PPHOOK

Carrying all M2M, H2M and H2H communication services

PPUser Porting Guide iOS

Version 1.0.0

Authors:

Tobby Lai

Technical Support

giles.chen@morelinktek.com.tw

tobby.lai@morelinktek.com.tw

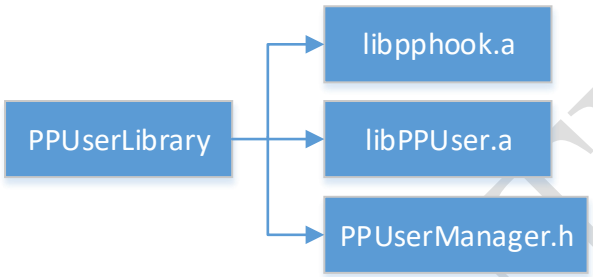
Date	Releases	Features
2017/06/20	V1.0.0	Initial release

內容

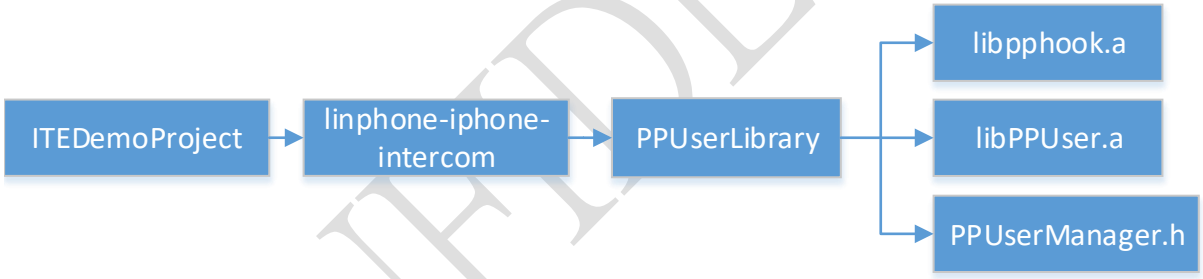
1.資料夾說明

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

- PPUserLibrary：iOS 使用的 libpphook.a、libPPUser.a 及 PPHookManager.h。



- ITEDemoProject：iOS 使用 PPUser 函式庫的 ITE Demo Project。



2.iOS PPUser 使用說明

2.1. 開發前準備

2.1.1. 運行環境

iOS6 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 至專案中

1. 將 PPUserManager 資料夾(libpphook.a、libPPUser.a 及 PPUserManager.h)加入專案中
2. 在設置中 Header Search Paths 加入 $(PROJECT_DIR)/PPUserLibrary
3. 在設置中 Library Search Path 加入 $(PROJECT_DIR)/PPUserLibrary
4. 專案 Build Phases 項目 Link Binary With Libraries 中點選「+」符號，加入額外動態函式庫 libz.tbd、libc++.tbd、libbz2.1.0.tbd、libstdc++.tbd、libresolv.tbd
5. 在使用 PPUser 函式庫 API 的檔案中加入 #import "PPUserManager.h"

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 PPUserRequestAuth()取得授權，其中參數 authorzationKey 是授權金鑰，參數 productNumber 是產品編號。使用 NSNotificationCenter 監聽 kPPUserAuthStat 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 instance] PPUserRequestAuth:authorizationKey productNumber:productNumber
];
```

★ 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 PPUserIsAuthed()確認是否已經授權成功。

```
BOOL authed = [[PPUserManager instance] PPUserIsAuthed];
```

ITE Demo Project 中，以 RegisterViewController.m 作為應用啟動時第一個呈現的頁面，此頁面的功能為取得 PPUser 線上授權以及掃描 QR Code，第一次啟動 APP 需於此頁面輸入**授權金鑰**與**產品編號**，授權成功之後，會進入掃描 QR Code 的畫面。以下為 ITE Demo Project 範例：

載入 RegisterViewController，判斷是否已經授權成功。若尚未授權，呼叫 getPPUserAuthCode()進行授權；若已經授權成功，則直接進入掃描 QR Code 的畫面。

```
1. // RegisterViewController.m
2. - (void)viewDidAppear:(BOOL)animated {
3.     [super viewDidAppear:animated];
4.     // PPUser
5.     if([[PPUserManager instance] PPUserIsAuthed] == FALSE) {
6.         [self getPPUserAuthCode];
7.     } else
8.     // PPUser --
9.     if (initialized && !initializedReading) {
10.            initializedReading = YES;
11.            if (IPAD) {
12.                [self showWaitAlertView];
13.                [self performSelector:@selector(startReadingForLoad) withObject:nil af
   terDelay:1.5];
14.            } else {
15.                [self startReading];
16.                [self layoutAllControllers];
17.            }
18.        }
19.    }
```

使用 NSNotificationCenter 監聽 kPPUserAuthStat，在 ppuserAuthStat()中取得授權結果

```
1. // RegisterViewController.m
2. - (void)viewWillAppear:(BOOL)animated {
3.     [super viewWillAppear:animated];
4.     [NSNotificationCenter.defaultCenter addObserver:self
5.                                             selector:@selector(registrationUpdateEvent:
   )
6.                                                 name:kLinphoneRegistrationUpdate
7.                                               object:nil];
8.     [NSNotificationCenter.defaultCenter addObserver:self
9.                                             selector:@selector(configuringUpdate:)
10.                                                name:kLinphoneConfiguringStateUpdat
   e
11.                                              object:nil];
```

```
12.
13.         // PPUser
14.         // add observer to get the status of PPUser authorization
15.         [NSNotificationCenter.defaultCenter addObserver:self
16.                                                selector:@selector(ppuserAuthStat:)
17.                                                    name:kPPUserAuthStat
18.                                                  object:nil];
19.         // PPUser --
20.
21.         new_config = NULL;
22.     /*==
23.         [self changeView:_welcomeView back:FALSE animation:FALSE];
24.     */
25.         number_of_configs_before = ms_list_size(linphone_core_get_proxy_config_list(LC
    ));
26.     }
27.
28.     - (void)ppuserAuthStat:(NSNotification *)notif {
29.         NSInteger authStat = [[notif.userInfo objectForKey:@"authStat"] integerValue];

30.         NSLog(@"[ppuserAuthStat] authStat:%ld", (long)authStat);
31.         if(authStat == PPUSER_AUTH_STAT_SUCC) {
32.             UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Authorization" m
    essage:@"PPUser APP is authorized successfully" delegate:nil cancelButtonTitle:@"OK" o
    therButtonTitles:nil];
33.
34.             [alert show];
35.             // check again and start PPUser
36.             if([[PPUserManager instance] PPUserIsAuthed] == TRUE) {
37.                 NSInteger ret = [[PPUserManager instance] PPUserInit];
38.                 if (ret < 0) {
39.                     NSLog(@"[ppuserAuthStat] PPUserStart failed:%ld", (long)ret);
40.                 }
41.             }
42.
43.             // start to scan QR code
44.             if (initialized && !initializedReading) {
45.                 initializedReading = YES;
```

```
46.                 if (IPAD) {
47.                     [self showWaitAlertView];
48.                     [self performSelector:@selector(startReadingForLoad) withObject:ni
   l afterDelay:1.5];
49.                 } else {
50.                     [self startReading];
51.                     [self layoutAllControllers];
52.                 }
53.             }
54.         } else {
55.             UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Authorization" m
   essage:@"Authorization Failed. Please enter correct authorization key and product numb
   er." delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
56.
57.             [alert show];
58.             // pop up input textfields again
59.             [self getPPUserAuthCode];
60.         }
61.     }
```

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 PPUserSaveUserInfo()儲存使用者資訊。其中 winPPHookServiceToken 為 Windows PPHookService 的 Token， groupId 用來分辨此使用者所屬的群組，sipAccount 為使用者的 SIP 帳號名稱。

```
[PPUserManager instance] PPUserSaveUserInfo:winPPHookServiceToken sipAccount:accountUser
Name groupId:groupId];
```

★ 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 中掃描 ICM Server 產生的 QR Code 資訊儲存於 APP 內，位於

RegisterViewController.m，以下範例只截取部分內容：

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

```
// RegisterViewController.m
- (void)pickupWithCode:(NSString*)code
{
    NSLog(@"QRCode:%@", code);

    BOOL status = NO;

    if (![Global isEmptyString:code]) {
        NSArray *array = [code componentsSeparatedByString:@"\n"];

            if (array != nil && [array count] > 2) {
                accountUserName = [array[0] stringByReplacingOccurrencesOfString:@"\r" withString:@""];
                accountPassword = [array[1] stringByReplacingOccurrencesOfString:@"\r" withString:@""];
                accountDomain = [array[2] stringByReplacingOccurrencesOfString:@"\r" withString:@""];

                // PPUser
                // get windoes PPHookService token and Group ID, and save it by calling PPUserSaveUserInfo()
                NSString *winPPHookServiceToken = @"";
                NSString *groupId = @"";
                if([array count] > 4) {
                    winPPHookServiceToken = [array[3] stringByReplacingOccurrencesOfString:@"\r" withString:@""];
                    groupId = [array[4] stringByReplacingOccurrencesOfString:@"\r" withString:@""];
                }
                [[PPUserManager instance] PPUserSaveUserInfo:winPPHookServiceToken sipAccount:accountUserName groupId:groupId];
                // PPUser --
```

```
26.
27.                 [UserConfig setWebServiceHost:accountDomain];
28.                 status = YES;
29.             }
30.         }
31.         //...
32.         //...
33.}
```

2.2.2.3. 初始化及運行 PPUser

當正確取得授權後，需呼叫 PPUserInit()，進行 PPUser 初始化並且開始運行。使用 NSNotificationCenter 監聽 kPPUserStat 取得 PPUser 的運行狀態。

```
NSInteger ret = [[PPUserManager instance] PPUserInit];
```

★ 註：此 API 需要網路連接，請確保網路暢通。

ITE Demo Project 中於 LinphoneAppDelegate.m，進行 PPUser 初始化並且開始運行；於 StatusBarView.m 監聽 PPUser 的運行狀態，並且更新狀態列的文字與圖示，以下為範例：

APP 開啟進入 didFinishLaunchingWithOptions()，判斷 PPUser 是否已經授權成功，若已經授權成功則進行 PPUser 初始化並且開始運行

```
1. // LinphoneAppDelegate.m
2. - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
3.
4.     // init UserConfig
5.     // init Linphone
6.     // init View
7.     // init APNS
8.     // ...
9.
```

```
10.       // PPUser
11.       // check the PPUser is authorized or not
12.       // if it is authorized then start it
13.       if([[PPUserManager instance] PPUserIsAuthed]) {
14.           NSInteger ret = [[PPUserManager instance] PPUserInit];
15.           if (ret < 0) {
16.               LOGI(@"PPUserInit failed:%ld", (long)ret);
17.           }
18.       }
19.       // PPUser --
20.
21.       return YES;
22.}
```

監聽 kPPUserStat，在 PPUserStatus()中取得 PPUser 的運行狀態，根據狀態更新狀態列的文字與圖示

```
1. // StatusBarView.m
2. - (void)viewWillAppear:(BOOL)animated {
3.     [super viewWillAppear:animated];
4.     // Set observer
5.     // PPUser
6.     // change from observing kLinphoneRegistrationUpdate to kPPUserStat
7.     //[NSNotificationCenter.defaultCenter addObserver:self
8.     //                                       selector:@selector(registrationUpdate:)
9.     //                                           name:kLinphoneRegistrationUpdate
10.   //                                         object:nil];
11.       [NSNotificationCenter.defaultCenter addObserver:self
12.                                              selector:@selector(PPUserStatus:)
13.                                                  name:kPPUserStat
14.                                                object:nil];
15.       // PPUser --
16.       // ...
17.       // observe kLinphoneGlobalStateUpdate
18.       // observe kLinphoneNotifyReceived
19.       // observe kLinphoneCallUpdate
20.       // observe kLinphoneCallEncryptionChanged
```

```
21.       // ...
22.   }
23.
24.   - (void)PPUserStatus:(NSNotification *)notif {
25.       PPUSER_STAT status = [[notif.userInfo objectForKey:@"status"] intValue];
26.       [self PPUserStatusUpdate:status];
27.   }
28.
29.   + (UIImage *)imageForPPUserStatus:(PPUSER_STAT)status {
30.       switch(status) {
31.           case PPUSER_STAT_UNINIT:
32.               return [UIImage imageNamed:@"led_error.png"];
33.           case PPUSER_STAT_JOINING:
34.               NSLog(@"[imageForPPHookStatus] PPUSER_STAT_JOINING");
35.               return [UIImage imageNamed:@"led_inprogress.png"];
36.           case PPUSER_STAT_JOINED:
37.               NSLog(@"[imageForPPHookStatus] PPUSER_STAT_JOINED");
38.               return [UIImage imageNamed:@"led_inprogress.png"];
39.           case PPUSER_STAT_EXPIRED:
40.               NSLog(@"[imageForPPHookStatus] PPUSER_STAT_EXPIRED");
41.               return [UIImage imageNamed:@"led_error.png"];
42.           case PPUSER_STAT_CONNECTED:
43.               NSLog(@"[imageForPPHookStatus] PPUSER_STAT_CONNECTED");
44.               return [UIImage imageNamed:@"led_connected.png"];
45.           case PPUSER_STAT_COMMUNICATING:
46.               NSLog(@"[imageForPPHookStatus] PPUSER_STAT_COMMUNICATING");
47.               return [UIImage imageNamed:@"led_connected.png"];
48.       }
49.       return [UIImage imageNamed:@"led_error.png"];
50.   }
51.
52.   - (void)PPUserStatusUpdate:(PPUSER_STAT)status {
53.       NSString *message = nil;
54.       switch(status) {
55.           case PPUSER_STAT_UNINIT:
56.               if(linphone_core_is_network_reachable(LC)) {
57.                   message = NSLocalizedString(@"Joining", nil);
58.               } else {
```

```
59.                     message = NSLocalizedString(@"Network down", nil);
60.                 }
61.
62.                 break;
63.             case PPUSER_STAT_JOINING:
64.                 message = NSLocalizedString(@"Joining", nil);
65.                 break;
66.             case PPUSER_STAT_JOINED:
67.                 message = NSLocalizedString(@"Connecting", nil);
68.                 break;
69.             case PPUSER_STAT_EXPIRED:
70.                 message = NSLocalizedString(@"Expired", nil);
71.                 break;
72.
73.                 // Communicate Status
74.             case PPUSER_STAT_CONNECTED:
75.                 message = NSLocalizedString(@"Connected", nil);
76.                 break;
77.             case PPUSER_STAT_COMMUNICATING:
78.                 message = NSLocalizedString(@"Talking", nil);
79.                 break;
80.         }
81.
82.         dispatch_async(dispatch_get_global_queue(0, 0), ^{
83.             dispatch_async(dispatch_get_main_queue(), ^{
84.                 [_registrationState setTitle:message forState:UIControlStateNormal];
85.                 _registrationState.accessibilityValue = message;
86.                 [_registrationState setImage:[self.class imageForPPUserStatus:status]
    forState:UIControlStateNormal];
87.             });
88.         });
89.     }
```

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，需要呼叫

PPUserProcessRemoteNotification()執行相對應的動作。

```
[[PPUserManager instance] PPUserProcessRemoteNotification:userInfo];
```

ITE Demo Project 中於 LinphoneAppDelegate.m 內處理推播訊息，以下為範例：

在推播處理函式 didReceiveRemoteNotification()中呼叫 PPUserProcessRemoteNotification()讓 PPUser 處理推播訊息

```
1. // LinphoneAppDelegate.m
2. - (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDict
   ionary *)userInfo {
3.     LOGI(@"%@ : %@", NSStringFromSelector(_cmd), userInfo);
4.     // PPUser
5.     // handle the received APNS message
6.     [[PPUserManager instance] PPUserProcessRemoteNotification:userInfo];
7.     // PPUser --
8.     [self processRemoteNotification:userInfo];
9. }
```

2.2.2.5. 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 PPUserSetConnectivityChanged()，將更快速地應對網路變化。

```
[[PPUserManager instance] PPUserSetConnectivityChanged];
```

ITE Demo Project 在 LinphoneManager.m 中監聽網路切換事件，以下為範例：

偵測網路切換事件的函式 networkReachabilityCallBack()中，根據網路切換的狀況呼叫 PPUserSetConnectivityChanged()

```
1. // LinphoneManager.m
2. void networkReachabilityCallBack(SCNetworkReachabilityRef target, SCNetworkReachabili
   tyFlags flags, void *nilCtx) {
3.     showNetworkFlags(flags);
4.     LinphoneManager *lm = LinphoneManager.instance;
5.     SCNetworkReachabilityFlags networkDownFlags = kSCNetworkReachabilityFlagsConnecti
   onRequired |
6.                                                   kSCNetworkReachabilityFlagsConnecti
   onOnTraffic |
7.                                                   kSCNetworkReachabilityFlagsConnecti
   onOnDemand;
8.
9.     if (theLinphoneCore != nil) {
10.           LinphoneProxyConfig *proxy = linphone_core_get_default_proxy_config(theLin
   phoneCore);
11.
12.           struct NetworkReachabilityContext *ctx = nilCtx ? ((struct NetworkReachabi
   lityContext *)nilCtx) : 0;
13.           if ((flags == 0) || (flags & networkDownFlags)) {
14.               // ...
15.               // PPUser
16.               [[PPUserManager instance] PPUserSetConnectivityChanged];
17.               // PPUser --
18.           } else {
19.               // ...
20.               if (lm.connectivity != newConnectivity) {
21.                   // ...
22.                   LOGI(@"Network connectivity changed to type [%s]", (newConnectivit
   y == wifi ? "wifi" : "wwan"));
23.                   // PPUser
24.                   [[PPUserManager instance] PPUserSetConnectivityChanged];
25.                   // PPUser --
26.               }
27.               // ...
28.           }
29.           // ...
30.       }
31.}
```

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 函式庫的 API。

2.4.1. 類別

類別	描述
PPUserManager	提供所有操作 PPUser 的物件
PPUSER_AUTH_STAT	定義 PPUser 授權的狀態碼
PPUSER_INIT_ERROR	定義 PPUser 初始化的狀態碼
PPUSER_STAT	定義 PPUser 的狀態碼
kPPUserAuthStat	通知名稱，用於監聽 PPUser 授權的狀態碼
kPPUserStat	通知名稱，用於監聽 PPUser 的狀態碼

2.4.2. API 功能

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**	PPUserRequestAuth, PPUserIsAuthed
PPUser 服務	初始化、退出、建立與關閉連線	PPUserInit, PPUserDestroy, PPUserBuildConnWithPPHookService, PPUserCloseConnWithPPHookService
使用者資訊管理	儲存、清除使用者資訊	PPUserSaveUserInfo
網路管理	網路變動處理	PPUserSetConnectivityChanged
狀態管理	狀態變化	PPUserGetStat

2.4.3. 授權服務

1. PPUserRequestAuth

原型	-(void) PPUserRequestAuth:(NSString *)authorzationKey productNumber:(NSString *)productNumber;
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	authorzationKey：授權金鑰 productNumber：產品編號
回傳	透過監聽 kPPUserAuthStat 取得授權結果，授權狀態碼詳見 2.3

2. PPUserIsAuthed

原型	-(BOOL) PPUserIsAuthed;
描述	是否已取得授權。
參數	無
回傳	BOOL：已取得授權返回 true，反之為 false

2.4.4. PPUser 服務

1. PPUserInit

原型	-(NSInteger) PPUserInit;
描述	初始化 PPUser，並開始運行，必須確保網路暢通。
參數	無
回傳	NSInteger，狀態碼詳見 2.3

2. PPUserDestroy

原型	-(void) PPUserDestroy;
描述	停止 PPUser 所有操作，釋放記憶體。
參數	無
回傳	無

3. PPUserBuildConnWithPPHookService

原型	-(void) PPUserBuildConnWithPPHookService;
描述	建立與 Windows PPHookService 的連線，用於監聽來電，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，並建立連線，若事先建立好連線將加快撥號響鈴時間。 註：需要先呼叫 PPUserSaveUserInfo，儲存使用者資訊 groupId、sipAccount 以及 winPPHookServiceToken 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	無
回傳	無

4. PPUserCloseConnWithPPHookService

原型	-(void) PPUserCloseConnWithPPHookService;
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	無
回傳	無

2.4.5. 使用者資訊管理

1. PPUserSaveUserInfo

原型	-(void) PPUserSaveUserInfo:(NSString *)_winPPHookServiceToken sipAccount:(NSString *)_sipAccount groupId:(NSString *)_groupId;
描述	儲存使用者資訊於 PPUser 中，僅需呼叫一次，重複呼叫將覆蓋上一次儲存的資訊。
參數	winPPHookServiceToken： 目標連線的 Windows PPHookService Token sipAccount：使用者的 SIP 帳號 groupId：使用者所屬群組
回傳	無

2.4.6. 網路管理

1. PPUserSetConnectivityChanged

原型	-(void) PPUserSetConnectivityChanged;
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。
參數	無
回傳	無

2.4.7. 狀態管理

1. PPUserGetStat

原型	-(PPUSER_STAT) PPUserGetStat;
描述	主動取得 PPUser 目前運行的狀態。
參數	無
回傳	PPUSER_STAT：目前 PPUser 運行的狀態，狀態碼詳見 2.3