**. For further information review the information on the Ubuntu web pages <http://www.ubuntu.com/> and the Ubuntu GIS web site <https://wiki.ubuntu.com/UbuntuGIS>.

3.1 HostGIS Linux

The installation of HostGIS Linux is described in detail on the project web site http://www.hostgis.com/linux/wiki/index.php/Install_HostGIS. You can always find the most up to date information on that website. To install HostGIS Linux you first will need to obtain the software package and burn it to onto CDs. Using a CD or DVD drive in your server you can then perform the installation process. The following sections are taken from the HostGIS Linux web site:

Obtain the HostGIS Distribution

The current HostGIS distribution can be downloaded from this website www.hostgis.com - see the Wiki page - download. Download the "Basic Server CD" and burn the downloaded ISO file to a CD. To start the installation, place the HostGIS CD in the computer's hard disk drive and start the computer. If the CD (or DVD) drive you use is recognized as a boot device, and is set in priority ahead of other devices like the hard disk drive, the computer will boot using the HostGIS CD.

If your computer does not boot from the CD, turn it off and restart it. As the computer powers up, request access to its basic setup system (e.g., by pressing the F2 key, which works on many Intel motherboards). This little system will generally give you the opportunity to select which device is checked first for the computer's boot image. Make sure your CD drive is the first device checked.

HostGIS Linux installation

The installation process appears as a series of pages or screens containing instructions and questions. In this section the pages are presented just as they appear in the example installation. Each page has been numbered in sequence, with as much of the text and menu choices documented as feasible. Note: The test computer for this installation is a custom-built server. The motherboard is Intel, as is the dual-core Pentium CPU. The board has 4 gigabytes of RAM. There is a single 80-gigabyte IDE hard drive and a single IDE DVD/CD drive. Place the HostGIS installation CD-ROM in the DVD/CD drive and power up the server.

From here follow the instructions on the installation screens. The process should be relatively self-explanatory. In case you need more details during the installation consult the *step by step installation guide* on the project web site http://www.hostgis.com/linux/wiki/index.php/Install_HostGIS.

Post Installation Configuration

<http://www.hostgis.com/linux/wiki/index.php/Post-Installation>

Apache HTTP web server is one of the components natively included in HostGIS Linux. A complete list of included software can be found here <http://www.hostgis.com/linux>.

3.2 Installation of GIS, Database and supporting components

All components listed in this section except for *ExtJS* are pre-installed if using HostGIS Linux (the original OCLA GIS set-up). The following installation instructions only apply in case Ubuntu Linux is chosen as the operating system for the server. There are multiple options to install the GIS components such as MapServer, PostgreSQL and PostGIS, GDAL/OGR, Proj4 and the scripting language PHP. The following paragraphs will describe one efficient installation option, using the Ubuntu GIS repositories <https://wiki.ubuntu.com/UbuntuGIS> and the pre-compiled software packages via the *apt* package manager.

To use the repositories (and to be able to install the GIS components via the *apt* package manager) first we need to add the Ubuntu GIS repositories to the list of available packages to make them known to the system. We also will need to determine which version of Ubuntu we are running. On the command shell this can be determined via entering *lsb_release -c*. The output will list out operating system version name. Assuming that the output response was *hardy*, which means that we are running a copy of the *hardy heron* Ubuntu operating system. The URL locations for the Ubuntu repositories will need to be added to the system repositories list, which is most likely located at */etc/apt/sources.list*. To edit you can use any available text editor e.g. *nano* will do:

```
nano /etc/apt/sources.list
```

Add the following entries (for the stable Ubuntu GIS packages) to the end of the source.list file and save the file.

```
deb http://ppa.launchpad.net/ubuntugis/ppa/ubuntu hardy main
deb-src http://ppa.launchpad.net/ubuntugis/ppa/ubuntu hardy main
```

Now we need to make the new entries known to the operating system:

```
sudo apt-get update
```

Most likely we will now get an error message output - probably similar to the one below:

```
GPG error: http://extras.ubuntu.com maverick Release: The following
signatures could not be verified because the public key is not available
: NO_PUBKEY 16126D3A3E5C1192\\
```

This happens because the two new repositories have not been verified by the system. The last step to make them available or use is to generate a public key e.g. similar to:

```
sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 089
EBE08314DF160;
gpg --export --armor 089EBE08314DF160 | sudo apt-key add -
```

To make this work we need to replace the arbitrary value *089EBE08314DF160* above with your specific error message output. For example if the error message was

```
... public key is not available: NO_PUBKEY 16126D3A72D340A3 ...
```

Then the command we will need to run would be

```
sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 16126
D3A72D340A3;
gpg --export --armor 16126D3A72D340A3 | sudo apt-key add -
```

Now we need to run the update command to make the new repositories known to the system:

```
sudo apt-get update
```

If this does not generate an error message but instead lists all available repositories then we have successfully completed the configuration for the additional repositories. We are now ready to install the GIS components on our Ubuntu system.

MapServer

To install MapServer:

```
sudo apt-get install cgi-mapserver
sudo apt-get install mapserver-bin
```

PostgreSQL and PostGIS

To install PostgreSQL and PostGIS you might need to add another repository to your sources.list file in order to be able to get the pre-compiled 8.4.4 version of PostgreSQL and the version 1.5 of PostGIS:

```
\url{deb http://de.archive.ubuntu.com/ubuntu hardy-backports}
```

The commands to install are as follows

```
sudo apt-get install postgresql-8.4
sudo apt-get install postgresql-server-dev-8.4 libpq-dev
sudo apt-get install libgeos-dev
sudo apt-get install postgis-1.5
```

To verify that the installation was successful for PostGIS we can query of the installed version in an SQL terminal (either via pgsq or using PgAdmin). The SQL command

```
select version();
```

will output the installed PostgreSQL version:

```
"PostgreSQL 8.4.1 on x86_64-unknown-linux-gnu, compiled by GCC gcc (GCC)
4.2.4, 64-bit"
```

while using the query:

```
select postgis_full_version();
```

will output a response similar to the one below if the PostGIS installation is complete

```
"POSTGIS="1.5.1" GEOS="3.1.1-CAPI-1.6.0" PROJ="Rel. 4.6.1, 21 August 2008"
LIBXML="2.6.32" USE_STATS"
```


In case PostgreSQL is installed but the PostGIS install is not complete consult the PostGIS manual at <http://postgis.refractory.net/docs/>.

PHP Scripting Language

To install **PHP 5** and the required PHP modules (more information at <https://help.ubuntu.com/8.04/serverguide/C/php5.html>)

```
sudo apt-get install php5 libapache2-mod-php5
sudo apt-get install php5-cli
sudo apt-get install php5-cgi
sudo apt-get install php5-mysql
sudo apt-get install php5-pgsql
```

Supporting Software Libraries: GDAL/OGR and Proj 4

To install **gdal**

```
sudo apt-get install gdal-bin
```

To install **proj**

```
sudo apt-get install proj
```

ExtJS - Java Script Library

The OCLA query builder is based on an additional JavaScript library called *ExtJS*. In order to install the JavaScript framework it only needs to be extracted (copied) to the correct location in the file system. The location in the original set-up is */var/www/sites/mapbender/http/ext/*. Thus, to install simply extract the zipped ExtJS library *ocla_ExtJS.zip* from the OCLA DVD to create the directory above. More information about ExtJS is available at www.sencha.com.

3.3 Mapbender Installation

3.3.1 Prerequisites

The following sections closely follow the instructions of the Mapbender installation documentation available at http://www.mapbender.org/Manual_Installation. Adjustments and additions have been made to adapt the online documentation for this technical guide and to extract the parts that are relevant for the **OCLA GIS** system. To be able to install and operate Mapbender on the server a variety of supporting libraries for the Apache HTTP server and the PHP scripting language are required. The list of packages below should be installed before continuing with the installation of the main Mapbender software:

- Webserver (Apache, MS IIS)
- an installation of PHP (scripting language, version > 5.1.x)
 - php-mbstring
 - php-gettext
 - php-gd2
 - php5-imagick
 - php5-pgsql
- Database (PostgreSQL >= 8.x, PostGIS has to be enabled)
- Installation of gettext see http://www.mapbender.org/Gettext#Utility_programms

The following instructions describe the overall configuration process for these packages, please follow the installation instructions for your operating system individually as the paths to files might vary on different systems. On HostGIS Linux almost all PHP extensions are pre-installed and some of them are actually running as an apache module. For the **Ubuntu** operating system one can use pre-compiled packages that can be installed via the apt-get command. For example the following commands will install the PHP extensions listed above:

```
sudo apt-get install php-mbstring
sudo apt-get install php-gettext
sudo apt-get install php-gd2
sudo apt-get install php5-imagick
sudo apt-get install php5-pgsql
```

After installation of these packages restart Apache e.g.

```
/etc/rc.d/rc.httpd stop
/etc/rc.d/rc.httpd start
```

Once all are installed the presence of the PHP modules can also be verified by a short PHP script (see below). To do this create a file with the content below and give it an arbitrary name e.g. *phpinfo.php* and copy it into the servers web directory. Visiting the web URL e.g. www.ocla.net/phpinfo.php with your web browser will list all installed PHP extensions as well as general information about the PHP installation on the server. The script should later be removed from the web accessible directory as it might give out some information about the system for potential hackers that potentially might facilitate hacking attacks on the web server.

```
<?php
    phpinfo() ;
?>
```

3.3.2 Main Installation

You can obtain a copy of the newest stable release Mapbender from http://www.mapbender.org/Download_Mapbender. If you want to install the bleeding edge version of the software you may use an SVN client to check out the code directly from the source code repository. During installation Mapbender will create the directory structure shown below on your file system:

- **conf** - directory for configuration files mapbender.conf digitize_default.conf gazetteer.conf
- **http** - directory for the application. Within this directory the modules are sorted by type in subdirectories. Some modules are pure JavaScript applications, even though they have .php as suffix.
- **license** - license information files
- **log** - logfiles will be saved here
- **owsproxy**
- **resource/db** - directory for the SQL-Dump to build up the Mapbender database or update the database
- **resources/locales** - directory contains translation files for different languages (Mapbender.po-files)

run the install-Script

Mapbender operates by using an administrative database. There is an install-script that creates the database and also creates the tables and content that are needed to run the Mapbender software. Please observe that for proper security you should not use the default PostgreSQL user *postgres* to access the Mapbender database. For example you can create a user called **mapbender** to access the database. Below are SQL statements you can use to create the user in the database. Note that those need to be run when connected to the database in a SQL terminal. For more information about *psql* see <http://www.postgresql.org/docs/8.4/static/app-psql.html> or alternatively one can use the **pgAdmin** program (see <http://www.pgadmin.org/> and its graphical user interface to connect to the database and to run the commands below in a SQL query terminal.

```
CREATE USER mapbender;
ALTER ROLE mapbender PASSWORD 'somepassword';
ALTER ROLE mapbender SUPERUSER;
```

To be able to create tables and constraints the Mapbender database user needs superuser privileges. After running the above SQL statements the super user privileges of the **mapbender** user should be revoked again because they are not required any more and might pose an unnecessary security risk.

```
ALTER ROLE mapbender NOSUPERUSER;
```

The next step is to run the install script *install_2.6.sh* from the directory *./mapbender/resources/db/* in your command shell.

```
./mapbender/resources/db/
```

Notes

These notes are from a standard Debian or Ubuntu Linux installation. If you are interested to see what is happening in the database you can check the log, accessible through:

```
tail -f /var/log/postgresql/postgresql-8.3-main.log
```

You will see several warnings and notices, these are normal, just watch out for **error** messages. The default user for Apache is *www-data*. This following section describes the details required for running a Mapbender installation.

Apache Installation and Configuration

In HostGIS Linux the Apache HTTP web server is already installed on the system. For installing Apache on Ubuntu please refer to the Apache HTTP Server documentation for the Ubuntu operating system. To configure Mapbender to work with Apache the next step is to create a *virtual* directory. It will allow you to access the Mapbender scripts from a browser. Follow the schema below and exchange *<path-to-mapbender-http>* with the path to the web directory of the Mapbender directory *./mapbender/http/*

```
Alias /mapbender /<path-to-mapbender-http>
<Directory /<path-to-mapbender-http>
    Options MultiViews
    DirectoryIndex index.php
    Order allow,deny
    Allow from all
</Directory>
```

The location of the Apache configuration files on the file system might vary depending on the version you are using, for example:

```
/etc/apache/httpd.conf
/etc/apache2/sites-available/default
```

Each time you make changes to the Apache configuration you will need to remember to reload the configuration file so that Apache knows about the changes (i.e. to stop and start the Apache web server). To optimize the performance of Mapbender served via Apache it is advantageous to enable output compression of your web server. Here's an instruction for Apache (see http://httpd.apache.org/docs/2.0/mod/mod_deflate.html for more details). Just append the content below to your *Directory* settings in your Apache configuration (e.g in */etc/apache2/httpd.conf* or */etc/apache2/sites-available/000-default* depending on your installation)

```
<Directory /var/www/apache2-default/mapbender/http> # (exchange with
    your path)
```

```
Options MultiViews
AllowOverride None
Order deny,allow
Deny from all
Allow from 127.0.0.0/255.0.0.0 ::1/128

# Insert filter
SetOutputFilter DEFLATE

# Netscape 4.x has some problems...
BrowserMatch ^Mozilla/4 gzip-only-text/html

# Netscape 4.06–4.08 have some more problems
BrowserMatch ^Mozilla/4\.0[678] no-gzip

# MSIE masquerades as Netscape, but it is fine
# BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

# NOTE: Due to a bug in mod_setenvif up to Apache 2.0.48
# the above regex won't work. You can use the following
# workaround to get the desired effect:
BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

# Don't compress images
SetEnvIfNoCase Request_URI \
\.(\?: gif|jpe?g|png)$ no-gzip dont-vary

# Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary
</Directory>
```

Make sure you have loaded the required modules (for example, Ubuntu and Apache2)

- a2enmod deflate
- a2enmod headers
- a2enmod setenvif (for BrowserMatch)

With these configuration settings in Apache Mapbender will now most likely load almost twice as fast on your web site.

PHP Configuration

You might have to make some adjustments to the file `php.ini`. Changes in the `php.ini` become operative only after restarting the Apache web server.

1. During the test and installation phase of Mapbender it can be helpful to log php-errors. Configure the file `php.ini` as follows:

```
error_reporting = E_ALL & ~E_NOTICE
log_errors = On
errorlog = "/var/log/php.log"
```

2. Check the following details in php.ini allow_url_fopen parameter:

```
extension_dir = (path to extensions—directory)
allow_url_fopen = On
```

3. Mapbender manages access authorization in SESSIONS. Check the following details:

```
session.save_handler = files
session.save_path = (Path to SESSIONS—Directory). (Check authorization!)
session.use_cookies = 1
```

4. Further details of session management should be customized according to server load (see 'garbage collection' in php.ini)

5. With regard to other modules the extension **gd2** should be included in the installation and configuration process.

```
extension=php_gd2.dll      (Windows)
extension=gd.so           (Linux)
```

On a Debian-System you find this lib in **/usr/lib/libgd.so**. If it is not installed you can do that using the command (or use Synaptic Packet Manager or similar. This information might be out of date for **Ubuntu 9.04**):

```
apt-get install libgd2-dev
```

6. To run Mapbender a PostgreSQL database is required. You have to check the availability of the required extension in the php.ini file:

```
extension=php_pgsql.dll    (Windows)
extension=pgsql.so         (Linux)
```

7. Mapbender needs php_gettext and php_mbstring

```
extension=php_gettext.dll  (Windows)
extension=php_mbstring.dll (Windows)
```

8. short_open_tag

```
short_open_tag = OFF
```

9. Mapbender requires simplexml and dom for XML parsing. If not installed, this will lead to some hard-to-find bugs, because PHP fails to throw exceptions. These modules should be already installed if you are using a Linux distribution. Furthermore check, whether the database information in **./conf/mapbender.conf** is correct.

3.3.3 Mapbender Database

Please use the install-script *install.sh* as described in the paragraphs above in order to create the PostgreSQL database unless you know exactly what you are doing. Use the update script (*update.sh*) to keep your database up-to-date. During installation and updates you need PostgreSQL SUPERUSER privileges to create and initialize the Mapbender database to temporarily disable constraint triggers. After installation a user with a regular role with INSERT, UPDATE and DELETE permissions on all tables, PostGIS functions and sequences should be used. For security reasons you should not use a PostgreSQL SUPERUSER role for regular operation of a public site.

3.3.4 References

For instructions on how to do a manual installation of the Mapbender database see http://www.mapbender.org/Manual_Installation. Additional information can also be found at the web sites listed below.

- PostgreSQL documentation <http://www.postgresql.org/docs/current/interactive/app-createuser.html>
- Notices Ubuntu notes for PostgreSQL users <https://help.ubuntu.com/community/PostgreSQL>

Mapbender Configuration File

The Mapbender core system configuration **mapbender.conf** file is located in the directory:

```
./mapbender/conf/
```

Please observe that the download package contains a default file named `mapbender.conf-dist`. This is to ensure that during a Mapbender update your configuration is not overwritten. Copy `mapbender.conf-dist` to a new file named `mapbender.conf` to take effect. Find a detailed description of every settings in the Mapbender.conf page. Check your Webserver configuration. The following example shows how it is configured for Apache. Define a directory for temporary files, this directory must have write access

```
define("TMPDIR", " ../tmp");
```

URL for login

```
define("LOGIN", "http://localhost/mapbender/frames/login.php");
```

```
define("MAXLOGIN", 3);
```

```
$login = LOGIN;
```

```
# URL to owsproxy (no terminating slash)
```

```
define("OWSPROXY", "http://localhost/owsproxy");
```

```
# get more information on how to configure owsproxy.
```

```
# type of server-connection curl, socket, http

#define("CONNECTION", "curl");
define("CONNECTION", "http");
#define("CONNECTION", "socket");

define("CONNECTION_PROXY", "<ip>");
define("CONNECTION_PORT", "<port>");
define("CONNECTION_USER", "<user>");
define("CONNECTION_PASSWORD", "<password>");

# security: path to modules

define("CHECK", true);
```

Note that setting "CHECK", true ensures that users cannot add modules to their applications which they do not already own. Use especially when running Mapbender in Portal mode and allowing anonymous users to create Mapbender accounts.

Mapbender can communicate with its users by email. For example, when a user lost his/her password, Mapbender can send an email to this person with a new password. Another example: Whenever a WMS is updated, the users using this WMS are notified by Mapbender via email. To enable Mapbender to send emails, the administrator must specify a mail server (and set up a mail account on this server). Note that Mapbender only features SMTP mailing. Note that this functionality is currently not set-up for OCLA.

For additional settings check 'class_administration.php', or please check phpmailer (<http://sourceforge.net/projects/phpmailer>) for more information. Currently, phpmailer 1.72 is in use.

```
define("USE_PHP_MAILING", "false");
define("MAILHOST", "<host>");
define("MAILUSERNAME", "<user>");
define("MAILPASSWORD", "<password>");
define("MAILADMIN", "<mailaddress of the mailadmin>");
define("MAILADMINNAME", "<mailadminname>");
$mailHost = MAILHOST;
$mailUsername = MAILUSERNAME;
$mailPassword = MAILPASSWORD;
```

Mapbender error logging

To set Mapbender error logging see the possible options below:

```
define("LOG\_LEVEL", "error"); // "off", "error", "warning", "notice" or "all"
define("LOG\_JS", "on"); // "on", "alert", "console" or "off"
```


Internationalisation

```
define("USE_I18N", true);
```

To define a default language see the example below

```
define("LANGUAGE", "de"); // "en", "de", "bg", "gr", "nl", "it", "es"
```

Additional configuration options:

- you can change the translations in the Mapbender.po (mapbender/resources/locale/de_DE/Mapbender.po files if you prefer a different translation.
- You can edit the Mapbender.po files with an editor
- or with poedit
- after changing Mapbender.po you have to compile the Mapbender.mo files again (make sure that gettext is installed).

Change the Permission of log Folder in Linux

Change the permission of the directory ./log to a value that allows the PHP script owner to write logs. On Linux systems give ownership of the directory to the Web server user (on Linux for example *www-data*) and allow write permissions:

```
chown -R www-data ./log
chmod -R 640 ./log
```

Checking the Mapbender Setup

You can check your setup with the script mapbender_setup.php. The script mapbender_setup.php moved to the directory:

```
/mapbender/tools/mapbender_setup.php
```

To access the script you have to move it temporary to /mapbender/http/tools/mapbender_setup.php. Note that for **security reasons** the directory tools should not be available for external users, as e.g. the mapbender_setup.php displays internal information about your system (PHP version, database name, database user...). You can protect the directory by htaccess settings in Apache or just remove the tools-directory after testing. Next run **mapbender_setup.php**: http://localhost/mapbender/tools/mapbender_setup.php. The mapbender_setup.php script checks whether all configurations of the mapbender.conf are correct and does some system settings checks. After the installation you can login with the user account *root* and the initial password *root*. Make sure to change this password asap in order to secure your Mapbender installation. Go to admin2_en and edit the **root** user to change the password. You should not delete the user root. To administer your applications you should create a new user e.g. *admin* which you can use instead of user root.

4 OCLA GIS System Configuration and Administration

Configurations and and post installation procedures specific for the OCLA GIS system and technical OCLA administrative procedures will be described in this chapter. In order to successfully proceed with the configuration and set-up of the OCLA system described in this chapter we assume that the installation of the entire software stack of the OCLA GIS system was successfully completed and that the components are running on a fully configured and Internet accessible HTTP web server.

4.1 File System Structure

The file system structure that is specific to OCLA is shown on page 26. Figure 1 and Table 2 list the general content of each of the folders of the two main directories. The file structure for the Mapbender will already be nearly complete for OCLA after the installation. There are a few files that will need to be copied or updated into some of the folder because they have been adapted for OCLA specific needs and differ from the regular Mapbender installation (see instructions below). Also note that using a Mapbender version other than the one provided on the original **OCLA GIS DVD** that was delivered to the *Office of Civil Legal Aid* together with this guide might need some changes in the PHP or Java Script files or even the database tables and structure. Thus, this needs to be done with caution and should only be done by an expert familiar with deeper knowledge of PHP, PostgreSQL, PostGIS and preferably familiar with Mapbender and if urgently needed. In

all other cases it is recommended to use the Mapbender version ¹ provided on the DVD and only to copy and update the files as noted below.

Table 1: Content of file system folders

conf	Mapbender configuration files
http	web accessible folder
license	software license notes
log	Mapbender error logs
owsproxy	OWS proxy server functionality
resources	set-up and maintenance scripts

Note that the OWS Proxy Server functionality of Mapbender (for use with secured web services) was not needed for OCLA GIS and has not been configured. For more information about OWS proxy functionality of MapBender see http://www.mapbender.org/OWS_Proxy

Table 2: Data sub folders

css	css style sheets
help	PDF help documents
img	web site graphics files
layers	GIS data, shape files
templates	MapServer query templates
alliance	MapServer configuration
main	*.map files

¹currently as of this writing this is Mapbender version 2.7.1 http://www.mapbender.org/download/mapbender_2.7.1.zip

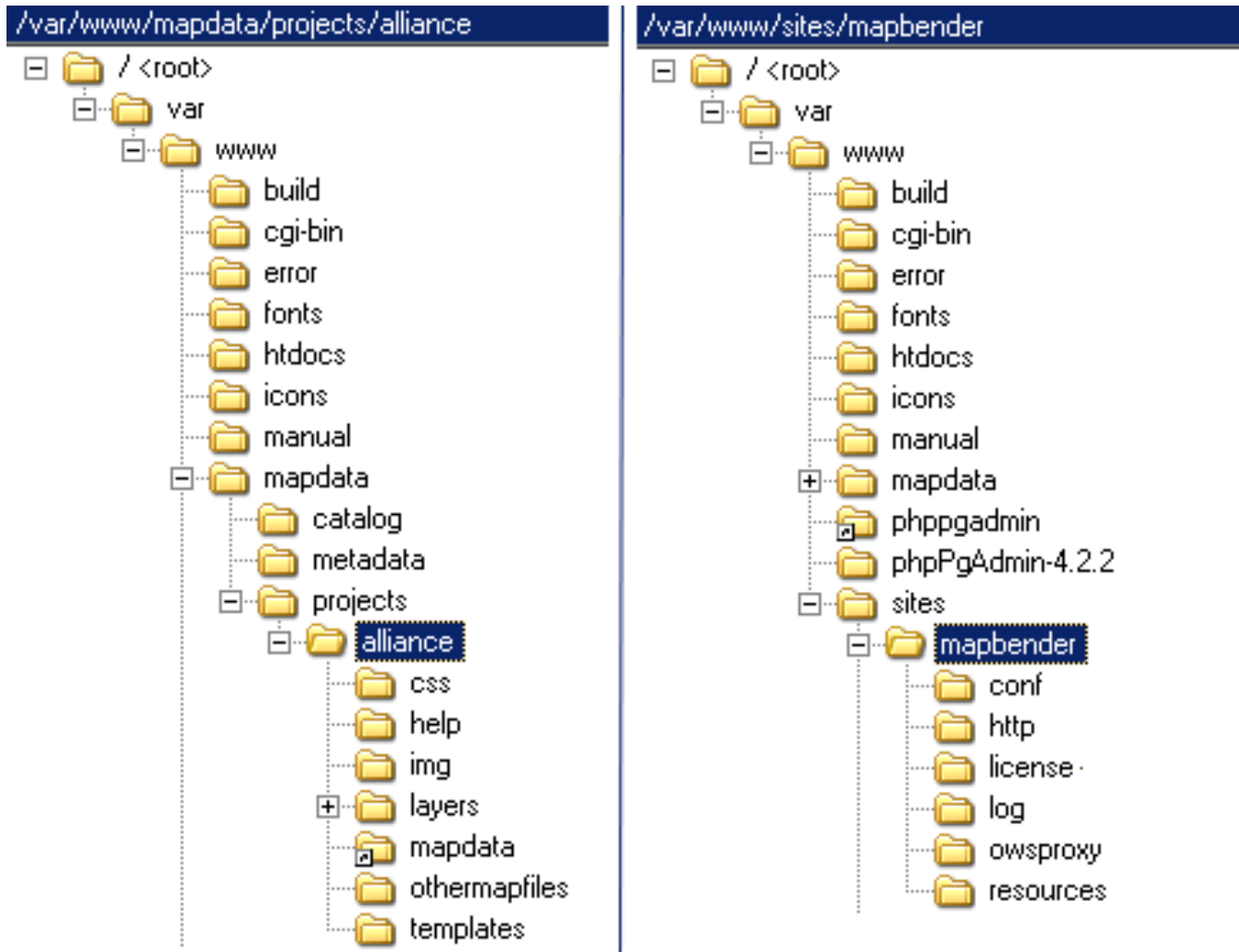


Figure 2: File system structure

4.2 Loading Data into PostGIS

To load data into PostGIS (and PostgreSQL) for our purposes and to restore or install the OCLA system GIS data on a server we can use `pg_restore` and import the database dump provided on the OCLA DVD. The command to restore the entire database including the data data is as follows:

```
pg_restore -d ocla --host localhost
--port 5432 --username postgres --
format custom --file /var/www/
mapdata/projects/alliance/db/
ocla_all_june3.backup ocla
```

Note that there might be error messages regarding already existing PostGIS functions in the database. If the database software version is not different in the first digit of the software package version the import potentially can output a lot of error messages but still should run through fine. You should check in the database if the GID layers have all been imported and the tables have been filled with the respective records. If the first digit of the PostgreSQL version is higher, e.g. when using version 9.1 instead of 8.4, then this process might become cumbersome and it is recommended to import all the single ta-

ble backup files one at a time (provided on the OCLA DVD). The command to import any individual table will look exactly the same as the one for the entire database above except that you will specify the backup file that contains only the one table. Another option is to run the plain SQL backup file (provided on the OCLA DVD) if the above process makes any trouble. This can be done using *psql* on the command line:

```
psql -d ocla -h localhost -U pgsql
-f counties.sql
```

Another method to load the GIS data into the database is loading the original source GIS data directly from the ESRI shape files into the data base (included in OCLA DVD). This process will also be needed in case new shape files have to be loaded into the database, e.g. if the query builder functionality should make use of them. Overall there are two utilities that are installed with PostGIS: the first *shp2psql* that can import shape files into PostGIS, and the second one for the reverse process to export PostGIS features out of the database as ESRI shape files. Note that there is another utility that is distributed with the ogr/gdal package (see 17) called *ogr2ogr* that can import and convert between other GIS formats and the PostGIS data base format. Further information about ogr2ogr can be accessed at <http://www.gdal.org/ogr2ogr.html>. To import a shape file into PostGIS you can use a command similar to the one below.

```
shp2psql -s 2285 -l wa_places.shp
wa_places| psql -d ocla -U
pgsql -h localhost
```

The command above will write the output of the shape file **wa_places.shp** conversion directly into the *ocla* database into the *wa_places* database table including all attribute data and spatial geometries, create an spatial index on the feature geometry

(to ensure adequate performance) using the *Washington State Plane North* map projection (EPSG code 2285 ²). Further information about the *shp2psql* and *pgsql2shp* utilities can be found at chapter 4.4.2. *Using the Loader* at <http://postgis.refractory.net/docs/ch04.html>

²More information about EPSG codes can be found at http://en.wikipedia.org/wiki/International_Association_of_Oil_and_Gas_Producers

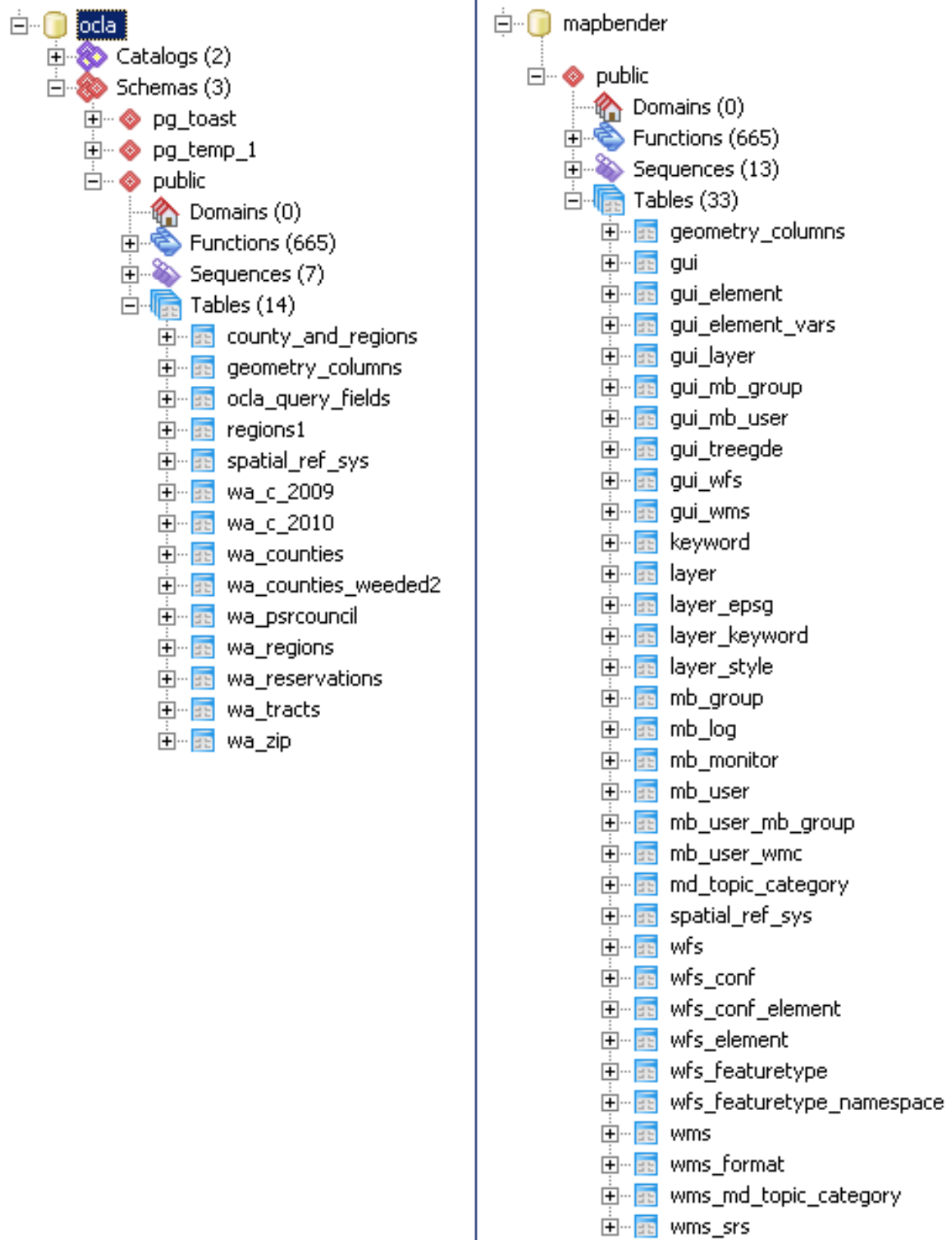


Figure 3: OCLA and Mapbender Database Tables

4.3 Database backup

Apart from regular backups of the entire file system of the server (including the files that represent the database content) it is advised to make regular database backups. Because the data in the OCLA system currently does not change frequently this can be done after changes to the system have been made e.g. when new data layers have been added to the database. To make database backups one can use either the graphical user interface provided by **PgAdmin** or the command line utility **pg_dump**. An example command to back up just one table called *wa_regions* is as follows:

```
pg_dump -t wa_regions --host localhost --port 5432 --username pgsq --format custom --file /var/www/mapdata/projects/alliance/db/ocla_wa_regions_june3.backup ocla
```

In the example above **pg_dump** (a utility installed along with PostgreSQL) will connect to the data base *ocla* using the specified parameters and create a *custom* (compressed format) file backup the table *wa_regions* and write it to the location specified into a file named *ocla_wa_regions_june3.backup*. The resulting backup file can be used to restore the table into the database at later point in time via the **pg_restore** command utility.

```
pg_dump --host localhost --port 5432 --username pgsq --format custom --file /var/www/mapdata/projects/alliance/db/ocla_all_june3.backup ocla
```

In the second example we are creating a backup of the entire content of the OCLA database and write it to a file named *ocla_all_june3*. For more specific information regarding **pg_dump** and **pg_restore** see

the reference web pages of the PostgreSQL database at <http://www.postgresql.org/docs/8.4/static/app-pgdump.html> and <http://www.postgresql.org/docs/8.4/static/app-pgrestore.html>.

To create a plain SQL backup of the OCLA database the command looks as follows:

```
pg_dump --host localhost --port 5432 --username pgsq --format plain --file /var/www/mapdata/projects/alliance/db/ocla_all_plain_june3.sql ocla
```

The above command created a plain SQL backup file (plain text) *ocla_all_plain_june3.sql* in the */var/www/mapdata/projects/alliance/db/* directory. In this directory you can run the SQL to create all tables and import data via copy SQL commands into the newly generated tables. Note that in order to run this successfully the database and the required database users have to already exist.

```
psql -d ocla -h localhost -U pgsq -f counties.sql
```

4.4 Cartography and Map Configuration with MapServer

The map images used in the Mapbender framework are rendered by MapServer on the fly every time when a user interacts with the map, performing actions such as zooming, panning or switching on and off GS layers. The configuration of the how the layers look, whether they are visible, whether they are scale dependent is configured via the MapServer *.map file. This configuration file determines the cartography and much of how the layers look in the OCLA viewer. For the purposes of this guide we will need to refer to the MapServer documentation

for in depth information if changes to the look and feel of the maps will be needed. For further information please refer to the MapServer map file documentation at <http://mapserver.org/mapfile/>

4.5 Configuring WMS Services with MapServer

The last section indicated that GIS data and their appearance in Mapbender rely on the MapServer configuration **.map* files. In order that a layer can be accessed as a web map service (WMS) specific parameters will need to be set in the map file. One WMS can have multiple layers that can be either requested individually or together as a group (potentially making up an entire map). To install the OCLA MapServer configurations we will only need to copy the map files to the file system folders as described in the section *Work flow for installing OCLA GIS* starting on page 34. Map configuration, cartography and setting up WMS services is a wide topic of it's own that goes beyond the scope of this technical installation guide. For further information about MapServer as a WMS server see http://mapserver.org/ogc/wms_server.html.

4.6 Loading and Configuring WMS services with Mapbender

The Mapbender framework itself does not render any map images fro display in the map viewer. Instead that functionality is provided by the map rendering engine - in our case MapServer (as described in the previous section). Mapbender manages data sources and provides a framework of tools, modules, user and administration services to configure and administer web mapping application and spatial data infrastructures. Map-



bender accepts only two main types of data input sources that can be incorporated into the system: OGC Web Map Services (WMS) and OGC Web Feature Services (WFS). Currently OCLA GIS only uses web map services (WMS).

4.7 Adding Data to the Mapping Application

To add new GIS data to OCLA GIS a two step process is needed: first we will need to create a WMS service, and secondly we will need to configure the WMS for use within the Mapbender framework. The first step of setting up a WMS is described in the section *Cartography and Map Configuration with MapServer* on page 30, the configuration to make the WMS available within the Mapbender framework is described in the section *Configuring WMS services with MapServer* on page 31. To add other data such as attribute data of census updates or other information that can be used in queries or the query builder we would need to load the data into the PostgreSQL database. This can be accomplished via SQL scripts and in a second step we can either join the data in the respective queries or create a new table that would encompass the spatial features alongside all attributes that need to be queried. For simplicity the latter option might be the best one. To add new data columns to the existing counties layer the following example illustrates how to add columns to the *counties* table in PostGIS, then how to load data from a text file (*.csv file) into a new table, and finally how to update the newly added columns with the new *census* data. In the second step we will need to load the WMS into the Mapbender database and enable the display in the GUI using the web based Mapbender administration tools. For specific instructions on managing WMS resources in Mapbender see

<http://www.mapbender.org/AddWMS>.

4.8 Enabling Attribute Identification for Map Layers

In OCLA GIS many attributes of the GIS layers can be queried via the  query builder and report tool and the identify tools . The following describes how a layer needs to be configured in order to return attributes values in a query when identifying a feature on the map³. To enable retrieval of attribute information we will need to follow a two step process. The first requirement is that the WMS service is configured to deliver feature attribute information. This is done via OGC standard **WMS GetFeatureInfo** requests⁴. Thus, we will need to configure this type of request in our WMS configuration for the layer in question. In the MapServer map file a MapServer HTML query template will need to be specified. In the template the attributes that will be retrieved by the query are specified. For example in the map file this can look like this (inside a layer declaration):


```
TEMPLATE "templates/wa_counties.  
html"
```

A sample map file GIS layer configuration and a MapServer query template can be found starting on page 41 in the appendix of this guide.

³For information about the query builder see the sections following this chapter

⁴For MapServer see http://mapserver.org/ogc/wms_server.html

4.9 Query Engine

The query engine for the  query builder and report tool is built using Java Script based on the ExtJS library. The interactive functionality of the query builder uses Ajax calls from Java Script to call PHP scripts on the server side that retrieve the requested information from the database. For more information about the ExtJS framework see www.sencha.com. For the purpose of this guide the steps required to enable the query builder are as follows⁵:

- unzip the ExtJS zip archive provided on the OCLA DVD to create the directory `/var/www/sites/mapbender/http/ext/`
- import the complete GIS data and supporting database data into the *mapbender* and *ocla* data bases into PostGIS as described in section *Work flow for installing OCLA GIS* starting on page 34.

The table that is used for the configuration of the query data layers, their name aliases in the query builder display is called `ocla_query_fields` see figure 3 on page 29.

4.10 Managing the Mapbender Installation and OCLA System Users

Many of the settings in the OCLA system can be configured via the web based administration interface of Mapbender. In case of the original set-up of the OCLA Mapbender installation the URL is <http://alliance.terragis.net/> or as an alias for the same site <http://ocla.terragis.net/>. When logging in as a regular user we will be forwarded to the OCLA GIS mapping application. When we are logging in as the site

⁵provided that the remainder of the OCLA system software is already configured as described in Chap-

Mapbender GUI list

Change personal settings



GUI name	GUI description
admin1	admin GUI containing all administrative modules (use only as a template)
admin2_de	admin GUI in deutscher Sprache
admin2_en	admin GUI in english
admin_de_services	erweiterte admin GUI - WMS, WFS, Metadaten (deutsch)
admin_en_services	extended admin GUI - WMS, WFS, metadata-handling (english)
gui	GUI with tab, search modules
gui1	GUI combining most of the Mapbender functionality
gui2	GUI with focus on layout
gui_digitize	GUI with WFS search and digitizing using WFS-T
ocla4	GUI with focus on layout

Figure 4: Mapbender Web based Administration GUI

administrator we will see the administration interface. Figure 4 on page 33 shows multiple GUI that can be accessed from here.

In general the interface can be used to administer many web mapping sites, however in our case of OCLA we are dealing only with one site. Some of the GUIs listed are examples that are included in the default Mapbender installation and are irrelevant for our purposes. The only ones that are relevant for OCLA are the administration GUI which are **admin1** and **admin2_en** for administration and **ocla4** which will get us to the OCLA GIS application. The interface allows to add data such as WMS and WFS services, add and change tool functionality that comes with Mapbender, arrange the look and feel and even determine change and store the positioning of the tool buttons in the OCLA web map. For more detailed information consult the Mapbender documentation and wiki pages at <http://www.mapbender.org/>. The steps that are needed to install and

configure the OCLA Mapbender application are as follows:

The **admin2_en** GUI can be used to manage users and user groups of the OCLA system. Under the *User Management* section of **admin2_en** GUI one can add, delete users, assign them to groups, and can change passwords. To create a new user and allow the user to access the OCLA system click on *Create and edit user* and fill in the details such as user name and password on the following screen page. After the user has been created we can use the tools in the *Authorization* section to allow access to the OCLA site. Click on *Allow user access to one GUI* select the GUI first, **ocla4** in our case (blue when selected), next select the user from the list on the lower left and then click on the arrow pointing right to add the user the OCLA GUI. Once the new user name is listed under *SELECTED USER* we have been successful. Another option is to use *user groups* which can assigned permission to access a GUI. For more information on user man-

agement see <http://www.mapbender.org/Benutzerverwaltung>. The users for OCLA GIS are all pre-configured once the **mapbender** data base from the DVD has been imported into PostGIS. For further details on Mapbender administration refer to the Mapbender Wiki at <http://www.mapbender.org/Tutorials#Documentation>.

4.11 Work flow for installing OCLA GIS

The notes below assume that the operating system and the GIS stack for OCLA GIS has been successfully installed (as described in chapter 3). The work flow sequence to configure and to complete the installation of the remainder of the OCLA system is as follows:

- copy data (GIS layers, map files, templates etc) into the *ocla* project directory called **alliance** by unzipping archive file **alliance.zip** from the OCLA DVD to create folder `/var/www/mapdata/project-s/alliance/`

This will create the directory structure shown in Figure 2 *File System Structure* on page 27.

- Verify that Mapbender is installed into the directory `/var/www/sites/mapbender/` shown in Figure 2 *File System Structure* on page 27.
- Replace all files in the Mapbender installation directory by unzipping the Mapbender archive **mapbender.zip** into the directory listed above and by replacing the original installation files.

This procedure will only work if the Mapbender version you are using is identical with the one contained in the zip archive on the OCLA DVD (currently version 2.7.1). If you would like to upgraded the Mapbender version to a newer distribution it is recommended to first follow the Mapbender installation procedure described in this guide and then in

a second step to follow instructions from the Mapbender wiki to upgrade to a newer Mapbender version. However, note that last procedure very likely will require that someone with PHP and JavaScript knowledge will need to alter the scripts of the new Mapbender installation in order to port the functionality added specifically for the OCLA system (custom id tools, print set-up and configuration, and layer conflict resolution) to the new installation to enable the full OCLA functionality.

The in the following steps you will import two database dump files provided on the DVD into the PostGIS installation.

- Import the database dump `ocla_all.backup` provided on the OCLA DVD as follows:

```
pg_restore -d ocla --host localhost
--port 5432 --username pgsql
--format custom ocla_all.
backup
```

- Import the database dump `mapbender_all.backup` provided on the OCLA DVD as follows:

```
pg_restore -d ocla --host localhost
--port 5432 --username pgsql
--format custom mapbender_all.
backup
```

Note that for this procedure we will need to use an empty **mapbender** database that will be populated with the database tables that have been pre-configured for the OCLA system. Thus, we will either first need to delete all database tables in the existing **mapbender** database before starting the database dump import, or we will need to delete the **mapbender** database and create a new one with the same name and the same configuration options (encoding etc) that will then be populated using the database restore procedure.

Alternatively to the use of the compressed (custom) database dump to populate the ocla database the plain SQL database dump *ocla_all.sql* can be used. Note that the ocla database has to exist prior to this procedure. To import individual tables the SQL scripts on the OCLAS DVD in the folder */database/sql* can be used. The command for the SQL import will look similar to the one below:

```
psql -d ocla -h localhost -U pgsql  
-f ocla_all.sql
```

If all steps worked correctly you will now be able to use the newly installed OCLA GIS.

5 Data used in OCLA GIS

The data used in OCLA GIS are derived from a variety of sources and have been updated over time. There are two main types of data in the system: **Demographic Data** (mostly derived from the US Census) and **Service Provider Information** (mostly coming from the *Alliance of Equal Justice* partners themselves. Initially, the main data was based on the census 2000 data, the ACS (Census data on County level) update for 2005, and Client Service Provider data (provided by Alliance partners) and WSBA ¹ data from 2007. Over time the data have been replaced with newer updates of the census for 2006, 2009 and recently with the new census 2010 data (for County level). The data that came in non-spatial form have been joined to their geographic location features and boundaries, such as *Census Tracts*, *Counties* to form a data source that supports spatial display (maps) as well as non spatial (tabular) display of information. The data will appear on the maps and also in tabular format in the results supplied by the identify and query tools. Table 19 on page 49 shows a summary of the data used in OCLA GIS.

¹Washington State Bar Association

6 Appendix

6.1 Software

HostGIS Linux download: <http://www.hostgis.com/linux/wiki/index.php/Download>

Table 3: MapServer Facts

Main supporter of	MapServer Core Team, Steven Lime
MapServer	
Functionality	Map rendering engine, web mapping development environment
Operating systems	Multi platform
Project started	1996
Implementation	C
OS libraries	OGR/GDAL
PostGIS support	Yes
License	BSD
Website	http://mapserver.org

Table 4: Mapbender Facts

Main supporter of Mapbender	WhereGroup, Bonn, Germany
Functionality	Client side and server side mapping framework, based on a database (MySQL and PostgreSQL)
Operating systems	Windows and Linux
Project started	2001
Implementation based on	JavaScript/Ajax, DHTML, PHP MySQL or PostgreSQL
PostGIS support	Via rendering engine
License	GPL
Website	http://www.Mapbender.org

Table 5: PostGIS Facts

Main supporter of PostGIS	Refractions Research, Victoria, Canada
Type	Spatial database. PostGIS is an extension for PostgreSQL
Functionality	Storage and retrieval of spatial data (geometries such as point, line, polygon, multipoint, multiline, multipolygon, geometrycollection). Spatial indexing. GIS functions via spatially enabled SQL. E.g. intersections, distance calculations, reprojection
Operating systems	Windows and Linux
Project started	2001
Implementation	C
OS libraries	GEOS, Proj4, OGR, and FDO
License	GPL
Website	http://postgis.refractions.net

6.2 Documentation

Table 6: OS Software Documentation

Software	Documentation - web site
PostgreSQL	http://www.postgresql.org/docs/
PostGIS	http://postgis.refrations.net/documentation/
MapServer	http://mapserver.org/documentation.html
MapBender	http://www.mapbender.org/Tutorials
PHP	http://php.net/
Apache HTTP Server	http://httpd.apache.org/

6.3 Support

This document was created by Terra GIS Ltd. <http://www.terrakis.net/> in an effort to ensure longtime viability of the OCLA web mapping system. While Terra GIS Ltd. is happy to provide long term support for the OCLA system other providers that support Open source based web mapping solutions can be found at the website of OSGeo, The Open Source Geospatial Foundation at <http://www.osgeo.org/>. The provider search at http://www.osgeo.org/search_profile allows to filter the service provider according to languages and software packages supported by them.

6.4 Resources

Table 7: OS GIS Reference Books

Reference Books	Open Source GIS and web mapping
PostGIS	PostGIS in Action by Regina Obe, Leo Hsu
MapServer	Beginning MapServer: Open Source GIS development by Bill Kropla
OS Web mapping	Web Mapping Illustrated: Using Open Source GIS Toolkits by Tyler Mitchell

6.5 Examples

6.5.1 MapServer map file layer example

```

LAYER                                     # start layer tag
    NAME "Counties"                      # layer name
    TYPE POLYGON                         # layer type (POINT, LINE, POLYGON)
    STATUS ON                            # status (ON, OFF, DEFAULT)
    DATA "../wa_states"                 #
    TEMPLATE "templates/counties_template.html" # location of query
                                          # template in relative relation to map file location
    CLASS                                # start class
        definition
            COLOR 240 240 230
                                          # Color (in this case polygon
                                          # fill)
        END                               # end class definitions
    DUMP true                             # parameter that is required
                                          # to enable WMS output
    PROJECTION                            # start projection of the layer
        "init=epsg:2285"                  # epsg:2285 from epsg coming
                                          # with Proj4 is Washington Sate Plane North
    END                                   # layer projection end tag
    METADATA                              # Metadata tag start
        "wms_title" "States"              # name of this individual WMS layer
        "wms_srs" "epsg:2285 epsg:4326" # list of spatial reference systems this
                                          # layer is available in
        "wms_feature_info_mime_type" "text/html" # format of the
                                          # GetFeatureInfo request
    END                                   # Metadata tag start
END                                     # end layer tag

#####
END                                     # End of map file

```


6.5.2 Query Template reference in a MapServer map file

```
<!-- MapServer Template -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD_HTML_4.01_Strict//EN">
<html>
  <head>
    <title>Identify - Results</title>
    <link href="../../css/template.css" rel="stylesheet" type="text/
      css">
    </head>
    <body>
      <table class="summaryStats">
        <tr><th>Attributes</th><th>[name] [type] Data (CENSUS 2010)</th></tr>
        <tr><td>Total Population 2010</td><td align="center">[total_n_10] </td></tr>
        <tr><td>White</td><td align="center">[white_n_10] </td></tr>
        <tr><td>Black</td><td align="center">[black_n_10] </td></tr>
        <tr><td>American Indian</td><td align="center">[amind_n_10] </td></tr>
        <tr><td>Asian</td><td align="center">[asian_n_10] </td></tr>
        <tr><td>Pacific Islander</td><td align="center">[paisl_n_10] </td></tr>
        <tr><td>Other Race</td><td align="center">[other_n_10] </td></tr>
        <tr><td>Mixed Race</td><td align="center">[twomo_n_10] </td></tr>
        <tr><td>White %</td><td align="center">[white_p_10] %</td></tr>
        <tr><td>Black %</td><td align="center">[black_p_10] %</td></tr>
        <tr><td>American Indian %</td><td align="center">[amind_p_10] %</td></tr>
        <tr><td>Asian %</td><td align="center">[asian_p_10] %</td></tr>
        <tr><td>Pacific Islander %</td><td align="center">[paisl_p_10] %</td></tr>
        <tr><td>Other Race %</td><td align="center">[other_p_10] %</td></tr>
        <tr><td>Mixed Race %</td><td align="center">[twomo_p_10] %</td></tr>
        <tr><td>Hispanic</td><td align="center">[hispa_n_10]</td></tr>
        <tr><td>Hispanic %</td><td align="center">[hispa_p_10] %</td></tr>
      </body>
    </html>
```

6.6 Data and GIS Data Layers

The tables on the following pages list the GIS data layers that are used in the OCLA GIS. For information about data sources of the data and GIS data layers used in OCLA GIS see table 19 on page 49.

Table 8: GIS Layers - Census Counties Year 2000

Layer	Type	Visible	switch on/off
People in Institutions - number	polygon	yes	yes
Percent Non English	polygon	yes	yes
Non English - number	polygon	yes	yes
Percent No Phone Service	polygon	yes	yes
No Phone Service - number	polygon	yes	yes
American Indian percent	polygon	yes	yes
American Indian - number	polygon	yes	yes
Asian percent	polygon	yes	yes
Asian - number	polygon	yes	yes
Black percent	polygon	yes	yes
Black - number	polygon	yes	yes
Mixed Race percent	polygon	yes	yes
Mixed Race - number	polygon	yes	yes
Pacific Islander percent	polygon	yes	yes
Pacific Islander number	polygon	yes	yes
White percent	polygon	yes	yes
White - number	polygon	yes	yes
Hispanic percent	polygon	yes	yes
Hispanic - number	polygon	yes	yes
Other Race percent	polygon	yes	yes
Other Race - number	polygon	yes	yes
Percent 17 and Younger - number	polygon	yes	yes
Seventeen and Younger	polygon	yes	yes
Seventeen and Younger in Poverty - number	polygon	yes	yes
Percent Sixty-five and Older	polygon	yes	yes
Sixty-five and Older - number	polygon	yes	yes
Percent Sixty-five and Older in Poverty	polygon	yes	yes
Percentage of People in Poverty per County 2000	polygon	yes	yes
Total Number of People in Poverty / County 2000	polygon	yes	yes

Table 9: GIS Layers - Census Tracts Year 2000

Layer	Type	Visible	switch on/off
People in Institutions - number	polygon	yes	yes
Percent Non English	polygon	yes	yes
Non English - number	polygon	yes	yes
Percent No Phone Service	polygon	yes	yes
No Phone Service - number	polygon	yes	yes
American Indian percent	polygon	yes	yes
American Indian - number	polygon	yes	yes
Asian percent	polygon	yes	yes
Asian - number	polygon	yes	yes
Black percent	polygon	yes	yes
Black - number	polygon	yes	yes
Mixed Race percent	polygon	yes	yes
Mixed Race - number	polygon	yes	yes
Pacific Islander percent	polygon	yes	yes
Pacific Islander - number	polygon	yes	yes
White percent	polygon	yes	yes
White - number	polygon	yes	yes
Hispanic percent	polygon	yes	yes
Hispanic - number polygon	yes	yes	
Other Race percent	polygon	yes	yes
Other Race - number	polygon	yes	yes
Seventeen and Younger in Poverty - number	polygon	yes	yes
Percent 17 and Younger	polygon	yes	yes
Seventeen and Younger - number	polygon	yes	yes
Sixty-five and Older in Poverty	polygon	yes	yes
Percent Sixty-five and Older	polygon	yes	yes
Sixtyfive and Older Number - number	polygon	yes	yes
Percentage of People in Poverty per Tract	polygon	yes	yes
Total Number of People in Poverty per Tract	polygon	yes	yes

Table 10: GIS Layers - Census Counties Year 2009

Layer	Type	Visible	switch on/off
Percent of People in Poverty per County 2009	polygon	yes	yes
Total Number of People in Poverty per County 2009	polygon	yes	yes
Percent Seventeen and Younger in Poverty per County 2009	polygon	yes	yes
Total Number Age Seventeen and Younger in Poverty 2009	polygon	yes	yes
Percent Age Five to Seventeen in Poverty per County 2009	polygon	yes	yes
Total Number Age Five to Seventeen in Poverty per County 2009	polygon	yes	yes

Table 11: GIS Layers - Census Counties Year 2010

Layer	Type	Visible	switch on/off
American Indian percent 2010	polygon	yes	yes
American Indian - number 2010	polygon	yes	yes
Asian percent 2010	polygon	yes	yes
Asian - number 2010	polygon	yes	yes
Black percent 2010	polygon	yes	yes
Black - number 2010	polygon	yes	yes
Mixed Race percent 2010	polygon	yes	yes
Mixed Race - number 2010	polygon	yes	yes
Pacific Islander percent 2010	polygon	yes	yes
Pacific Islander number 2010	polygon	yes	yes
White percent 2010	polygon	yes	yes
White - number 2010	polygon	yes	yes
Hispanic percent 2010	polygon	yes	yes
Hispanic - number 2010	polygon	yes	yes
Other Race percent 2010	polygon	yes	yes
Other Race - number 2010	polygon	yes	yes
Change Number of White People per County 2000-2010	polygon	yes	yes
Change Percentage of White People per County 2000-2010	polygon	yes	yes

Table 12: GIS Layers - Base Layers

Layer	Type	Visible	switch on/off
Cities	polygon	yes	yes
Place Features	Text /Label	yes	no
Populated Places	Text /Label	yes	no
Parks	polygon	yes	yes
Uninhabited Areas	polygon	yes	yes
Indian Reservations	polygon	yes	yes
Rivers and Streams	lines	yes	yes
Puget Sound	polygon	yes	no
Ocean	polygon	yes	no
Lakes	polygon	yes	yes
Roads	lines	yes	yes
Railway	lines	yes	yes
Public Facilities	point	yes	yes
City Names	Text /Label	yes	no

Table 13: GIS Layers - Boundaries

Layer	Type	Visible	switch on/off
Congressional Districts	polygon outline	yes	yes
Legislative Districts	polygon outline	yes	yes
Counties	polygon outline	yes	yes
County Labels	Text /Label	yes	yes
Tract Boundaries	polygon outline	yes	yes

Table 14: GIS Layers - Client Service Resources

Layer	Type	Visible	switch on/off
WSBA Members	point	yes	yes
Northwest Justice Project	point	yes	yes
Northwest Justice Project FTE Label	Text /Label	yes	yes
Columbia Legal Services	point	yes	yes
Columbia Legal Services FTE Label	Text /Label	yes	yes
Pro Bono Programs	point	yes	yes
Specialty Legal Aid Providers	point	yes	yes
Mediation Services	point	yes	yes
Law School	point	yes	yes

Table 15: GIS Layers - Background

Layer	Type	Visible	switch on/off
States	polygon	yes	no
Topography	raster	yes	yes
Counties	polygon	yes	no
Providers by Region (2007)	polygon	no	no
Providers Counties	polygon	no	no
Puget Sound	polygon	yes	no
Ocean	polygon	yes	no
Cities	polygon	yes	no

Table 16: GIS Layers - Other Info

Layer	Type	Visible	switch on/off
Farm Workers (2000 Larson Study)	polygon	yes	yes
WSBA Members per ZIP code area (2008)	polygon	yes	yes
Percentage of People in Poverty per Tract 1980	polygon	yes	yes
Percentage of People in Poverty per Tract 1990	polygon	yes	yes
Percentage of People in Poverty per Tract 2000	polygon	yes	yes

Table 17: GIS Layers - Comparative Demographics

Layer	Type	Visible	switch on/off
Change People in Poverty (County) 2000-2009	polygon	yes	yes

Table 18: Conflicting GIS data layers in the OCLA GIS

Layer	Area	Layer	Area
People in Institutions	Tract	Other Race number	County
Percent Non English	Tract	Pacific Islander number	County
Non English	Tract	Asian number	County
Percent No Phone Service	Tract	American Indian number	County
No Phone Service	Tract	Black number	County
Mixed Race percent	Tract	White number	County
Other Race percent	Tract	Seventeen and Younger in Poverty	County
Pacific Islander percent	Tract	Percent 17 and Younger	County
Asian percent	Tract	Seventeen and Younger	County
American Indian percent	Tract	Sixtyfive plus in Poverty	County
Black percent	Tract	Percent 65plus	County
White percent	Tract	FTEs per Region	County
Mixed Race number	Tract	Percentage of People in Poverty	County
Other Race number	Tract	Total Number of People in Poverty	County
Pacific Islander number	Tract	Percentage of People in Poverty 2000	County
Asian number	Tract	Percentage of People in Poverty 2006	County
American Indian number	Tract	Total Number of People in Poverty 2000	County
Black number	Tract	Total Number of People in Poverty 2006	County
White number	Tract	Increase Number of People in Poverty 2000-2006	County
Seventeen and Younger in Poverty	Tract	Increase Percent of People in Poverty 2000-2009	County
Percent 17 and Younger	Tract	125 Total Number of People in Poverty 2006	County
Seventeen and Younger	Tract	Active Attorneys	County
Sixtyfive plus in Poverty	Tract	WSBA Members	ZIP code
Percent 65plus	Tract	Hispanic percent	Tract
People in Institutions	County	Hispanic number	Tract
Farm Workers	County	Sixtyfive and Older Number	Tract
Percent Non English	County	Hispanic percent	Tract
Non English	County	Hispanic number	Tract
Percent No Phone Service	County	Sixtyfive and Older	County

Continued on next page

Conflicting GIS data layers continued from previous page

Layer	Area	Layer	Area
No Phone Service	County	125 Percentage of People in Poverty 2000	County
Mixed Race percent	County	125 Total Number of People in Poverty 2000	County
Other Race percent	County	125 Percentage of People in Poverty 2006	County
Pacific Islander percent	County	Sixtyfive and Older Number	Tract
Asian percent	County	Percentage of People in Poverty	Tract
American Indian percent	County	Total Number of People in Poverty	Tract
Black percent	County	Percentage of People in Poverty 1980	Tract
White percent	County	Percentage of People in Poverty 1990	Tract
Mixed Race number	County	Percentage of People in Poverty 2000	Tract

Table 19: Sources of GIS Data Layers

Name	Data Source	Link
2000 Stats county and census tract	US Census Bureau, 2000 Decennial Census (100% + 125% of FPL)	http://factfinder.census.gov
2010 Stats county	US Census Bureau, 2010 Decennial Census (100% + 125% of FPL)	http://factfinder.census.gov
Larson	2000 Larson Study	http://www.ncfh.org/enumeration/PDF11%20Washington.pdf
ACS	US Census Bureau, American Community Survey 2009. Poverty Status imputed at county level (125% of FPL)	http://factfinder.census.gov
PSRC	Puget Sound Regional Council Statistical Information (7/2008). Forecast Analysis Zones (FAZ) 2000-2010 for the Puget Sound area.	http://www.psrc.org/data/gis/shapefiles
GIS Base Layers	Various public sources and State agencies: WSDOT, Washington State Geospatial Clearinghouse, and other sources	http://www.wsdot.wa.gov/mapsdata/geodatacatalog/default.htm , http://wa-node.gis.washington.edu/
Client Service Resources	Alliance of Equal Justice Partners, OCLA, WSBA	N/A

Index

- Administration
 - OCLA users, 32
- Apache, 20
 - compression, 21
 - installation, 20
 - Output Compression, 20
- Apache HTTP Server, 8, 13
- Appendix, 37
- architecture, 10
- background layers, 46
- backup, 6
 - database, 30
- Base Layers, 45
- Books
 - MapServer, 39
 - PostGIS, 39
 - Web mapping, 39
- Boundaries, 45
- Cartography, 30
- Census Data
 - Counties Year 2000, 42
 - Counties Year 2009, 44
 - Counties Year 2010, 44
 - Tracts Year 2000, 43
- census data, 36
- Client Service Resources, 42
- Comparative Demographics, 46
- data, 42
 - adding, 31
 - census, 36
 - client service provider, 36
 - convert, 28, 30, 34
 - demographic, 36
 - file system structure, 27
 - files, 27
 - GIS, 31
 - GIS layer, 30
 - identify, 32
 - import, 28, 30, 34
 - layers, 31
 - loading, 27
 - provider, 36
 - shape file, 9, 11, 28
 - sources, 36, 49
 - spatial, 9, 27, 30, 36
- Data folders, 26
- Database
 - PostGIS, 9
 - PostgreSQL, 9
 - Spatial data, 9
- database, 9
 - backup, 30
 - import, 28, 30, 34
 - PostGIS, 9
 - PostgreSQL, 9
 - schema, 29
 - spatial, 9
- documentation, 39
- examples, 40, 41
- ExtJS, 17, 32
 - installation, 17
- firewall, 6
- framework, 11
- GDAL/OGR, 17
 - installation, 17
- Geographic Information System, 6
- GIS, 6
 - architecture, 10
 - attributes, 32
 - cartography, 30
 - data, 31, 36, 42
 - data sources, 49

- layers, 31, 36, 42–46
 - software, 9
 - support, 39
- HostGIS Linux, 14
 - installation, 14
- hosting service, 6
- installation
 - Apache, 20
 - ExtJS, 17
 - GDAL/OGR, 17
 - HostGIS Linux , 14
 - libraries, 17
 - Mapbender, 18
 - mapping framework, 18
 - MapServer, 16
 - overview, 13, 18
 - PHP, 17
 - PostGIS, 16
 - PostgreSQL, 16
 - Proj4, 17
 - server, 13
- JavaScript, 9, 11
 - ExtJS, 32
- layer
 - conflicts, 47
- layers, 42–46
- libraries
 - installation, 17
- Linux
 - HostGIS Linux, 8, 13
 - Ubuntu, 8, 13
- Maintenance, 39
- map
 - images, 11
 - rendering engine, 11
- Mapbender, 11
 - configuration, 31
 - gui, 33
 - OWS Proxy Server, 26
 - WMS, 31
- Mapbender folders, 26
- mapping
 - engine, 11
 - framework, 11
- mapping framework, 11
- MapServer, 11, 16
 - cartography, 30
 - configuration, 30, 31
 - GIS data layer, 30
 - installation, 16
 - map file, 30, 40
 - query template, 41
 - templates, 32
 - WMS, 31
- OCLA
 - add users, 32
 - administration, 32, 34
 - data, 29
 - data base tables, 29
 - installation, 34
 - support, 39
- Open Source Geospatial Foundation, 39
- OpenLayers, 12
- operating system, 8, 13
 - HostGIS Linux, 8, 13
 - Ubuntu, 8, 13
- OSGeo, 39
- OWS Proxy Server, 26
- PHP, 8, 11, 17
 - configuration, 21
 - installation, 17
 - performance, 21
- PostGIS, 9, 16
 - backup, 30
 - data, 29
 - installation, 16
 - loading data, 27
- PostgreSQL, 9, 16
 - backup, 30
 - installation, 16
- Poverty per Tract, 46
- programming languages, 8
 - JavaScript, 9, 11
 - PHP, 8, 11
- Proj4, 17
 - installation, 17

- query
 - engine, 32
- query builder and report tool, 32
- Reference books
 - PostGIS, 39
- resources, 39
- security monitoring, 6
- Server
 - performance, 21
- shape file, 9, 11, 28
- software
 - Apache, 20
 - ExtJS, 17
 - GDAL/OGR, 17
 - GIS, 9
 - libraries, 17
 - Mapbender, 11
 - MapServer, 11
 - PostGIS, 9
 - PostgreSQL, 9
 - Proj4, 17
 - stack, 9
- support, 39
- system
 - architecture, 10
 - support, 39
 - users, 32
- tool
 - identify, 32
 - query, 32
 - report tool, 32
- Ubuntu, 8
 - GIS repository, 13
- US Census Data
 - Counties Year 2000, 42
 - Counties Year 2009, 44
 - Counties Year 2010, 44
 - Tracts Year 2000, 43
- users, 32
 - add, 32
- web
 - mapping framework, 18
 - server, 8, 13
 - web based gui, 33
 - Web Map Service, 31
 - WMS, 31
 - OWS Proxy Server, 26
 - secured, 26