Push Notification Manager Documentation

最後修改日期: 2018.2.21

目錄

1. 目的

此份文件的目的是在說明 Push Notification Manager (簡稱 PNM) 的安裝方式及其使用、設定相關細節.

2. PNM 介紹

PNM 是基於 CentOS 7.4 上開發出來的一支程式, PNM的全名是 Push Notification Manager, 其主要的工作內容是將 CTS 的 IoT Gateway 所發送出來的事件, 依據使用者在 App 上所訂閱之事件篩選後, 透過 Google Firebase Cloud Messaging (簡稱 FCM) 服務將訊息推送到 CTS 的 App 端呈現給使用者.

3. PNM 功能

PNM 支援以下功能:

. UID 白名單批次新增、修改、刪除

. UID 有效日期管理

. App Token 管理 (依據 FCM 回覆, 自動刪除無效之 Token)

. App 事件訂閱管理

. 事件紀錄

. 事件之訊息推送成功數及失敗數

4. PNM 用到的 Open Source

PNM 主要用到了以下幾個 Open source:

. OpenSSL library (Apache 2.0): 用來與 Google FCM server 溝通之連線 HTTPS.

. SQLite library (Public domain): 儲存事件、App Token 之資料庫.

. JSON-C library (MIT): 與 GW 和 App 溝通內容之格式.

. Libev library (BSD): 異步事件處理 library. 用來提高 PNM 之執行效率及處理性能.

. Tiny-AES source code (Public domain): 與 GW 和 App 溝通內容之加密方式.

5. PNM 原始碼架構

PNM 原始碼中 C code 的架構如下:

. aes.c: AES 加解密.

. fcm_helper.c: Google FCM 推送相關之實作.

. json_helper.c: 解析 JSON 格式訊息內容之實作.

. management.c: Admin 管理功能之實作. 如批次新增、刪除、修改白名單內容.

. pnm.c: PNM 之主程序、邏輯實現.

. sqlite3_helper.c: 資料庫相關操作之實作.

. tls_connect.c: 與 FCM server 之間的 SSL/TLS 連線實作.

. utility.c: 一些通用小工具之實現.

6. 安裝 PNM 以及其所需 libraries

PNM 是基於 CentOS 7.4 上開發出來的, 因此在安裝 PNM 前, 需要先準備一台 CentOS 7.4 之 server. (PNM 與 OS 之相關性非常低, 且其原始碼是純用 C 語言實現, 因此非常容易移植到其他 Unix-based 的 OS)

6.1. 安裝 openssl 開發用之標頭檔

. 執行 *sudo yum install -y openssel-devel*

6.2. 安裝 SQLite

. 執行 *mkdir -p /home/userA/pnm/contrib-src* 在使用者家目錄下新增 pnm 及 contrib-src 目錄 (假設當前使用者名稱為 *userA*)

. 執行 *cp sqlite-autoconf-3220000.tar.gz ~/pnm/contrib-src* 將 sqlite-autoconf-3220000.tar.gz 複製到 ~/pnm/contrib-src 目錄

. 執行 cd ~/pnm/contrib-src 切換到該目錄

. 執行 *tar -zxvf sqlite-autoconf-3220000.tar.gz* 解開壓縮檔

. 執行 *cd sqlite-autoconf-3220000* 進入該目錄

. 執行 ./configure --prefix=/home/userA/pnm/contrib

. 執行 *make && make install* 編譯並將所產生之 libraries 安裝至 ~/pnm/contrib 目錄

6.3. 安裝 Libev

. 執行 *cp libev-master.zip ~/pnm/contrib-src* 將 libev-master.zip 複製到 ~/pnm/contrib-src 目錄

. 執行 cd ~/pnm/contrib-src 切換到該目錄

. 執行 *unzip libev-master.zip* 解開壓縮檔

. 執行 *cd libev-master* 進入該目錄

. 執行 ./configure --prefix=/home/userA/pnm/contrib

. 執行 *make && make install* 編譯並將所產生之 libraries 安裝至 ~/pnm/contrib 目錄

6.4. 安裝 JSON-C

. 執行 *cp json-c-master.zip ~/pnm/contrib-src* 將 json-c-master.zip 複製到 ~/pnm/contrib-src 目錄

. 執行 cd ~/pnm/contrib-src 切換到該目錄

. 執行 *unzip json-c-master.zip* 解開壓縮檔

. 執行 *cd json-c-master* 進入該目錄

. 執行 *sh autogen.sh*

. 執行 ./configure --prefix=/home/userA/pnm/contrib

. 執行 *make && make install* 編譯並將所產生之 libraries 安裝至 ~/pnm/contrib 目錄

6.5. Tiny-AES

. Tiny-AES 是以原始碼的方式整合進 PNM, 因此不需要另外安裝.

6.6. 編譯並安裝 PNM

. 執行 *cp pnm_20180221.tar.gz ~/pnm* 將 pnm_20180221.tar.gz 複製到 ~/pnm 目錄, 檔名請依實際拿到之檔案更改.

. 執行 cd ~/pnm 切換到該目錄

. 執行 *tar -zxvf pnm_20180221.tar.gz* 解開壓縮檔

. 執行 *cd src* 進入該目錄

. 執行 *make* 編譯, 完成後不需安裝即可使用

6.7. PNM 啟動及停止

. 在 ~/pnm/src 目錄下, 執行 *./pnm* 即可啟動 PNM

. 如果要停止 PNM, 必須用 *kill -15 pidOfPNM*, 不可以用 *kill -9 pidOfPNM*, 因為這樣會造成 db 資料遺失或損毀.

7. 資料庫

PNM 用到了以下幾個 SQLite 資料庫, 資料庫路徑必須為 pnm 執行檔同一層的 db 目錄內.

7.1. config.db

config.db 儲存了 PNM 非常重要的資訊, 包含了用來與 Google FCM 溝通的 server key, 還有 socket listen 的 port number 以及一個 UID 最多可以同時支援幾隻手機的 push.

其欄位說明如下:

param: TEXT (Primary key).

value: TEXT.

主要有 3 筆資料, 第一筆 param 為 *fcmServerKey*, value 為 FCM 的 server key. 第二筆 param 為 *listenPort*, value 為 51139 (default port number), 第三筆 param 為 *maxNumOfTokensPerUid*, value 為 10 (default value).

7.2. whiteList.db

whiteList.db 儲存了合法的 UID資訊, 包含了UID, Key, Key1, activateDate, expiryDate, enabled.

其欄位說明如下:

uid: TEXT (Primary key), GW 的 UID.

key: TEXT (Primary key), AES 用來加解密的 key.

key1: TEXT (Primary key), 為了避免將 uid 和 key 以明碼的方式放在封包標頭內, 以 key1取代.

activateDate: INTEGER, Epoch time.

enabled: INTEGER, 0 代表此筆資料未啟用, 如同 UID不在白名單內.

expiryDate: INTEGER, Epoch time.

7.3. subscriptions.db

subscriptions.db 儲存了 App針對個別 GW/UID 所訂閱的所有事件以及其相關資訊.

其欄位說明如下:

uidindex: INTEGER (Primary key), 對應真實 UID 在 uids.db 內的索引號.

tokenindex: INTEGER (Primary key), 對應真實 Token 在 tokens.db 內的索引號.

gwname: TEXT, 存在 App 內的 GW 的名稱.

devid: INTEGER (Primary key), device 的 node id.

evclass: TEXT (Primary key), 訂閱的事件類別.

evtype: TEXT (Primary key), 訂閱的事件種類.

subtime: INTEGER, Epoch time, App 訂閱此事件的時間.

7.4. publishRecords.db

publishRecords.db 儲存了 GW 發出的每筆事件紀錄以及推送到 App 端成功及失敗的個數.

其欄位說明如下:

uid: TEXT (Primary key), GW 的 UID.

pubtime: INTEGER (Primary key), Epoch time, GW 發出此事件的時間.

roomid: INTEGER, room 的 id.

roomname: TEXT, room 的名稱.

devid: INTEGER, device 的 node id.

devname: TEXT, device 的名稱.

evclass: TEXT, 觸發的事件類別.

evtype: TEXT, 觸發的事件種類.

seq: INTEGER (Primary key), GW 發出事件的序列號.

successCount: INTEGER, 此事件推送到 App 的成功個數.

failureCount: INTEGER, 此事件推送到 App 的失敗個數.

7.5. tokens.db

tokens.db 儲存了 App 從 FCM server 取得的 Token 資訊.

其欄位說明如下:

tokenindex: INTEGER (Primary key), Token 在 tokens.db 內的索引號.

token: TEXT (UNIQUE), 真實的 Token.

provider: TEXT, iOS 或是 android. 目前尚未用到此欄位. 以後可以分析 iOS 和 Android 手機數量用.

7.6. uids.db

uids.db 儲存了 UID 和 對應的 uidindex 資訊.

其欄位說明如下:

uidindex: INTEGER (Primary key), UID 在 uids.db 內的索引號.

uid: TEXT (UNIQUE), 真實的 UID.

7.7. toBeSent.db

toBeSent.db 儲存了尚未發出的事件內容. 由於 PNM 可能會在很短時間內收到大量的事件, 造成在短時間無法將所有事件全部發送給 FCM server (HTTPS), 因此尚未發出的事件會暫時的儲存在此 db, 一但發送出去後, 會立即被移除.

其欄位說明如下:

msg: TEXT, 需要被推送出去的內容, JSON 格式.

msg_len: INTEGER, 需要被推送出去的內容長度.

addedTime: INTEGER, 此訊息被加入此 db 的時間, Epoch time, PNM 會依照順序從最舊的開始依序發出.

isSent: INTEGER, default 為0, 發出後會改為 1, 然後等待被移除.

publishRecord_rowid: INTEGER, 此訊息當初是根據哪個事件所產生的, 其 rowid. 用意是為了更新 App 發送後, 成功或失敗的個數.

tokenindex: INTEGER, 當訊息發送後, 如果 FCM server 返回此 Token 已經失效 (ex. App 被移除), PNM 會用此 tokenindex 來反查出真實的 Token, 並將它移除, 以避免之後再次推送到無效的 Token, 浪費系統資源及頻寬.

7.8. 資料庫備份

由於 SQLite 的特性, 一個檔案即為一個完整的資料庫, 因此備份極為單純, 只要將上述幾個 db 直接複製, 即可備份完成.

7.9. 資料庫還原

還原資料庫,只要將備份的 db 覆蓋當前的, 即可完成. 但還原之前請務必先將 PNM 停止.

8. Admin 批次管理

由於 White list 是極為重要的資料, 因此此操作只限於本機端. Admin 首先需登入到 PNM 所在的 CentOS server, 然後準備好要新增、修改、刪除的資料庫, 然後將其資訊發給 PNM, PNM 會依據所提供的資料庫及命令, 對 whiteList.db 做出相對應的動作. 為了不影響 PNM 對其他 GW 和 App 的反應, 建議將要動作的資料量分批, 例如以 1000筆為單位進行處理.

9. Sample code

任何跟 PNM 溝通的訊息/命令, 皆可在 PNM source code 同一個目錄下的 app.c 裡找到範例, 包含了 App 所用到的 subscribe/unsubscribe/update, GW 所用到的 publish/update, 以及 Admin 批次管理的相關 code. 可以透過 make app 去編譯 app.c, 然後輸入 ./app 去執行範例.