PPHOOK

Carrying all M2M, H2M and H2H communication services

PPUser Porting Guide Android

Version 1.0.3

Authors:

Tobby Lai

Xue-Yi Chen

Technical Support

tobby.lai@morelinktek.com.tw

giles.chen@morelinktek.com.tw

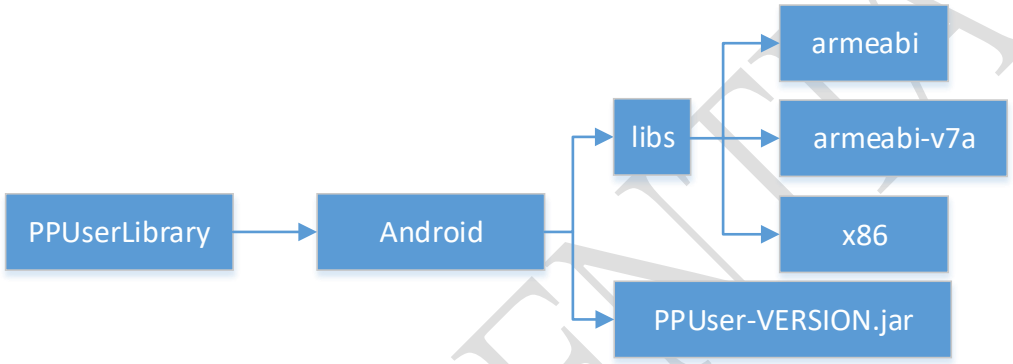
Date	Releases	Features
2017/05/25	V1.0.0	Initial release
2017/06/09	V1.0.1	Proofreading
2017/06/16	V1.0.2	Modify some graphs and add new APIs
2017/06/22	V1.0.3	Modify some description and typography

內容

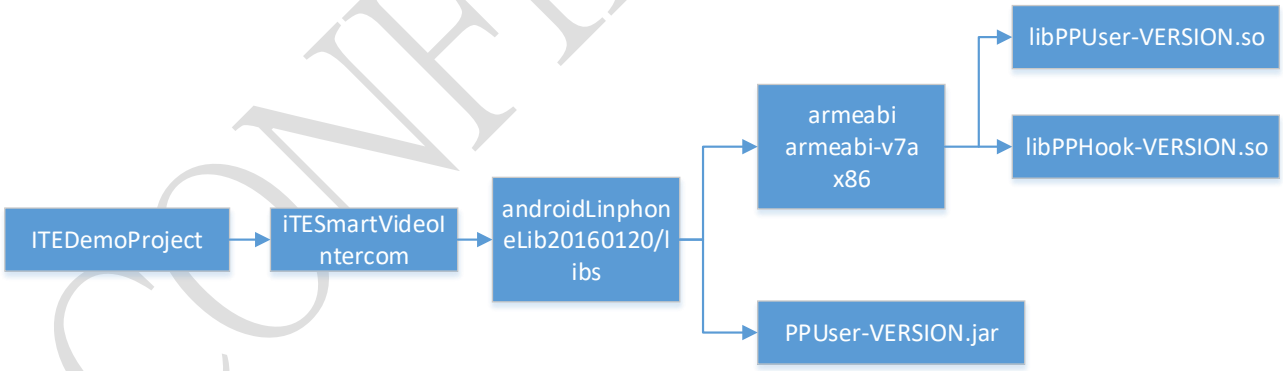
1.資料夾說明

SDK 資料夾內有二個資料夾 PPUserLibrary、ITEDemoProject：

- PPUserLibrary：Android 使用的 PPUser-VERSION.jar、libpphook-VERSION.so、libPPUser-VERSION.so。



- ITEDemoProject：Android 使用 PPUser 函式庫的 ITE Demo 專案。



2.Android PPUser 使用說明

2.1. 開發前準備

2.1.1. 運行環境

Android 2.3（API Level 9）及以上版本運行。

2.1.2. 獲取授權金鑰與產品編號

請連絡展連科技獲取**授權金鑰**與**產品編號**。PPUser 函式庫需要授權成功之後，才能夠正確運行。因此每台手機在使用 PPUser 函式庫的時候，都需要先使用**授權金鑰**與**產品編號**取得線上授權。

2.2. 使用 SDK 開發

2.2.1. 導入 SDK 至專案

1. 將 PPUser-VERSION.jar 複製到專案的 libs 資料夾中，若無 libs 資料夾請自行創建。將 libs 下各目錄複製到專案 libs 目錄中。如果使用 Android Studio 開發，則需於 src/main 目錄中創建 jniLibs 資料夾，並將 libs 下各目錄複製到 jniLibs 資料夾中。若專案使用其它.so 文件，則僅需複製 libs 文件下對應的目錄。
2. 將上述 jar 加入專案中的 **Java Build Path**。
3. AndroidManifest.xml 加入 **Permission**

```
1. <!-- 網路存取 --!>
2. <uses-permission android:name="android.permission.INTERNET"/>
3. <!-- 偵測網路狀態 --!>
4. <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

4. AndroidMainfest.xml 加入 PPUserService 配置

```
<service android:name="com.pphook.libppuser.PPUserService"/>
```

2.2.2. API 呼叫

以下介紹如何呼叫 PPUser 函式庫中的 API 並完成操作，介紹中將展示部分 ITE Demo Project 使用之範例，完整流程可直接參考 ITE Demo Project。

2.2.2.1. 授權

每台手機第一次執行 APP，使用 API 的時候，都需要呼叫函數 requestAuth，其中參數 authKey 是授權金鑰，參數 prodNum 是產品編號。使用 addAuthListener 新增監聽函式，取得授權的結果，如果授權失敗，PPUser 函式庫將無法正常運行。

```
PPUserManager.getInstance().requestAuth(context, authKey, prodNum)
```

★ 註：此 API 僅需於 APP 第一次執行時進行呼叫，重複呼叫可能會影響執行效率。因此在 APP 運行的時候，可以使用 isAuthed()確認是否已經授權成功。

```
boolean authed = PPUserManager.getInstance().isAuthed()
```

ITE Demo Project 中 androidSmartVideoIntercomApp 以 src/main/java/com/ite/smartvideointercom 中的 LauncherAuthActivity.java 作為應用啟動時第一個呈現的頁面，需於此處輸入授權金鑰與產品編號，方能成功執行應用，此 Demo Project 為範例專案，因此每次啟動應用時皆會顯示此頁面，實際開發時僅需呼叫一次或是取得**授權金鑰**與**產品編號**後，使用背景呼叫的方式完成認證即可，以下為 ITE Demo Project 範例：

顯示 EditText 讓使用者輸入**授權金鑰**與**產品編號**，按下確認鍵之後呼叫 requestAuth()要求授權。使用函式 onAuthStateUpdated()監聽取得授權結果，授權成功切換至 LauncherActivity。

```
1.  public class LauncherAuthActivity extends AppCompatActivity implements View.OnClickLi
    stener, PPUserAuthListener{
2.      EditText et_authKey;
3.      EditText et_prodNum;
4.      Button btn_confirm;
5.      ProgressDialog mProgressDialog;
6.
7.      @Override
8.      protected void onCreate(@Nullable Bundle savedInstanceState) {
9.          super.onCreate(savedInstanceState);
10.         setContentView(R.layout.activity_auth_launcher);
11.
12.         et_authKey = (EditText) findViewById(R.id.et_authKey);
13.         et_prodNum = (EditText) findViewById(R.id.et_prodNum);
14.         btn_confirm = (Button) findViewById(R.id.btn_confirm);
15.         btn_confirm.setOnClickListener(this);
16.         // 監聽授權結果
17.         PPUserManager.getInstance().addAuthListener(this);
18.
19.         // 每一次顯示頁面時，自動於欄位中填入授權成功的授權金鑰與產品編號
20.         et_authKey.setText(PPUserManager.getInstance().getAuthKey(this));
21.         et_prodNum.setText(PPUserManager.getInstance().getProdNum(this));
22.     }
23.
24.     @Override
25.     protected void onDestroy() {
26.         super.onDestroy();
27.         // Activity 結束時移除監聽
28.         PPUserManager.getInstance().removeListener(this);
29.     }
30.
31.     @Override
32.     public void onClick(View v) {
```

```
33.         mProgressDialog = ProgressDialog.show(this, "progressing", "Please wait...",
    true);
34.         // 請求授權
35.         PPUserManager.getInstance().requestAuth(this, et_authKey.getText().toString()
    , et_prodNum.getText().toString());
36.     }
37.
38.     /**
39.      * 監聽授權結果
40.      * @param stat
41.      *         PPUserStat.AUTH_SUCC 授權成功
42.      *         PPUserStat.AUTH_FAIL 授權失敗
43.      *         PPUserStat.AUTH_NETWORK_ERROR 網路連接失敗
44.      */
45.     @Override
46.     public void onAuthStateUpdated(int stat) {
47.         /* 使用者自定義處理與顯示 */
48.
49.         if(stat == PPUserStat.AUTH_SUCC){ // 授權成功
50.             mProgressDialog.dismiss();
51.             // 跳轉至應用啟動流程
52.             Intent newIntent = new Intent(this, LauncherActivity.class);
53.             startActivity(newIntent);
54.             finish();
55.         } else if(stat == PPUserStat.AUTH_FAIL){ // 授權失敗
56.             mProgressDialog.dismiss();
57.             Toast.makeText(this, "認證失敗，請確認授權金鑰與產品編號
    ", Toast.LENGTH_SHORT).show();
58.         } else if(stat == PPUserStat.AUTH_NETWORK_ERROR) { // 網路連接失敗
59.             mProgressDialog.dismiss();
60.             Toast.makeText(this, "認證失敗，請確認網路狀態
    ", Toast.LENGTH_SHORT).show();
61.         }
62.     }
63. }
```

2.2.2.2. 儲存使用者資訊

為了能夠與 Windows PPHookService 建立連線以及進行影音通話，APP 需要呼叫 saveUserInfo()儲存使用者資訊。其中 groupId 用來分辨此使用者所屬的群組，myAccount 為使用者的 SIP 帳號名稱，pphookServiceToken 為 Windows PPHookService 的 Token。

```
PPUserManager.getInstance().saveUserInfo(context,groupId,myAccount,pphookServiceToken)
```

★ 註：此 API 將使用者資訊儲存於 PPUser Manager 中，不需於每次啟動 PPUser 的時候重複呼叫。若 APP 變更使用者資訊，可於任何時間呼叫此 API 更新使用者資訊。

ITE Demo Project 中將掃描 ICM Server 的 QR Code 資訊儲存於 APP 內，位於 androidSmartVideoIntercomApp\src\main\java\com\ite\smartvideointercom\fragments 的 SettingsFragment.java，以下為範例只截取部分內容：

在 doActivityResult ()中分析 QR Code 的內容，若參數正確則呼叫 saveUserInfo()儲存使用者資訊

```
1.  public class SettingsFragment extends PreferencesListFragment {
2.      // ...
3.
4.      // 掃描完 QRcode 後，呼叫的方法
5.      private void doActivityResult(int requestCode, int resultCode, Intent data) {
6.          if ((resultCode == android.support.v7.app.AppCompatActivity.RESULT_OK)) {
7.              switch (requestCode) {
8.                  default:
9.                      break;
10.                 case 129:
11.                     final String[] info = data.getStringExtra("SCAN_RESULT").replaceA
    ll("(^[\\s\\r\\n]+)|([\\s\\r\\n]+$)|([\\t ]+)", "").split("\\r?\\n");
12.
```

```
                    // ...

                    } else if(info.length == 5){
                        doSipSignin(info[0], info[1], info[2]);
                        // 將 groupId、使用者帳號、pphookServiceToken 儲存於應用中
                        PPUserManager.getInstance().saveUserInfo(getActivity(), info[4], info[0], info[3]);
                        // 與 PPHookService 建立連線
                        PPUserManager.getInstance().buildConnWithPPHookService(getActivity());
                    }
                    break;
            }
        }
    }
}
```

2.2.2.3. 初始化及運行 PPUser

當正確取得授權後，需呼叫 init()，進行 PPUser 初始化並且開始運行。

```
PPUserManager.getInstance().init(context)
```

★ 註：初始化需要網路連接，請確保網路暢通。

使用 addListener 加入 PPUserStatListener 函式監聽 PPUser 的運行狀態。

```
PPUserManager.getInstance().addListener(new PPUserStatListener() {
    @Override
    public void onStateUpdated(int stat) {
            /* 自定義狀態處理與顯示*/
    }
});
```

ITE Demo Project 中 androidSmartVideoIntercomApp\src\main\java\com\ite\smartvideointercom 的 LauncherActivity.java 啟動 linphone 時，進行 PPUser 初始化並且開始運行；於 androidSmartVideoIntercom App/src/main/java/com/ite/smartvideointercom/fragments 的 MainFragment.java 監聽 PPUser 的運行狀態，並且修改頁面上顯示的狀態，以下為範例：

在 LauncherActivity 中進行 PPUser 初始化並且開始運行

```
1.  // LauncherActivity.java
2.  public class LauncherActivity extends AppCompatActivity {
3.
4.      @Override
5.      protected void onCreate(Bundle savedInstanceState) {
6.          super.onCreate(savedInstanceState);
7.
8.          // ...
9.
10.         // 初始化 PPUserManager
11.         PPUserManager.getInstance().init(this);
12.         startActivity(new Intent(this, LinphoneLauncherActivity.class));
13.     }
14. }
```

以 onStateUpdated()作為 PPUserStatListener 監聽 PPUser 的運行狀態，根據狀態修改頁面上顯示的狀態

```
1.  public class MainFragment extends android.app.Fragment implements View.OnClickListene
    r, View.OnLongClickListener, View.OnTouchListener, PPUserStatListener {
2.      ImageView status_led;  // 狀態燈號
3.      TextView status_text;  // 狀態文字
4.      LinphonePreferences mPrefs; // 取得 linphone 帳號資訊
5.
6.      // ...
7.
```

```
8.      @Override
9.      public void onCreate(Bundle savedInstanceState) {
10.         super.onCreate(savedInstanceState);
11.         Log.e("MainFragment::onCreate");
12.         // 新增監聽
13.         PPUserManager.getInstance().addListener(this);
14.         mPrefs = LinphonePreferences.instance();
15.     }
16.
17.     // ...
18.
19.     @Override
20.     public void onResume() {
21.         super.onResume();
22.         Log.e("MainFragment::onResume");
23.         // 每一次恢復頁面時，更新狀態
24.         onStateUpdated(PPUserManager.getInstance().getPPUserStat());
25.     }
26.
27.     // ...
28.
29.     @Override
30.     public void onDestroy() {
31.         super.onDestroy();
32.         Log.e("MainFragment::onDestroy");
33.         // 移除監聽
34.         PPUserManager.getInstance().removeListener(this);
35.     }
36.     /**
37.      * 監聽授權結果
38.      * @param stat
39.      *         PPUserStat.COMMUNICATING 通話中
40.      *         PPUserStat.CONNECTED 與 PPHookService 成功建立連線
41.      *         PPUserStat.USING_EXPIRED_AUTH 憑證過期
42.      *         PPUserStat.ONLINE 初始化成功，已加入 PPHook
43.      *         PPUserStat.JOINING 初始化中，正在加入 PPHook
44.      *         PPUserStat.UNINIT 未初始化
45.      */
```

```
46.     @Override
47.     public void onStateUpdated(int stat) {
48.         /* 使用者自定義狀態處理與顯示 */
49.         // 修改燈號與文字顯示
50.         switch (stat) {
51.             case PPUserStat.COMMUNICATING:
52.             case PPUserStat.CONNECTED:
53.                 status_led.setImageResource(R.drawable.led_connected);
54.                 status_text.setText(R.string.Connected);
55.                 break;
56.             case PPUserStat.CERT_EXPIRED:
57.                 status_led.setImageResource(R.drawable.led_error);
58.                 status_text.setText(R.string.Expired);
59.                 break;
60.             case PPUserStat.ONLINE:
61.             case PPUserStat.JOINING:
62.                 if(mPrefs.getAccountCount() > 0) {
63.                     status_led.setImageResource(R.drawable.led_inprogress);
64.                     status_text.setText(R.string.connecting);
65.                 } else {
66.                     // 尚未有任何帳號資訊
67.                     status_led.setImageResource(R.drawable.led_inprogress);
68.                     status_text.setText(R.string.NotAccount);
69.                 }
70.             case PPUserStat.UNINIT:
71.                 status_led.setImageResource(R.drawable.led_disconnected);
72.                 status_text.setText(R.string.Joining);
73.                 break;
74.         }
75.     }
76. }
```

2.2.2.4. 處理推播訊息

當對講機播號給使用者，使用者的 APP 尚未連接上 Windows PPHookService，Windows PPHookService 會發送推播訊息給使用者。APP 收到推播訊息之後，需執行相對應的動作。

ITE Demo Project 中於 androidSmartVideoIntercomApp\src\main\java\com\ite\smartvideointercom\baidu 的 PushMessageReceiver.java 百度推播的 API 內處理推播訊息。以下為範例：

為了讓使用者手機省電，未開啟 APP 的時候，不會運行 PPUser。因此當收到推播時，使用 isInstanciated 檢查 PPUser 是否正在運行，若尚未運行，則呼叫 init()進行初始化與運行 PPUser；使用 isPPHookServiceConnected 檢查是否已經與 Windows PPHookService 建立連線，若尚未連線，則呼叫 buildConnWithPPHookService 進行連線

```
1.  public class PushMessageReceiver extends com.baidu.push.lib.BaiduPushMessageReceiver
     {
2.      // ...
3.      @Override
4.      public void onMessage(final Context context, String message, String customConten
    tString) {
5.          Log.d(TAG, "+++onMessage@2");
6.          super.onMessage(context, message, customContentString);
7.          // 收到推播，若尚未初始化則開始初始化
8.          if(!PPUserManager.getInstance().isInstanciated())
9.              PPUserManager.getInstance().init(context);
10.         // 收到推播，若沒有連上 PPHookService 則連回 PPHookService
11.         if(!PPUserManager.getInstance().isPPHookServiceConnected())
12.             PPUserManager.getInstance().buildConnWithPPHookService(context);
13.           // ...
14.       }
15.           // ...
16.  }
```

2.2.2.5. 監聽來電

為了能夠監聽來電，需與 Windows PPHookService 建立連線。

```
PPUserManager.getInstance().buildConnWithPPHookService(context)
```

★ 註：只有監聽來電才需與 Windows PPHookService 保持連線，使者主動撥號則不需事先建立連線。

ITE Demo Project 中使用 buildConnWithPPHookService()的範例，如 2.2.2.2 與 2.2.2.4 所示。

2.2.2.6. 網路切換事件

當手機網路發生改變時，如 Wi-Fi、3G、4G 切換，PPUser 將自動偵測網路環境，但反應較慢。為了能快速應變，當 APP 偵測到網路產生變化，主動呼叫 connectivityChanged ()，將更快速地應對網路變化。

```
PPUserManager.getInstance().connectivityChanged(context, connectivityManager)
```

ITE Demo Project 中 androidLinphoneLib20160120\src\org\linphone 的 NetworkManager.java 為 linphone 監聽網路變化的 Receiver，以下為範例：

在 NetworkManager 偵測網路切換事件的函式 onReceive ()中，根據網路切換的狀況呼叫 connectivityChanged ()

```
1. public class NetworkManager extends BroadcastReceiver {
2. 
3.     @Override
4.     public void onReceive(Context context, Intent intent) {
5.         ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Conte
   xt.CONNECTIVITY_SERVICE);
6.         Boolean lNoConnectivity = intent.getBooleanExtra(ConnectivityManager.EXTRA_NO
   _CONNECTIVITY,false);
7.         if (LinphoneManager.isInstanciated()) {
8.             LinphoneManager.getInstance().connectivityChanged(cm, lNoConnectivity);
9.         }
10.        // PPUserManager 進行網路變化處理
11.        if(PPUserManager.getInstance().isInstanciated()) {
12.            PPUserManager.getInstance().connectivityChanged(context, cm);
13.        }
14.    }
15. 
16.}
```

2.3. 狀態碼說明

狀態	狀態編號	描述
UNINIT	1000	尚未初始化
JOINING	1001	初始化中，需保持網路暢通
ONLINE	1002	初始化完成，需保持網路暢通
CONNECTED	1003	已連接 Windows PPHookService
COMMUNICATING	1004	SIP 通話中
UDP_SIP_SOCK_FAIL	1005	創建 UDP SIP socket 失敗
UDP_AUDIO_RTP_SOCK_FAIL	1006	創建 Audio RTP socket 失敗
UDP_AUDIO_RTCP_SOCK_FAIL	1007	創建 Audio RTCP socket 失敗
UDP_VIDEO_RTP_SOCK_FAIL	1008	創建 Video RTP socket 失敗
UDP_VIDEO_RTCP_SOCK_FAIL	1009	創建 Video RTCP socket 失敗
TCP_SIP_SOCK_FAIL	1010	創建 TCP SIP socket 失敗
HTTP_SERVER_SOCK_FAIL	1011	創建 HTTP server socket 失敗
NOT_FIND_AUTH	1012	尚未獲取授權
ERROR_FORMAT_AUTH	1013	授權錯誤
USING_ EXPIRED _AUTH	1014	目前授權過期
AUTH_SUCC	2000	成功獲取授權
AUTH_FAIL	2001	獲取授權失敗
AUTH_NETWORK_ERROR	2002	網路無法連接失敗，請檢查網路

2.4. API 說明

以下說明 PPUser library 的 API。

2.4.1. 類別

類別	描述
PPUserManager	提供所有操作 PPUserManager 的方法
PPUserStat	定義 PPUserManager 的狀態碼
PPUserStatListener	監聽 PPUserManager 狀態的 Listener
PPUserAuthListener	監聽.授權狀態的 Listener

2.4.2. API

分類	功能	API 方法原型
授權服務	請求授權、取得成功的**授權金鑰**與**產品編號**	requestAuth, getAuthKey, getProdNum, addAuthListener, removeAuthListener, onAuthStateUpdated
PPUserManager 服務	初始化、退出、建立與關閉連線	init, destroy, buildConnWithPPHookService, closeConnWithPPHookService
使用者資訊管理	儲存、清除使用者資訊	saveUserInfo, clearUserInfo
網路管理	網路變動處理	connectivityChanged
狀態管理	狀態變化	addListener, removeListener, onStateUpdated, getPPUserStat, isInstanciated, isPPHookJoined, isPPHookServiceConnected, isCommunicated

2.4.3. 授權服務

1. requestAuth

原型	**public void** requestAuth(Context context,String authKey, String prodNum)
描述	請求 PPUser 使用授權時，需要進行網路操作。此動作僅需於應用第一次執行時呼叫，成功授權後將自動儲存授權金鑰與產品編號於 PPUser 中，無須開發者自行儲存。
參數	context：目前執行的 context authKey：授權金鑰 prodNum：產品編號
回傳	將透過監聽的 Auth Listener 返回結果，狀態碼詳見 2.3

2. getAuthKey

原型	**public** String getAuthKey(Context context)
描述	取得成功授權後的授權金鑰。
參數	context：目前執行的 context
回傳	回傳最後一次輸入成功授權的金鑰

3. getProdNum

原型	**public** String getProdNum(Context context)
描述	取得成功授權後的產品編號。
參數	context：目前執行的 context
回傳	回傳最後一次輸入成功授權的產品編號

4. isAuthed

原型	**public boolean** isAuthed()
描述	是否已取得授權。
參數	無
回傳	boolean：已取得授權返回 true，反之為 false

5. addAuthListener

原型	**public void** addAuthListener(PPUserAuthListener listener)
描述	新增監聽的 Auth Listener，可於多個不同的類別中監聽請求結果，當獲取結果時，所有加入的 Auth Listener 皆會被觸發。
參數	listener：監聽事件的 listener
回傳	無

6. removeAuthListener

原型	**public void** removeAuthListener(PPUserAuthListener listener)
描述	移除監聽的 Auth Listener，不再使用到的 Auth Listener 必須進行釋放，被移除的 Auth Listener 將不再被觸發。
參數	listener：監聽事件的 listener
回傳	無

7. onAuthStateUpdated

原型	**void** onAuthStateUpdated(**int** stat)
描述	PPUserAuthListener 類別的抽象方法，當請求授權時，監聽請求結果。
參數	stat：授權結果，狀態碼詳見 2.3
回傳	無

2.4.4. PPUser 服務

1. init

原型	**public void** init(Context context)
描述	初始化 PPUser，並開始運行，必須確保網路暢通。
參數	context：目前執行的 context
回傳	將透過監聽的 Listener 返回結果，狀態碼詳見 2.3

2. destroy

原型	**public void** destroy(Context context)
描述	停止 PPUser 所有操作，釋放記憶體。
參數	context：目前執行的 context
回傳	無

3. buildConnWithPPHookService

原型	**public void** buildConnWithPPHookService(Context context)
描述	建立與 Windows PPHookService 的連線，以監聽來電，若使用者主動撥號，則不需事先呼叫此 API，PPUser 將會自動偵測撥號，並進行連線建立，若事先建立好連線將加快撥號響鈴時間。 註：需要先呼叫 saveUserInfo，儲存使用者資訊 groupId、myAccount 以及 pphookServiceToken 才能成功建立連線，使用者資訊由掃描 ICM Server 產生之 QR Code 取得。
參數	context：目前執行的 context
回傳	將透過監聽的 Listener 返回結果，狀態碼詳見 2.3

4. closeConnWithPPHookService

原型	**public void** closeConnWithPPHookService(Context context)
描述	關閉與 Windows PPHookService 的連線，若未建立連線，則不會運行任何操作。
參數	context：目前執行的 context
回傳	將透過監聽的 Listener 返回結果，狀態碼詳見 2.3

2.4.5. 使用者資訊管理

5. saveUserInfo

原型	**public void** saveUserInfo(Context context, String groupId, String myAccount, String pphookServiceToken)
描述	儲存使用者資訊於應用中，僅需呼叫一次，重複呼叫將覆蓋上一次儲存的資訊。
參數	context：目前執行的 context groupId：使用者所屬群組 myAccout：使用者的 SIP 帳號 pphookServiceToken：目標連線的 Windows PPHookService Token
回傳	無

6. clearUserInfo

原型	**public void** clearUserInfo(Context context)
描述	清除 PPUser 中儲存的使用者資訊，若 PPUser 的狀態為連線中，將關閉與 Windows PPHookService 的連線。
參數	context：目前執行的 context
回傳	無

2.4.6. 網路管理

1. connectivityChanged

原型	**public void** connectivityChanged(Context context, ConnectivityManager cm)
描述	當網路發生改變時，如切換 Wi-Fi、3G、4G 網路時，呼叫此 API，將快速自動重啟網路相關服務。若發生網路變更時，沒有呼叫此 API，將由 PPUser 自行偵測，反應速度較慢。 若要使用此 API，需於 AndroidManifest.xml 中宣告以下 Permission： **<uses-permission** android:name="android.permission.ACCESS_NETWORK_STATE"**/>**
參數	context：目前執行的 context cm：系統中管理網路連線的類別
回傳	無

2.4.7. 狀態管理

1. addListener

原型	**public void** addListener(PPUserStatListener listener)
描述	新增監聽的 Listener，可於多個不同的類別中進行監聽，當發生狀態改變時，所有加入的 Listener 皆會被觸發。
參數	listener：監聽事件的 listener
回傳	無

2. removeListener

原型	**public void** removeListener(PPUserStatListener listener)
描述	移除監聽的 Listener，為了避免記憶體洩漏，不再使用到的 listener 必須進行釋放，被移除的 Listener 將不再被觸發。
參數	listener：監聽事件的 listener
回傳	無

3. onStateUpdated

原型	**void** onStateUpdated(**int** stat)
描述	PPUserStatListener 類別的抽象方法，當 PPUser 狀態改變時，將被觸發。
參數	stat：目前 PPUser 的狀態，狀態碼詳見 2.3
回傳	無

4. getPPUserStat

原型	**public int** getPPUserStat()
描述	主動取得 PPUser 目前運行的狀態。
參數	無
回傳	int：目前 PPUser 運行的狀態，狀態碼詳見 2.3

5. isInstanciated

原型	**public boolean** isInstanciated()
描述	PPUser 是否初始化成功。
參數	無
回傳	Boolean：成功初始化返回 true，反之為 false

6. isPPHookJoined

原型	**public boolean** isPPHookJoined()
描述	目前 PPUser 是否成功加入 PPHook 網路。
參數	無
回傳	Boolean：成功加入返回 true，反之為 false

7. isPPHookServiceConnected

原型	**public boolean** isPPHookServiceConnected()
描述	目前 PPUser 是否已經與 Windows PPHookService 成功建立連線。
參數	無
回傳	Boolean：已經成功建立連線返回 true，反之為 false

8. isCommunicated

原型	**public boolean** isCommunicated()
描述	目前 PPUser 是否正在通話中。
參數	無
回傳	Boolean：正在通話中返回 true，反之為 false