Postal
Push Notification Server

Christian Hergert
christian@catch.com

November 2012

Postal is a free and open source push notification server written in C.

1 Introduction

Postal is a Push Notification Server. It supports device management and delivery of push notifications to *Apple Push Notification Service (APS)*, *Google Cloud to Device Messaging (C2DM)*, and *Google Cloud Messaging (GCM)*. These are the services currently supported by the Apple iOS[1] and Google Android[2] mobile device platforms.

Postal is a server that you run as part of your infrastructure. You interact with it over an HTTP REST API. It stores and manages devices using MongoDB[3]. Optionally, you can receive events about events within Postal via the publish-subscribe mechanism of Redis[4].

[1] Apple and iOS are registered trademarks of Apple, Inc.
[2] Google and Android are registered trademarks of Google.
[3] For more information on MongoDB, visit http://www.mongodb.org.
[4] For more information on Redis, visit http://redis.io

2 Installation

Postal is designed to run on GNU/Linux, however other operating systems may be supported.

2.1 Installing from Source

You can fetch the most recent version of Postal from https://github.com/catch/postal/downloads. Installation follows the typical procedures for Free and Open Source software on GNU/Linux.

Many of the configuration options seen below are not required but illustrate their use. See `./configure --help` for more options.

```
tar --lzma -xf postal-0.2.0.tar.xz
cd postal-0.2.0
./configure --prefix=/usr --libdir=/usr/lib64 --
   sysconfdir=/etc/postal --enable-redis=yes --enable-
   debug=minimal --enable-trace=no
make
sudo make install
```

Figure 1: Installing Postal

2.2 Debian

Debian packages are not available at this time, but are planned.

2.3 Ubuntu 12.04 LTS

Ubuntu packages are not available at this time, but are planned.

2.4 Fedora 18

Fedora packages are not available at this time, but are planned.

3 Configuration

Postal uses a standard `key-file` format for configuration. This is similar to an `.ini` format from other Operating Systems.

Each configuration section is provided below with their options.

3.1 http

This section provides configuration of the HTTP REST API. The embedded HTTP is how external systems communicate with Postal.

Option	Value	Description
nologging	`true` or `false`	If HTTP logging is disabled.
port	`5300`	The TCP port to listen on.
logfile	`/var/log/postal/access.log`	The HTTP access logfile.

Figure 2: HTTP Configuration Options

See the example below.

```
[http]
nologging = false
port = 8080
logfile = /var/log/postal/access.log
```

Figure 3: HTTP Configuration Example

3.2 mongo

This section provides configuration for communicating with MongoDB. You are responsible for configuring and managing your MongoDB server.

It would be wise to have a *replicaSet* configured to prevent loosing data!

Option	Value	Description
db	*myapp*	The name of the MongoDb database.
collection	*devices*	The collection to store devices in.
uri	*mongodb://localhost/?w=2*	The MongoDb connection URI.

Figure 4: Mongo Configuration Options

See the example below.

```
[mongo]
db = myapp
collection = devices
uri = mongodb://localhost/?replicaSet=myrep&journal=true&
   w=2&wtimeoutMS=5000
```

Figure 5: Mongo Configuration Example

3.3 aps

The Apple Push Notification System requires that you use client SSL certificates to communicate with their gateway. You will need both the private key and certificate available in PEM format to communicate with the Apple Push Notification gateway.

TODO - Add section on how to generate sub keys and publish to Apple.

See the example below.

Option	Value	Description
ssl-cert-file	`cert.pem`	Path to SSL certificate.
ssl-key-file	`key.pem`	Path to SSL private key.
sandbox	`true` or `false`	Connect to sandbox gateway.

Figure 6: Apple Push Notification Configuration Options

```
[aps]
ssl-cert-file = /etc/ssl/certs/aps.pem
ssl-key-file = /etc/ssl/private/aps.pem
sandbox = false
```

Figure 7: Apple Push Notification Configuration Example

3.4 c2dm

Google's C2DM service requires an old Google Client style auth token. You can find out more about generating one of these tokens at https://developers.google.com/gdata/articles/using_cURL#authenticating-clientlogin.

Option	Value	Description
auth-token	`my_auth_token`	A Google Client Auth token.

Figure 8: Google C2DM Configuration Options

See the example below.

```
[c2dm]
auth-token = 12345678901234567890123456789O
```

Figure 9: Google C2DM Configuration Example

3.5 gcm

Google's GCM requires an authorization token. You can find out more about generating one of these tokens at http://developer.android.com/guide/google/gcm/gs.html.

Option	Value	Description
auth-token	`my_auth_key`	A Google Authorization Key.

Figure 10: Google GCM Configuration Options

See the example below.

```
[gcm]
auth-token = abcdefghijklmopqrstuvwxyz
```

Figure 11: Google GCM Configuration Example

4 REST API

TODO Describe REST API.

4.1 Creating and Managing Devices

TODO This format is not finalized! We will allow more fields soon.

Allowed `device_type` values.

- aps
- c2dm
- gcm

```
PUT /users/:user_id/devices/:device_token
Content-Type: application/json

{
  "device_type": "c2dm"
}

HTTP/1.1 201 Created
Content-Type: application/json
Location: /v1/users/:user_id/devices/:device_token
```

```
{
  "device_token": ":device_token",
  "device_type": "c2dm",
  "removed_at": null,
  "user": ":user_id"
}
```

4.1.1 Removing a Device

```
DELETE /users/:user_id/devices/:device_token

HTTP/1.1 204 No Content
```

4.2 Sending Notifications

TODO This format is not finalized!

```
POST /v1/notify
Content-Type: application/json

{
  "collapse_key": "abcdefg",

  "aps": {
    "alert": "",
    "badge": 0,
    "sound": "",
    "key": "value",
    "key2": 123.45,
    "key3": 456,
    "key4": true,
    "key5": [{"abc": "def"}]
  },

  "c2dm": {
    "key": "value",
    "key2": 123.45,
    "key3": 456,
```

```
    "key4": true
  },

  "gcm": {
    "delay_while_idle": false,
    "dry_run": false,
    "time_to_live": 86400,
    "data": {
      "key": "value",
      "key2": 123.45,
      "key3": 456,
      "key4": true,
      "key5": [{"abc": "def"}]
    }
  },

  "users": ["000011110000111100001111"],
  "devices": []
}

HTTP/1.1 200 OK
```

5 Metrics

Many system operators will want to know what is going on with their system. Postal provides a mechanism for being notified about events happening within the system. This mechanism uses a `PUBSUB` channel via `Redis` to deliver JSON formatted events. This feature requires that you configure Postal with `--enable-redis=yes`.

5.1 Configuration

Metrics delivery via `Redis` provides the following configuration options.

The options should be stored in the `redis` group in `postal.conf`. See the example below.

Option	Value	Description
enabled	`true` or `false`	If metric delivery is enabled.
host	*hostname*	The hostname of the Redis server.
port	6379	The port of the Redis server.
channel	*event:postal*	The Redis PUBSUB channel name.

Figure 12: Redis Configuration Options

```
[redis]
enabled = true
host = localhost
port = 6379
channel = events:postal
```

Figure 13: Redis Configuration Example

5.2 Events

The following sections describe the various types of events that you may receive.

5.2.1 Device Added

This event is published when Postal has received a request to add a new device.

```
{
  "Action": "device-added",
  "DeviceType": "gcm",
  "DeviceToken": "12341234-12341234",
  "User": "000000000000111122223333"
}
```

5.2.2 Device Updated

This event is published when Postal has updated the information for a device. This could happen if an API request has been processed to change the attributes of the device.

```
{
  "Action": "device-updated",
  "DeviceType": "gcm",
  "DeviceToken": "12341234-12341234",
  "User": "00000000000000111122223333"
}
```

5.2.3 Device Removed

This event is published when Postal has been requested to remove a device. This could happen from an API request that specifically requests the removal of a device. This could also happen if push notifications have been disabled for a device. Additionally, with APS, this could happen if the feedback service has requested the removal of a device.

```
{
  "Action": "device-removed",
  "DeviceType": "gcm",
  "DeviceToken": "12341234-12341234",
  "User": "00000000000000111122223333"
}
```

5.2.4 Device Notified

This event is published when Postal has successfully delivered a notification to the gateway. This does not mean that the device has received the notification. The various gateways to not provide this information so that is not available.

```
{
  "Action": "device-notified",
  "DeviceType": "gcm",
  "DeviceToken": "12341234-12341234",
  "User": "00000000000000111122223333"
}
```