# Wild Pockets Server Setup

The Wild Pockets server backend consists of two parts; the resource server, which is a set of Python scripts intended to be executed as CGI scripts via a web server, and the authentication server (Moneyserv), which is a daemon written in Python, intended to run persistently.

The two components can be deployed on the same server or on two different servers. This document is generally written with the assumption that you will deploy them on the same server, but separating them out isn’t much different.

## Requirements

Although all of the server components are written in a cross-platform language (Python), they have only been tested on Linux. It is unlikely that they will work without modification on Windows. They may work on BSD, but this is not tested.

They have also only been tested on a 32-bit architecture. They will probably work on a 64-bit architecture as well, but there are a few places in Python (particularly in the struct module, which we use) that you can get tripped up moving between the two. I think we are OK in that regards, but again, it has not been tested.

We have only tested the system on Ubuntu Hardy (which is what has been used in our production environment), but some adjustments have been made to account for Ubuntu Maverick and it should be okay. In general, most Linuxes should work; the most likely place you would get tripped up is if you have different versions of the Python modules we use.

Here’s a general list of the software you should need to run this:

* Python 2.5 or 2.6; 2.7 will probably work, but we haven’t tested. 2.4 appears to work as well (Ubuntu package ‘python’)
  + FormEncode module >= 1.2.2 (Ubuntu package ‘python-formencode’). I believe that the package in Ubuntu Hardy is out of date and we installed a more recent version manually. The package in Maverick seems fine.
  + PyCrypto module >= 2.0.1 (Ubuntu package ‘python-crypto’).
  + MySQLdb module >= 1.2.2 (Ubuntu package ‘python-mysqldb’).
  + Dateutil module >= 1.3 (Ubuntu package ‘python-dateutil’).
  + Magic module >= 4.21 (Ubuntu package ‘python-magic’). Corresponds to ‘file’ version 4.21. Note: this module seems fairly Ubuntu specific. I added a workaround so that systems lacking this module will still work.
* A mail server. You could use an external SMTP server or run a local mail daemon. We like postfix (Ubuntu package ‘postfix’) but it doesn’t really matter.
* MySQL >= 5.0 (Ubuntu package ‘mysql-server’). Unfortunately we’re pretty MySQL-specific right now; eventually I’d like to support other database engines, but for now you need MySQL.
* Apache 2.2 (Ubuntu package ‘apache2’). Actually any web server ought to do, but you need to support CGI and PHP execution. These instructions assume Apache.
  + PHP module >= 5.2.4 (Ubuntu package ‘libapache2-mod-php5’). You can, theoretically, separate the web server that hosts the resource (resource server) from the one hosting Embed.php, the login script, etc. In that case, the resource server does not need PHP, but the other one does.
* SSH (Ubuntu package ‘openssl-client’), for administering your server, mainly. In more esoteric configurations, the backend can also use SSH for building secure tunnels between separate hosts running Moneyserv, resource server, etc., for passing sensitive data over insecure links. However that isn’t covered in this document.

### Debian/Ubuntu

apt-get install python python-formencode python-crypto python-mysqldb \

python-dateutil python-magic postfix mysql-server apache2 \

libapache2-mod-php5 openssh-client

### RedHat/CentOS

yum groupinstall "Development Tools"

yum install python python-setuptools httpd mysql mysql-server php \

python-devel mysql-devel

easy\_install FormEncode

easy\_install hashlib

easy\_install pycrypto

easy\_install MySQL-python

easy\_install python-dateutil

Note: Redhat configuration has not been thoroughly tested. Expect bumps in the road.

## Create user account and group for authentication server

For security, the authentication server (Moneyserv) will run under its own user account (with its own group). The process for creating the group and account may differ depending on your flavor of Linux.

On Ubuntu:

groupadd –r moneyserv

useradd –r –m –g moneyserv moneyserv

This will create the group, create the user, and place the user in the group. If that doesn’t work, research the correct commands for your distribution.

## Creating directories, setting permissions

A number of directories need to be created to get off the ground. We will give the directory, the recommended permission bits, and recommended ownership. You can set this up with ‘mkdir’, ‘chmod’, and ‘chown’, respectively. If you aren’t sure how to do that, you should probably go read up on UNIX system administration before continuing.

Note: It is assumed that your web server (Apache) is running under UID:GID www-data. If it is not, you should, of course, adjust the permissions specified below as appropriate.

### Variable data (resource server content):

/var/wp root:root 0755

/var/wp/resources www-data:www-data 0755

/var/wp/thumbnails www-data:www-data 0755

### Configuration files:

/etc/wp root:root 0755

### Runtime data (PIDs, etc):

/var/run/wp root:root 0755

/var/run/wp/moneyserv moneyserv:moneyserv 0755

### Programs:

/srv/wp root:root 0755

/srv/wp/resourceserver root:root 0755

/srv/wp/moneyserv root:root 0755

## Install resource and authentication server software

Now that the directories are in place, you can upload and ‘install’ the resource and authentication server software. You won’t be able to run them, as we still need to set up configuration files and many other things, but the files can be put in place now.

### Resource server

* Upload contents of resourceserver/\* from the codebase to your server.
  + Make sure to configure your transfer client to upload .py (Python) files as ASCII text. If you do not, you may get mangled line feeds and the scripts won’t run. WinSCP will let you do this- check out the ‘Transfer’ tab in the preferences. FileZilla will NOT let you do this, so you should avoid it or find another way to correct the line feeds.
* Copy all of those files to /srv/wp/resourceserver
  + You will need to use recursion (cp -r) to do the copy because there are subdirectories
* Make all files and directories owned by root:root, mode 755 (chown -R root:root /srv/wp/resourceserver ; chmod -R 755 /srv/wp/resourceserver)

### Authentication server

* Upload contents of moneyserv/\* from the codebase to your server.
  + Again, make sure to upload .py files as ASCII text.
* Copy all of those files to /srv/wp/moneyserv
  + You will need to use recursion (cp -r) to do the copy because there are subdirectories
* Make all files and directories owned by root:root, mode 755 (chown -R root:root /srv/wp/moneyserv ; chmod -R 755 /srv/wp/moneyserv)

## Create database users and permissions

We will need two databases, one for the resource server, and one for the authentication server. We will call them ‘resourcedb’ and ‘moneyservdb’. We will also need two database users for the same. We will call them ‘resourceserver’ and ‘moneyserv’. Additionally, the two users will need permissions to their respective databases.

Note: if building from scratch on ubuntu, make sure to configure the postfix e-mail on install. E-mail is needed for the server to work correctly.

The following code snippet will set all of that up (you should be able to copy-paste it into a bash shell). You should change the passwords for the two users to your own secure values. Make note of the passwords that you use, as you will need to use them again later.

Note: when you issue these commands, you will be prompted for your MySQL root password. You should have that from when you installed MySQL. If not, you will have to reset your MySQL root password, a process which you should be able to find described on the internet.

mysql --user root --password <<'EOF'

CREATE DATABASE IF NOT EXISTS `resourcedb`;

USE `resourcedb`;

GRANT ALL PRIVILEGES ON `resourcedb`.\* TO 'resourceserver'@'localhost';

DROP USER 'resourceserver'@'localhost';

CREATE USER 'resourceserver'@'localhost' IDENTIFIED BY 'password1';

GRANT ALL PRIVILEGES ON `resourcedb`.\* TO 'resourceserver'@'localhost';

CREATE DATABASE IF NOT EXISTS `moneyservdb`;

USE `moneyservdb`;

GRANT ALL PRIVILEGES ON `moneyservdb`.\* TO 'moneyserv'@'localhost';

DROP USER 'moneyserv'@'localhost';

CREATE USER 'moneyserv'@'localhost' IDENTIFIED BY 'password2';

GRANT ALL PRIVILEGES ON `moneyservdb`.\* TO 'moneyserv'@'localhost';

EOF

You can also find this snippet in the codebase as serversetup/database/createdb.sh

## Load database schema

Now you need to load the schema (table definitions) for each database. To do this, upload the following two files from the ‘serversetup’ directory in the Wild Pockets codebase to your server:

* dbschema.resourcedb.sql
* dbschema.moneyservdb.sql

Now, you can issue the following commands to load the schemas:

mysql --user resourceserver \

--password resourcedb < serversetup/database/dbschema.resourcedb.sql

mysql --user moneyserv \

--password moneyservdb < serversetup/database/dbschema.moneyservdb.sql

You can also find this snippet in the codebase as serversetup/database/loadschema.sh

You could also load the schema as the root user, but doing it this way gives you the opportunity to ensure that the two users you set up in the last step are working correctly.

## Setting up configuration files

Configuration files need to be set up so that the various parts of the Wild Pockets backend can find each other, and so that clients can be told how to communicate back to the backend.

Create the following configuration files, as per the following templates. Also set ownership and permissions as specified.

### Resource server config (/etc/wp/resource.conf, www-data:www-data, 0640):

# /etc/wp/resource.conf

# Ownership should be www-data:www-data

# Permissions should be 0640

# Set the dbPass to the password you chose

dbUser='resourceserver'

dbPass='password1'

dbDb='resourcedb'

# If you are using an external SMTP server, change this value

smtpServer = '127.0.0.1'

# Set the e-mail addresses to a valid account

errorReportTo='support@yourdomain.com'

errorReportFrom='support@yourdomain.com'

# If you have configured Apache for SSL, you should set this to True.

requireSslForAuth=False

# If you are running the authentication server on a separate machine,

# then change this value.

moneyservHost = '127.0.0.1'

moneyservPort = 1234

You can also find this snippet in the codebase as serversetup/configsamples/resource.conf

Make sure to update dbPass to match the passwords set previously.

Note the ‘requireSslForAuth’ parameter. We strongly recommend that you acquire an SSL certificate and configure Apache to support SSL (https) connections. However this requires money and a valid domain name. Still, it is strongly recommended if you are going to be exposing your server to the world.

### Authentication server config (/etc/wp/moneyserv.conf, moneyserv:moneyserv, 0640):

# /etc/wp/moneyserv.conf

# Ownership should be moneyserv:moneyserv

# Permissions should be 0640

# Set MYSQL\_PASSWORD to the password you chose

MYSQL\_USER='moneyserv'

MYSQL\_PASSWORD='password2'

MYSQL\_DATABASE='moneyservdb'

EMAIL\_ADDRESS\_FROM='support@yourdomain.com'

YOUR\_DOMAIN='yourdomain.com'

You can also find this snippet in the codebase as serversetup/configsamples/moneyserv.conf

Make sure you change MYSQL\_PASSWORD to the password set for moneyserv previously.

### Web site config (/etc/wp/web.conf, www-data:www-data, 0640):

<?php

// /etc/wp/web.conf

// Ownership should be www-data:www-data

// Permissions should be 0640

// The location of your home page

define('WP\_HOME\_PAGE', 'http://www.yourdomain.com/');

// The location of the embed page

define('WP\_EMBED\_PAGE', 'http://www.yourdomain.com/Embed.php');

// Cookie domain; this could be the domain your site is hosted on, or it could be up one or more

// levels in the hierarchy, depending on how you lay things out.

define('WP\_COOKIE\_DOMAIN', 'yourdomain.com');

// The location of the WildPockets.js file

define('WP\_WILDPOCKETS\_JS', 'http://www.yourdomain.com/WildPockets.js');

// Passed to clients- the external hostname (or IP) of your resource server

define('WP\_RESOURCE\_SERVER', 'www.yourdomain.com');

// Passed to clients- the external hostname (or IP) of the web server hosting lookup files for the engine

define('WP\_LOADER\_LOOKUP\_SERVER', 'www.yourdomain.com');

// The hostname (or IP) to use for internal connections to the resource server.

define('WP\_RESOURCE\_SERVER\_INT', '127.0.0.1');

// If the resource server does not support SSL, then enable this flag

define('WP\_DISABLE\_SSL', true);

// If you are running the authentication server on a separate machine, then change this value.

define('WP\_MONEYSERV\_HOST', '127.0.0.1');

define('WP\_MONEYSERV\_PORT', '1234');

?>

You can also find this snippet in the codebase as serversetup/configsamples/web.conf

## Set up authentication server startup script

Now that all of the configuration files are in place, we are at a point where the authentication server is actually startable. To do this, you will need a startup script to place into /etc/init.d.

There are sample init scripts for Debian-like (Ubuntu) and RedHat-like operating systems in serversetup/initscripts in the codebase. Most systems will probably resemble one of these two.

Upload the appropriate init script to your system, copy it to /etc/init.d, and rename it to ‘moneyserv’. Set the permissions to 755 and ownership to root:root

Now you need to set the script up to run at the appropriate times. See the following overview for your OS:

### Debian/Ubuntu:

cp ~/serversetup/initscripts/moneyserv-debian.sh /etc/init.d/moneyserv

chown root:root /etc/init.d/moneyserv

chmod 755 /etc/init.d/moneyserv

update-rc.d moneyserv defaults 85 15

### RedHat/CentOS

cp ~/ serversetup/initscripts/moneyserv-redhat.sh /etc/init.d/moneyserv

chown root:root /etc/init.d/moneyserv

chmod 755 /etc/init.d/moneyserv

chkconfig –add moneyserv

chkconfig moneyserv on

Now, you can start the daemon. With any luck, you will get no errors and you will be able to see that moneyserv is running in a process list.

/etc/init.d/moneyserv start

ps auxw | grep moneyserv

## Set up Apache

Now that the moneyserv authentication server is running, we need to get the resource server in shape. To do this, you will need to configure Apache.

Unfortunately, Apache configurations vary widely, but I will provide some guidelines and a sample configuration block that ought to work on a vanilla Ubuntu system, and should put you on the right track for other systems.

You need to edit the configuration file for the appropriate ‘virtual host’ in your Apache install. In a vanilla install, there will probably only be one, default vhost. In Ubuntu, you’ll find it at /etc/apache2/sites-available/default. In RedHat, there does not seem to be a virtual host set up by default (I am not very familiar with RedHat)- you will need to dig in /etc/httpd/conf/httpd.conf

We need to add directives to do the following:

* Alias /srv/wp/resourceserver/ into the vhost directory space at /wp/, and allow script execution (ScriptAlias);
* Alias a number of scripts in that directory to shorter names that the client will look for;
* Alias the directory /var/wp/resources/ and /var/wp/thumbnails/ as /res/ and /thumbnails/ respectively;
* Set directory permissions appropriately to allow access to /srv/wp/resourceserver with script execution, and to allow access to /var/wp/resources and /var/wp/thumbnails without script execution

Here is a configuration snippet that will accomplish this:

ScriptAlias /wp/errorreport /srv/wp/resourceserver/errorreport.py

ScriptAlias /wp/fn /srv/wp/resourceserver/fn.py

ScriptAlias /wp/res /srv/wp/resourceserver/resource.py

ScriptAlias /wp/updatescript /srv/wp/resourceserver/updatescript.py

ScriptAlias /wp/ /srv/wp/resourceserver/

Alias /res/ /var/wp/resources/

Alias /thumbnails/ /var/wp/thumbnails/

<Directory "/srv/wp/resourceserver">

AllowOverride None

Options +ExecCGI -Indexes -FollowSymLinks

Order allow,deny

Allow from all

</Directory>

<Directory "/var/wp/resources">

AllowOverride None

Options -ExecCGI -Indexes -FollowSymLinks

Order allow,deny

Allow from all

</Directory>

<Directory "/var/wp/thumbnails">

AllowOverride None

Options -ExecCGI -Indexes -FollowSymLinks

Order allow,deny

Allow from all

</Directory>

You can also find this snippet in the codebase as serversetup/apache/configsnippet.txt

In a vanilla Apache installation, you can most likely literally just copy and paste that at the bottom of your VirtualHost block. In a more complex scenario, you will have to adapt things to match your setup, which is of course beyond the scope of this document.

A note about SSL: As mentioned earlier in this document, it is *strongly* recommended that you set up SSL on your Wild Pockets server. Doing so is beyond the scope of this document, however, if you have set up SSL, then you will most likely have two virtual host definitions, one for SSL and one without. In such a scenario, you will need to put the Wild Pockets directives into both vhosts, as Wild Pockets will use non-SSL connections for efficiency when anonymous requests are being made. The ‘requireSslForAuth’ directive in resourceserver.conf should then be set to True, and the ‘WP\_DISABLE\_SSL’ directive in web.conf should be set to False. Encryption will then be used appropriately.

Once you have the configuration block inserted, you should restart/reload Apache, and with any luck the resource server will be in business.

### Debian/Ubuntu

/etc/init.d/apache2 restart

### RedHat/CentOS

/etc/init.d/httpd restart

## Install sample website files

Now we need to install some files on your website to actually make this all accessible. For the purposes of this document, we are assuming that the website files will go onto the same server as the resource server- in fact, into the same virtual host. However, it ought to be possible to run them on a separate server, as long as it also has an appropriately-configured web.conf.

We’re going to put the following components up on the server:

* The Wild Pockets sample website, which includes a simple login script, an embed script, and a script to access the builder;
* A WildPockets.js, used by the scripts, and configured appropriately for your site;
* Binaries of the Wild Pockets loader for IE, Firefox and Chrome;
* Binaries of the Wild Pockets engine;
* ‘Lookup’ files so that the loader can find the engine(s).

For now we will use precompiled binaries and a pre-built WildPockets.js; if you rebuild the loader/engine from source, you will get new binaries and a new WildPockets.js file that you can use instead.

These files will all need to go into your ‘web root’, which, in a vanilla Apache installation, is /var/www. If your Apache configuration is custom, use the appropriate directory for your site instead.

* Upload the contents of websites/\* from the codebase to your server. Copy all the files to /var/www, and make sure all files are mode 644.
  + This will give you the login/embed/builder webpages and their supporting files.
* EITHER:
  + Upload the contents of prebuilt/\* from the codebase to your server. Copy all the files to /var/www, and make sure all files are mode 644.
    - This will give you the loaders, an engine, and a lookup file to tie them together, as well as a WildPockets.js file.
    - NOTE: if you are using files from the prebuilt folder. You must customize Wildpockets.js, and the files in the lookup directory. These will be autoconfigured if you used releaseBuild.py to build a personal engine.
  + OR if you’ve created your own release build of the engine. Upload the contents of buildtools/\* from the codebase to your server. And copy the files as follows:
    - dest/xml/\* -> /var/www/lookup/
    - dest/WildPockets.js -> /var/www/
    - dest/WildPocketsLoader\* -> /var/www/common/
    - dest/InstallWildPocketsPlugin\* -> /var/www/common/
    - dest/<<Engine Version>> -> /var/www/engines/

Your file hierarchy should now look something like the following:

www

|- EmbedImages

|- many files in here…

|- ...

|- builder.php

|- common.php

|- Embed.php

|- EmbedUtils.php

|- login.php

|- moneyserv.php

|- WildPockets.js

|- WildPocketsLoader-15079.cab

|- WildPocketsLoader-15079.xpi

|- InstallWildPocketsPlugin-1.0.9.15079.exe

|- lookup

|- 1.6

|- engines

|- 1.6

|- 1.6.6.16700

|- WildPocketsEngine.exe

Now, you need to modify some files to tie this all together.

* Edit WildPockets.js
  + Modify the block of variables at the top of the file to match the *version* and *URLs* for the loader that you are putting online. If you are using the prebuilt loader, the version numbers are probably already correct (since WildPockets.js came from the same directory). The URLs, of course, you will need to customize to reflect your own site.
* Modify the lookup file(s)
  + For each major version of the loader you are placing online, you will need to modify the lookup file in the lookup/ subdirectory. For example, lookup/1.6.
  + The lookup files are just XML. Modify the engine/files/file/url tag to reflect the URL of the engine that you are placing online for that version. For example, the 1.6 lookup file would presumably reference an engine in the engines/1.6/ subdirectory.

This should now have tied the webpage to the loader via WildPockets.js, and the loader to the engine via the lookup file(s).

## Creating system accounts

Now that the authentication server, resource server, and the sample website are up and running, you need to prime the Wild Pockets system before it is usable. The first step of this is to create two system accounts that Wild Pockets requires. You can do this with the (hopefully) now-functional website login script.

Access <http://yourserver/login.php> in your browser.

Use the “Sign up” form to create two new accounts:

* kernel
* Wild Pockets Team

Unfortunately the authentication backend will require you to use a separate e-mail address for each thread. Many mail systems (such as Gmail) will allow you to alias mail accounts by using a plus sign, for example, [myaccount+kernel@gmail.com](mailto:myaccount+kernel@gmail.com) and [myaccount+wpteam@gmail.com](mailto:myaccount+wpteam@gmail.com) would be separate addresses but go to the same mailbox. So you could use this to sidestep the requirement. Otherwise, you will need to get two e-mail accounts to use.

When you sign up, the authentication server should send you an e-mail with the initial password. If you don’t get it, check that moneyserv is running (/etc/init.d/moneyserv restart) and check that you have configured your SMTP server and e-mail addresses correctly in /etc/wp/moneyserv.conf. Check your system logs for more information if needed.

Once you have the initial passwords for the two accounts, you should log in and change the passwords to something secure. Be sure to make note of the new passwords, as you will need them later.

## Uploading core files

The Wild Pockets builder needs a number of files to get up and running (art assets, scripts, samples, etc.) These files must be pushed up to the server before the builder will work- unfortunately, the standard method of uploading them with the builder, is, of course, not going to work!

There is a special script at serversetup/tools/pushDirectory.py – you will need to invoke it *on your PC* (not the server) as follows:

X:\Python27\Python.exe X:\wildpockets\serversetup\tools\pushDirectory.py

--no-ssl --logfile=uploadlog.txt

X:\wildpockets\baseresources\wild pockets team

yourserver.yourdomain.com "Wild Pockets Team" <password>

Of course, you should adjust the paths as appropriate, specify the correct hostname for your server, and specify the password that you set for the Wild Pockets Team account. Also, if your server supports SSL, you should remove the --no-ssl parameter.

The upload process will take quite some time; it is uploading a lot of files. When it is done, you should get an indication of whether or not the upload was successful- hopefully it was, but if not, you should be able to glean some information from uploadlog.txt.

Once the upload is finished, you will need to give everyone permission to view the files in Wild Pockets Team. To accomplish this normally you can use the builder itself, however, for the first time you don’t have access to use the builder. To solve this use the script at serversetup/tools/makePublic.py. Again, you will need to invoke it from your PC.

X:\Python27\Python.exe X:\wildpockets\serversetup\tools\makePublic.py

--no-ssl --logfile=uploadlog.txt yourserver.yourdomain.com

"Wild Pockets Team" <password>

## Conclusion

That should be it! At this point, you should be up and running. If you’re running into difficulties, the first thing to do is start checking logs- all of the server components will be writing to logs (generally syslog or the Apache error log) and you may be able to learn something from that.

From here, there are many things you can do. The sample webpages are very much just samples; depending on your needs, they may be just fine, or you may want to explore extending them with new capabilities.

## Conclusion

That should be it! At this point, you should be up and running. If you’re running into difficulties, the first thing to do is start checking logs- all of the server components will be writing to logs (generally syslog or the Apache error log) and you may be able to learn something from that.

From here, there are many things you can do. The sample webpages are very much just samples; depending on your needs, they may be just fine, or you may want to explore extending them with new capabilities.