

BALLOT PATH SERVER DOCUMENTATION

INTRODUCTION

AUDIENCE AND INTENTION

This document is intended for anyone who is responsible for ongoing implementation or maintenance of the Ballot Path project. It is not written for end users of the system. Anyone reading this document will be assumed to have some technical skill and experience, though it is not required. As the intent of this document is to provide a roadmap to building and maintaining the Ballot Path Application Server, not every detail of implementation is included. All of the tools used to build the server were deliberately chosen for their wide use and thorough documentation. We leave the details of that documentation to their respective owners. Links to these repositories will be included as a reference.

SKILLS REQUIRED

It is assumed that anyone reading this document will have some Linux administration skills. Though actual commands and procedures are included here, there are inevitably things that come up that will require resolution using information that is not included here. It is highly recommended that any future administrator of this system be familiar with:

- SSH
- Unix text editors (vi, vim, emacs, nano, etc.)
- Git
- Linux commands (sudo, apt, services, file systems, user administration)

TECHNOLOGY STACK

- Amazon AWS
- Python 2.7
- Flask
- Apache Web Server
- PostgreSQL
- SQLAlchemy
- Postgis
- Git

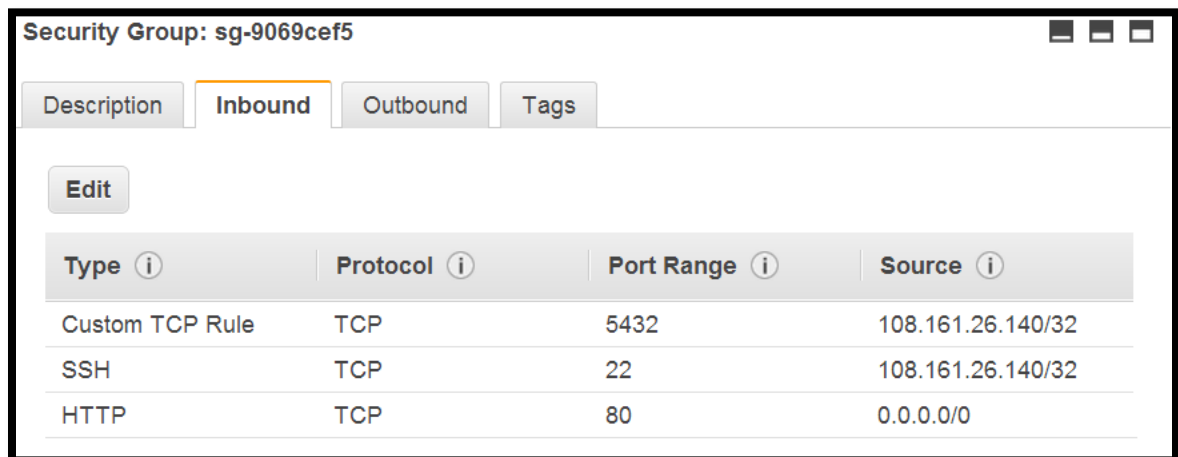
INSTALLATION

SETTING UP UBUNTU

Ballot Path runs on an Ubuntu (14.04 Trusty) virtual machine in the Amazon Web Services (AWS) environment. This document assumes that the user has an AWS account and a basic understanding of standing up server instances.

1. Set up a Security Group that includes rules for:
 - a. SSH (port 22)
 - b. HTTP (port 80)
 - c. Postgresql (custom TCP rule enabling port 5432)

When complete, the security group will look similar to this:

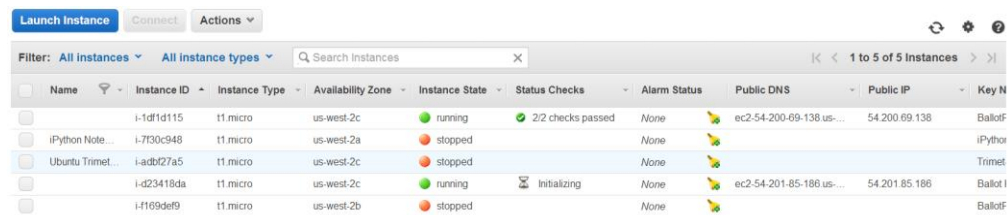


Security Warning: it is **HIGHLY RECOMMENDED** that source IP addresses not be open to the world. AWS is a constant target for malicious hackers seeking to exploit security vulnerabilities. The security group pictured above was created using the 'my IP' selection which limits access to a single IP address. As new developers are added, new IP addresses may be included in the security group rules. To include a range of addresses, just use a 0 in one of the octets. For example, if I want to allow access to anyone within the 108.161.26.x subnet, I would change the source IP address to 108.161.26.0/24.

2. Create a Key Pair
 - a. Click the Key Pairs link in the EC2 dashboard.
 - b. Create a new Key Pair by clicking the **Create Key Pair** button.
 - c. Name the key pair and download the .PEM file to your local computer. Save this key in a secure location. It is the keys to the Ballot Path kingdom. Take note of the location. You will need it later when connecting to the server.
3. Launch Server
 - a. Within the AWS EC2 Dashboard, click the **Launch Instance** button.
 - b. Select the **Amazon Linux AMI 2014.03.1** - ami-043a5034 (64-bit) instance.
 - c. Choose the instance type. The Free Tier was used for development, but it has bandwidth and computing limitations that are not suitable for production. Select

one that fits the current needs of the Ballot Path application. Click the **Review and Launch** button to continue.

- d. Click the Edit Security Groups link and choose or create the security group as described above.
- e. Once all options are complete, click the **Launch** button.
- f. Choose an existing or create a new key pair. Acknowledge the warning about access to the key file and click **Launch Instance**.
- g. The server is now being brought up. It takes a few minutes to build. Progress can be seen in the EC2 dashboard.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name
	i-1df1d115	t1.micro	us-west-2c	running	2/2 checks passed	None	ec2-54-200-69-138.us-...	54.200.69.138	BallotF
iPython Note...	i-730c948	t1.micro	us-west-2a	stopped		None			iPython
Ubuntu Trimet...	i-adbf27a5	t1.micro	us-west-2c	stopped		None			Trimet
	i-d23418da	t1.micro	us-west-2c	initializing		None	ec2-54-201-85-186.us-...	54.201.85.186	BallotI
	i-1f109def9	t1.micro	us-west-2b	stopped		None			BallotF

LOGGING IN WITH DEFAULT UBUNTU ACCOUNT

Once the server is initiated, you will need to log in via SSH. AWS does not allow access without using public key encryption (PKE). If you are running on a Mac or Linux system, you will have to configure your SSH client to use the key pair you downloaded. This document is limited to configuring a Windows system with Putty (an SSH client). Details on connecting can be found on Amazon's AWS web site at <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-connect-to-instance-linux.html>.

INSTALLING COMPONENTS

Once connected, the software stack must be installed and configured. This is done using SSH and apt-get commands. Configuration is done by editing configuration files on the server.

POSTGRESQL

```
sudo apt-get install postgresql postgresql-contrib postgresql-client postgres-  
xc build-dep python-psycpg2
```

Once installed, postgresql must be configured to listen to outside IP address requests. Edit the **/etc/postgresql/9.3/main/pg_hba.conf** file (you must use the sudo command). The postgresql version number in the path may vary.

```
sudo vi /etc/postgresql/9.3/main/pg_hba.conf
```

Add this line to the file. This will enable the postgresql server to trust requests from any client, anywhere in the world. This is a huge security hole and should only be done during development. It should later be modified or removed so as to limit access to specific IP addresses or only locally (as in the case of an integrated server that needs no outside SQL access).

```
host all all 0.0.0.0/0 trust
```

Next, edit the **/etc/postgresql/9.3/main/postgresql.conf** file.

```
sudo vi /etc/postgresql/9.3/main/postgresql.conf
```

Change this line

```
#listen_addresses = 'localhost'          # what IP address(es) to listen on;
```

to this:

```
listen_addresses = '*'                  # what IP address(es) to listen on;
```

We need to change the PostgreSQL postgres user password; we will not be able to access the server otherwise. As the "postgres" Linux user, we will execute the psql command.

```
sudo -u postgres psql postgres
```

Set a password for the "postgres" database role using the command:

```
\password postgres
```

and give your password when prompted. The password text will be hidden from the console for security purposes. Type Control+D to exit the posgreSQL prompt.

See <https://help.ubuntu.com/community/PostgreSQL> for additional details on installing and using postgresql on Ubuntu. For information on configuring postgresql access from external addresses, see <http://www.cyberciti.biz/tips/postgres-allow-remote-access-tcp-connection.html>.

POSTGIS

What I did to install it: <http://trac.osgeo.org/postgis/wiki/UsersWikiPostGIS21Ubuntu1310src>

1. Install the whole postgresql mess:

```
sudo apt-get install build-essential postgresql-9.3 postgresql-server-dev-9.3  
libxml2-dev libgdal-dev libproj-dev libjson0-dev xsltproc docbook-xsl docbook-  
mathml
```

2. Then, download, unpack, compile, and install Geos:

```
wget http://download.osgeo.org/geos/geos-3.4.2.tar.bz2  
tar xjf geos-3.4.2.tar.bz2  
cd geos-3.4.2  
./configure  
make  
sudo make install  
cd ..
```

3. Download, extract, and build postgis:

```
wget http://download.osgeo.org/postgis/source/postgis-2.1.3.tar.gz  
tar xzf postgis-2.1.3.tar.gz  
cd postgis-2.1.3
```

```
./configure
make
sudo make install
sudo ldconfig
sudo make comments-install
```

4. Lastly, enable the command-line tools to work from your shell:

```
sudo ln -sf /usr/share/postgresql-common/pg_wrapper /usr/local/bin/shp2pgsql
sudo ln -sf /usr/share/postgresql-common/pg_wrapper /usr/local/bin/pgsql2shp
sudo ln -sf /usr/share/postgresql-common/pg_wrapper /usr/local/bin/raster2pgsql
```

Related documentation:

PostGIS Documentation: <http://postgis.net/docs/manual-2.1/>

PostGIS install instructions for Ubuntu: <http://postgis.net/install/> and <http://www.qgis.org/en/site/>

PYTHON

Python 2.7.6 is installed by default with the Ubuntu AWS server instance. There is no need to install or upgrade.

FLASK

Flask installation is documented at <http://flask.pocoo.org/docs/installation/#installation>. I have used their instructions as a foundation, but I include specifics here.

```
sudo apt-get install python-virtualenv
```

Now that virtualenv installed, just fire up a shell and create your own environment. I usually create a project folder and a venv folder within:

```
$ mkdir myproject
$ cd myproject
$ virtualenv venv
New python executable in venv/bin/python
Installing distribute.....done.
```

Now, whenever you want to work on a project, you only have to activate the corresponding environment. On OS X and Linux, do the following:

```
$ . venv/bin/activate
```

Now you can just enter the following command to get Flask activated in your virtualenv:

```
$ pip install Flask
```

A few seconds later and you are good to go.

GIT

To install git

```
sudo apt-get install git
```

APACHE2

Install Apache using the apt-get command:

```
sudo apt-get install apache2
```

MAINTENANCE

ADDING USERS TO UBUNTU LINUX

Instructions here: <http://brianflove.com/2013/06/18/add-new-sudo-user-to-ec2-ubuntu/>

```
sudo adduser --home /home/mclyde --shell /bin/bash --ingroup admin mclyde
```

After creating the new user, let's give the user the ability to use the sudo command. I am going to make it so that the user doesn't need to enter their password after using the sudo command. Let's do this by using our favorite editor, vi.

```
sudo vi /etc/sudoers
```

Add the following line of code to the file. I added this after the root user in the "User privilege specification" section:

```
# User privilege specification  
mclyde ALL=(ALL) NOPASSWD:ALL
```

Now we have to set up public key encryption so that the user can log in. In a shell, su to the newly created user, change to their home directory, and generate a key:

```
su mclyde -  
cd  
ssh-keygen -t rsa
```

Then, create a directory (if not already created), put the public key string into .ssh/authorized_keys, and set permissions.

```
mkdir .ssh  
mv id_rsa.pub .ssh/authorized_keys  
chmod 700 .ssh  
chmod 600 .ssh/authorized_keys
```

Finally, copy the private key to whatever computer the user will be using. In the case of Putty on Windows, save the id_rsa file somewhere safe, load it into puttygen, and save the private key as a putty-compatible .PPK file. Use this file as the private key for logging in with Putty. See

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-connect-to-instance-linux.html> for details on configuring Putty and saving a private key.

ADDING USERS TO POSTGRESQL

There are only two users allowed to access the Postgresql server: postgres (password "Democracy!"), a fully administrative user account, and BallotPath (password "Democracy!"), a limited account for the web service. All other users will use these accounts for access.

TRANSFERRING THE AWS SERVER INSTANCE TO ANOTHER ACCOUNT

See <http://knackforge.com/blog/sivaji/how-move-aws-ec2-instance-one-account-another> for details.

PULLING SOURCE FROM GIT

All source code is stored in the Ballot Path GIT repository on Github.com

The Github account is at:

Pulling from the Github server is done from the server in a terminal window or via SSH from the /var/www/ directory.

STARTING AND STOPPING THE SERVER

RECOMMENDED TOOLS

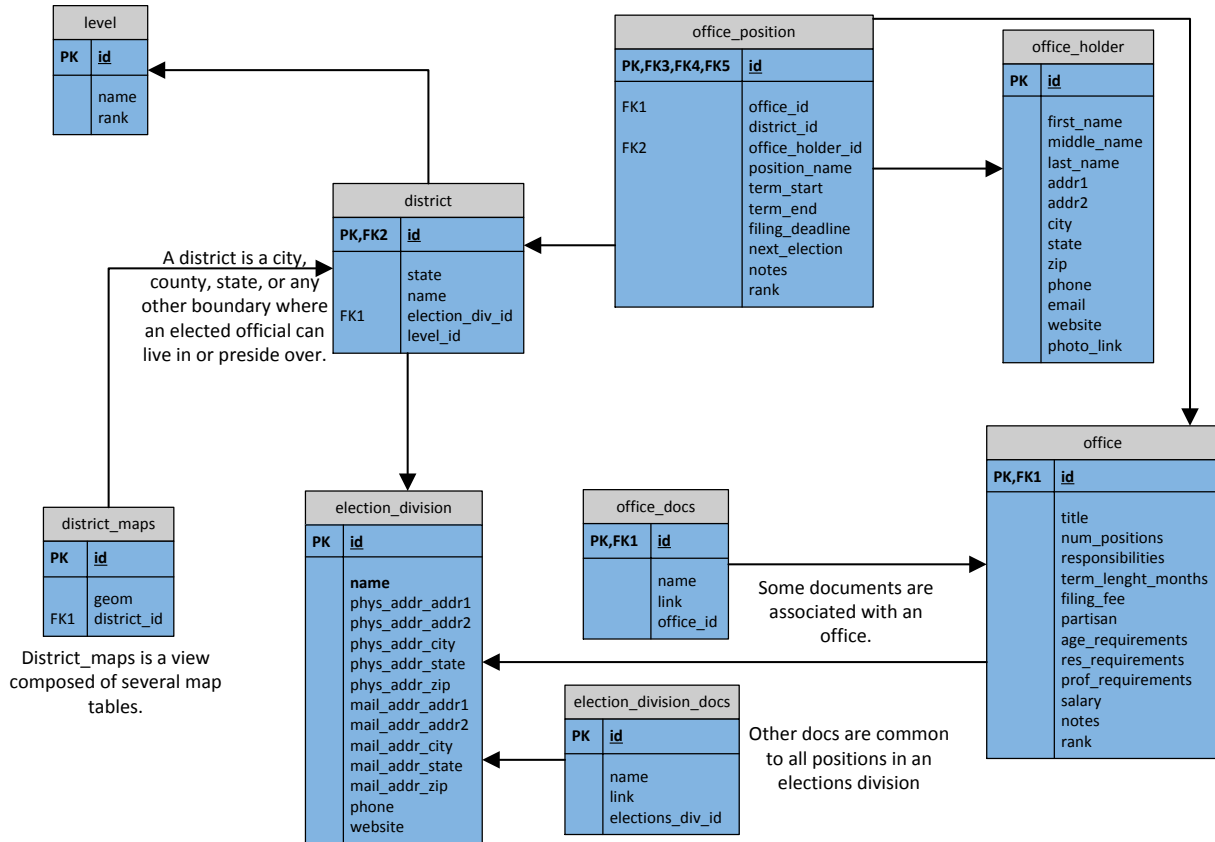
- pgAdmin III for database manipulation and maintenance
- Putty and/or mRemoteNG for SSH access to the Linux server
- WinSCP for transferring files between Linux and Windows

DATABASE INFORMATION

DESIGN

Ballot Path Database Schema

Federal, State, County,
Municipal, or Local



SETTING UP THE DEVELOPMENT ENVIRONMENT

VIRTUAL ENVIRONMENT SETUP:

First, create the virtual environment directory inside the `api/app` folder:

```
sudo virtualenv venv
```

Next, activate it so next installation steps will install into the virtual environment:

```
source venv/bin/activate
```

Then install the required Python packages:

```
pip install flask==0.9
pip install sqlalchemy==0.7.9
pip install flask-sqlalchemy==0.16
pip install Flask-Psycopg2
pip install flask-restful
pip install flask-simplerest
pip install csvvalidator
```



```
pip install Flask-HTTPAuth
pip install simplejson
```

Finally, install sendmail for the password recovery systems:

```
sudo apt-get install sendmail-bin
```

APACHE SETUP:

Make sure that you have **mod_wsgi** installed for Apache; this allows you to serve the Flask site. If you don't, you can get it by performing the following command:

```
sudo apt-get install libapache2-mod-wsgi
```

Also enable **mod_speling**:

```
sudo a2moden speling
```

Next, you need to set up your **Apache.conf** file. Go to `/etc/apache2/sites-available/` and change/add the following lines to the default config (e.g. **000-default.conf**):

```
CheckSpelling on
CheckCaseOnly on

DocumentRoot /var/www/BallotPath/html

Alias /BPAdmin "/var/www/admin"
<Directory /var/www/admin>
    Order allow,deny
    Allow from all
    Require all granted
    Header always set Access-Control-Allow-Origin "*"
    Header always set Access-Control-Allow-Headers "origin, x-
requested-with, content-type"
    Header always set Access-Control-Allow-Methods "GET, OPTIONS"
</Directory>
WSGIScriptAlias /api "/var/www/BallotPath/api/flaskapp.wsgi"
<Directory /var/www/BallotPath/api/app>
    Order allow,deny
    Allow from all
    Require all granted
    Header always set Access-Control-Allow-Origin "*"
```

```
Header always set Access-Control-Allow-Headers "origin, x-
requested-with, content-type"

Header always set Access-Control-Allow-Methods "GET, OPTIONS"

</Directory>
```

You could also use a similar Alias directory to serve the html directory if you already have your DocumentRoot set to somewhere else.

Next you'll need to move into the /var/www folder (create them if they don't already exist) and clone the git repo into there. First install git:

```
sudo apt-get install git
```

You'll need to have an account set up at www.github.com as well. Next, from the /var/www directory, clone the BallotPath repo:

```
sudo git clone https://github.com/mclyde/BallotPath
```

Enter the github username and password when prompted. The BallotPath directory should be there containing all of the necessary files and on the master branch. Do not change the branch unless you are working on a non-production server just for testing or development purposes.

Finally, you need to make sure the .wsgi file exists in the repo so Apache can serve the directory. Create this file in the api folder if it doesn't exist or edit it as necessary to contain the following:

```
#!/usr/bin/python
import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0, "/var/www/BallotPath/api/")

from app import app as application
```

Once you are done with all this, restart Apache so it loads the new files:

```
sudo service apache2 restart
```

THE ADMINISTRATOR INTERFACE

1. Access the administrator tools via the public-facing web site and clicking the Developer link at the top.
2. Use the login credentials set up and discussed outside of this document.

Please enter password to access this page

Login:
 *
Password:
 *
 [Password Reminder](#)

3. After logging in, the main menu will appear where you may perform several administrative functions.

Developer Access

[User Manager \(Administrator\)](#)

[Bulk Upload Page](#)

[Developer Database](#)

[Shape File Upload](#)

[Relation File Upload](#)

[Logout](#)

Menu Item Descriptions and Instructions

User Manager

This is used to add users to the system for maintenance. This should only be used to add members of the Ballot Path organization (and its trusted representatives). The public interface needs no credentials so this tool should not be used for the general public.

To add users, click on Administrator Control and log in using the Administrator credentials. Once you are logged in, users may be added in a simple “spreadsheet style” method. Simply add login, password, and email addresses for each user desired. This page may also be used to change the administrator password.

php include(/var/www/BallotPath/html/scr/users.php)

/var/www/BallotPath/html/scr/users.php

Login	Password	Email	Redirect URL	Mark
Andrew	Erland	andywde@gmail.com		<input type="checkbox"/>
Matthew	Clyde	mclyde@pdx.edu		<input type="checkbox"/>
Blake	Helm	helm.blake@gmail.com		<input type="checkbox"/>
Shawn	Forge	forgie@pdx.edu		<input type="checkbox"/>
Blake	Clough	eblake@pdx.edu		<input type="checkbox"/>
				<input checked="" type="checkbox"/>

Save Changes and Delete marked

New Administrator Password:

Bulk Upload

This tool is used to import spreadsheets full of bulk office/district/election data. Spreadsheets must be in the specific format supplied by the developers.

To upload a file, click the Upload a File link and browse to the CSV file you wish to upload. Click the **Upload** button.

Upload a File

No file chosen

1. Open spreadsheet in excel save a version/export as csv with delimiter '|'
2. Navigate to BulkUpload page from developer tab
3. Choose file and upload
4. If No bad_isnerts.csv is returned then it was successful
5. If bad_inserts.csv returned then review if the errors are non-issues (ie you know there should have been duplicate documents detected etc.) then its up to the importer to decide if it was successful
6. If bad_inserts.csv returned and errors are actual errors then open up excel sheet again and edit as necessary save as excel
7. Then export to csv with delimiter '|' and repeat 2-7 as necessary.

I want to emphasize opening and making edits through excel rather than the csv. It is possible to edit the csv directly but only when opening it as a plain text document in textedit etc. I have encountered issues opening the csv in Libre office, editing and then saving as csv again, columns are not lined up and the delimiter is not always recognized. I have had best results editing the excel and exporting as csv every time.

For additional details, see Appendix A at the end of this document.

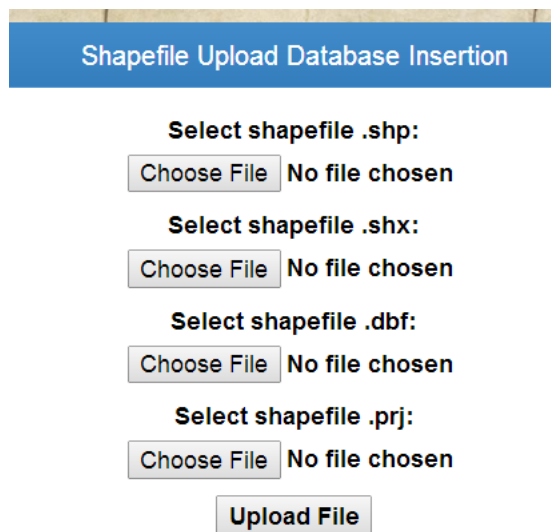
Developer Database

This is the menu-driven access to each of the main tables in the database. It is used to make small, one-time modifications and corrections to the data and should not be used for mass data entry, though it can be used this way in a pinch. Detailed instructions for this interface are given in a separate document, ***Using the BallotPath Database Administration Tool***.

Shape File Upload

This interface is used for importing GIS information (adding shape files).

After logging in to the system and selecting Shape File Upload, the interface below is presented.



The interface is titled "Shapefile Upload Database Insertion" in a blue header bar. Below the header, there are four sections, each with a label and two buttons:

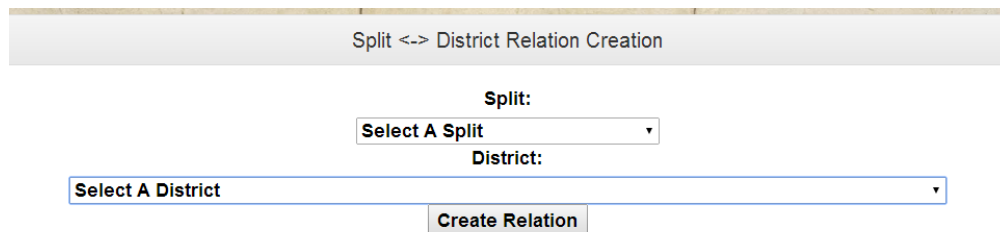
- Select shapefile .shp:** "Choose File" and "No file chosen"
- Select shapefile .shx:** "Choose File" and "No file chosen"
- Select shapefile .dbf:** "Choose File" and "No file chosen"
- Select shapefile .prj:** "Choose File" and "No file chosen"

At the bottom of these sections is a single "Upload File" button.

In order to insert shape information into the database, you must upload 4 files as noted above. Browse to each file of the archive, then click **Upload File**. The GIS information will be inserted into the database. You must then associate the shape file with a district, described in the next section, Relation File Upload.

Relation File Upload

Used as a way of creating a data relationship between GIS information and Bulk Upload data.



The interface is titled "Split <-> District Relation Creation" in a light gray header bar. Below the header, there are two dropdown menus and one button:

- Split:** A dropdown menu with "Select A Split" selected.
- District:** A dropdown menu with "Select A District" selected.
- Create Relation** button.

After uploading GIS information, the splits must be associated with districts in the database. Use this interface to make that association.

Appendix A: Bulk Upload Detailed Instructions and Information

Guidelines are as follows.

Inserting district or election division minimum requirements

- district_name
- district_state
- el_div_name
- phys_addr_state

Inserting office holders minimum requirements

- district_name
- district_state
- election_div_name
- position_name
- first_name, last_name

Inserting office position minimum requirements

- district_name
- district_state
- election_div_name
- position_name
- title

Inserting office minimum requirements

- district_name
- district_state
- election_div_name
- position_name
- title

Inserting office docs minimum requirements

- district_name
- district_state
- election_div_name
- position_name
- title
- office_doc_name
- office_doc_link

Inserting election docs minimum requirements

- district_name
- district_state
- election_div_name
- position_name
- title

- election_div_doc_name
- election_div_doc_link

When importing the Districts and Election Divisions spreadsheets must be inserted first otherwise any office, position or holder inserts referencing them will not be able to be saved to the database.

The process for adding documents is the same for election division docs and office docs. It is important that there is only one document per row. For example if adding an office holder, position and office with 3 documents A, B and C, the proper format would be as follows:

Row 1:

first_name, last_name, position_name, title, office_doc_name(For A), office_doc_link(For A),
district_name, state, election_div_name

Row 2:

position_name, title, office_doc_name(For B), office_doc_link(For B), district_name, state,
election_div_name

Row 3:

position_name, title, office_doc_name(For C), office_doc_link(For C), district_name, state,
election_div_name

If there are multiple positions for an office you can change the position_name to reflect that so you do not have to add extra rows for example position 1 and position 2 with 3 docs for an office:

Row 1:

first_name, last_name, position_name(For position1), title, office_doc_name(For A),
office_doc_link(For A), district_name, state, election_div_name

Row 2:

position_name(For position 2), title, office_doc_name(For B), office_doc_link(For B), district_name,
state, election_div_name

Row 3:

position_name(Position name 1 or 2), title, office_doc_name(For C), office_doc_link(For C),
district_name, state, election_div_name

Because office_documents only relate to the offices and not each individual position if there is only one document and multiple positions you only have to put the document in one row to tie it to the office. Election division docs are the same. The examples above do not have any other fields but you

can add any other data you want along with documents the examples are just to display the minimum requirements for inserting docs.

Between the two types of spreadsheets care must be taken to maintain exact spelling and relation between district names and election division names otherwise when inserting holders, positions and offices the import queries will not be able to make the proper connections.

- Bulk_Insert_District_Election_Division_Multnomah, Washington, Clackamas_docs_edited_2.xls and Multnomah_Spreadsheet_Template_(Successful_Import).xls are an example of successful import.
- Database_Variable_Types.rtf shows what types the database fields are expected to be.
- bad_inserts_2014-08-18-04_18_28.csv is an example of one type of validation report returned by the upload process, these errors are more type and field checks before the imports get to the database
- bad_inserts_Lincoln_County_Offices.csv is an example of the other type of validation report returned, these errors are from duplicate data detections made during the import process (in this case Lincoln County Offices had already been imported and this was a second run through the data). To make it a little easier to read open in excel with delimiter as '|' The top rows are usually the most important, for example with this document you will see each position as being detected as a duplicate and then for that row you would find another row detecting the office holder and office also as duplicates.
- bad_inserts_BENTON_DIV_DIST.csv is an example of when documents are attempted to be inserted multiple times, you can also reference how the election_div_docs were attempted to be added to see why there are so many duplicate detections (SEE Bulk_Insert_District_Election_Division_Benton REFORMATTED.xls). An Election_division only needs to have a unique document assigned to it once in this case the documents were repeated for every unique election_div -- district_name combination in the spreadsheet. This is wrong but because we detect it it really doesn't hurt it does ensure that the document has been assigned to the election division.

The general process for importing spreadsheets is as follows:

1. Open spreadsheet in excel save a version/export as csv with delimiter '|'
2. Navigate to BulkUpload page from developer tab
3. Choose file and upload
4. If No bad_isnerts.csv is returned then it was successful
5. If bad_inserts.csv returned then review if the errors are non-issues (ie you know there should have been duplicate documents detected etc.) then its up to the importer to decide if t was successful
6. If bad_inserts.csv returned and errors are actual errors then open up excel sheet again and edit as necessary save as excel
7. Then export to csv with delimiter '|' and repeat 2-7 as necessary.

I want to emphasize opening and making edits through excel rather than the csv. It is possible to edit the csv directly but only when opening it as a plain text document in textedit etc. I have encountered issues opening the csv in Libre office, editing and then saving as csv again, columns are not lined up and the delimiter is not always recognized. I have had best results editing the excel and exporting as csv every time.