

I. 簡介

本 plugin 提供 phoneGap 開發者一個與 iOS/Android COIMOTION SDK 的介接。讓開發者使用 phoneGap/Cordova 開發 app 時可透過 plugin 的使用，來獲得原生開發者使用 COIMOTION SDK 的便利性。例如處理使用者登入/註冊/更新密碼及自動儲存/更新 token 等等。開發者亦可透過一般的 Web 開發方式使用直接使用 **COIMOTION API**。

0.9.6.3 之後新增新增 SWS 模組的支援，開發者可透過 facebook 或是 google 帳號進行登入與 graph API/Google+ API 的操作。

II. 安裝

開發者想要在自己的專案中使用 **COIMOTION** plugin 的話，可以透過以下步驟來完程 SDK 的使用設定：

A. 取得 COIMOTION SDK

1. 由網址下載 <http://tw.coimotion.com/wcoim/SDK/coimPlugin.zip>
2. COIMOTION 網站亦提供載點連結，開發者們可以在開發者後台的“捷徑”→“下載”進入工具下載專業，選擇下載 plugin 壓縮檔即可



3. 透過 gitHub 安裝
在專案目錄下執行：
“# cordova plugin add https://github.com/coimotion/coimPlugin.git” 或是
“# phonegap local plugin add https://github.com/coimotion/coimPlugin.git”

B. 在 phoneGap 中使用本 plugin

1. 將下載的壓縮檔解開，在專案目錄下執行：
“# cordova plugin add path/to/coimPlugin” 或是

“# phonegap local plugin add path/to/coimPlugin”

即可在專案中安裝好本 plugin。

C. 設定 app_key 與 app_code

1. 如果是 android 專案的話

請在.../ **[project-name]/platforms/android/AndroidManifest.xml** 中的<application>中加入兩個<meta-data>：

```
<meta-data android:name="coim_app_key" android:value="blah..." />
```

```
<meta-data android:name="coim_app_code" android:value="blah..." />
```

2. 如果是 ios 專案的話

請在.../ **[project-name]/platforms/ios/[project-name]/[project-name]-Info.plist** 中加 properties (xcode 下開啟 plist，右鍵“Add Row”)

”coim_app_key”

”coim_app_code”

D. coim_app_key 與 coim_app_code

值請登入 **COIMOTION** 網站，進入”APP 管理”選擇對應的 APP。切換至”基本資料”分頁。其中的代碼即為 coim_app_code，程式金鑰則是 coim_app_key 填入對應的值後後便可以在程式中使用 plugin 囉。

E. 輸出 debug 訊息

1. Android

同 coim_app_code 設定，在 AndroidManifest 中加入 **<meta-data android:name="coim_debug" android:value="true" />**

2. iOS

同 coim_app_code 設定，在 plist 中加入”coim_debug”，型別為布林值，設為 YES

如未設定則預設為 false/NO，開啟後可在 console.log 中看到輸出的除錯訊息(以[DEBUG] – 開頭)。目前會輸出 API 呼叫時的完整 URL 與完整參數以及回傳的原始資料(json 字串)

F. Timeout 設定

加入 timeout 機制，當 API 呼叫超過 30 秒沒有完成的話，plugin 會以 fail 的 callback 傳回 timeout 的錯誤訊息，如果開發者想要自訂 timeout 的時間的話：

1. Android

在 AndroidManifest.xml 下加入 **<meta-data android:name="coim_timeout" android:value=timeoutInterval />**，timeoutInterval 為整數，單位為秒。

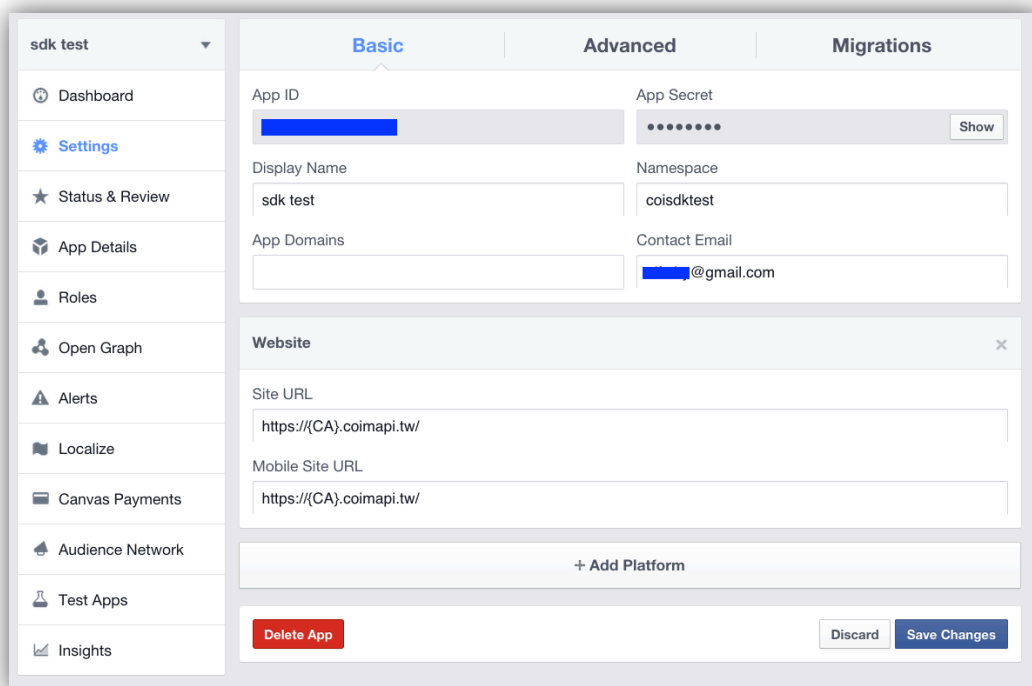
2. iOS

在 plist 中加入” coim_timeout” 的鍵，type 為 number，值則是開發者欲自訂的 timeout 時間，單位為秒。

G. SWS 設定

如果要使用 coim.sws 的功能的話，請進行下列的設定

1. 接下來在專案的 AndroidManifest.xml 中的<application>標籤下加入 coim_fb_app_id 或 coim_gl_app_id 的<meta-data>並填入對應的 app id(請至 Facebook developer 或 Google API console 頁面申請)。
2. 登入 **COIMOTION** 網站，進入”APP 管理”點選要使用的 APP，會顯示”APP: :基本資料”的視窗，將 APP ID 與 APP Secret 填入對應的欄位。
3. 在 Facebook developer 或 Google API console 登錄 redirect_uri
 - a. Facebook 設定如下圖，Add Platform 選擇 Website，site URL 填入 **COIMOTION** 的 API 引擎網址(ex: https://{CA}.coimapi.tw/，CA 為 COIMOTION APP 代碼)



The screenshot shows the Facebook Developer console interface. On the left is a sidebar with navigation links: Dashboard, Settings, Status & Review, App Details, Roles, Open Graph, Alerts, Localize, Canvas Payments, Audience Network, Test Apps, and Insights. The main content area is titled 'Basic' and contains the following fields:

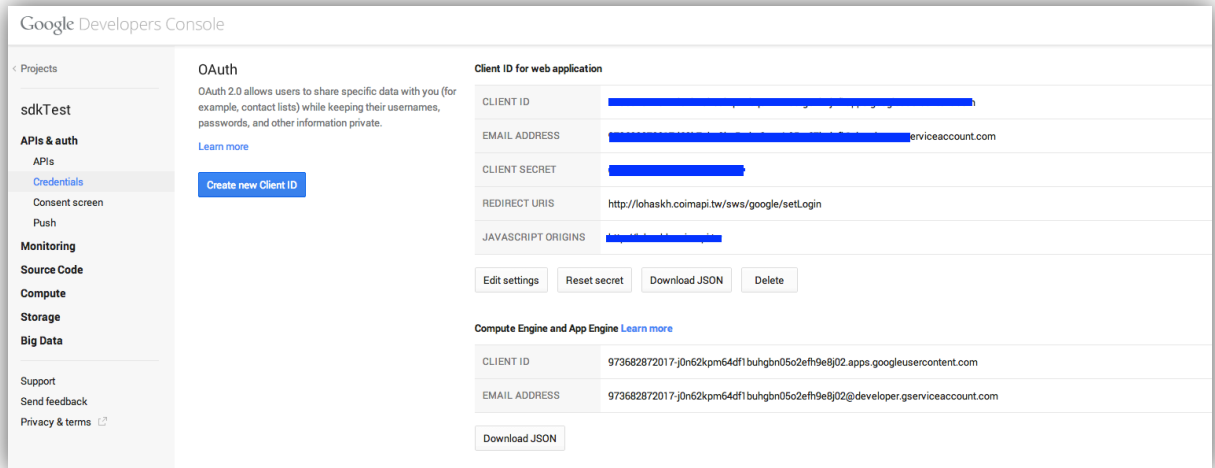
- App ID:** A blue box representing the app ID.
- App Secret:** A field with masked characters and a 'Show' button.
- Display Name:** 'sdk test'.
- Namespace:** 'coisdktest'.
- App Domains:** An empty text field.
- Contact Email:** '[redacted]@gmail.com'.

Below these fields is a 'Website' section with a close button (X). It contains:

- Site URL:** 'https://{CA}.coimapi.tw/'.
- Mobile Site URL:** 'https://{CA}.coimapi.tw/'.

At the bottom of the 'Basic' tab is a '+ Add Platform' button. At the very bottom of the console are three buttons: 'Delete App' (red), 'Discard' (grey), and 'Save Changes' (blue).

- b. Google API console 設定如下圖：redirect uri 為 http://{CA}.coimapi.tw/sws/google/setlogin 。



III. Plugin 文件

本 plugin 提供與 coimSDK 的介接方法與 **callback** 的物件。安裝完成後，會與 cordova API 一同 load，所以必須在” **deviceready**” 這個 event 之後才能確保 plugin 的正常使用 (ref: http://docs.phonegap.com/en/3.5.0/cordova_events_events.md.html#deviceready)。未來會隨著 **COIMOTION** 的成長逐步增加更多的功能。在此版本 plugin 提供有 **callback** 物件與 **COIMOTION** API 呼叫物件：

A. coim.callback

本 plugin 提供 **coim.callback** 物件的方法可以讓開發者使用。此 **callback** 物件提供了 **success**、**fail**、**invalid** 以及 **progress** 四個方法來對應 介接方法所需要的 **callback** 函式。開發者使用 **callback** 物件時可透過實作自己的 **function** 來覆蓋 (override) **callback** 物件中的方法。建立 **callback** 物件方法如下：

```
var cb = new coim.callback();
```

1. success(result)

當 API 呼叫完成時的 **callback**，此 **result** 為 **js** 物件，對應 API 回傳結果 (JSON 格式)，開發者可根據 JSON 的格式來進行剖析。

2. fail(err)

SDK 在處理、發送 API 或解析回應資料時有任何錯誤，會呼叫此方法並將錯誤訊息(**err** 為字串)傳回，開發者可以在此方法中處理各種錯誤的對應。

3. invalid()

COIMOTION API 發送時會透過 App key 與 App code 進行權限的驗證，當發送的 API 中包含 **token**(不為空字串)時，如果該 **token** 已失效，且該 API

為可匿名使用的 API 時，回傳的結果中會增加一個 **token=""** 的欄位，plugin(SDK 亦然)會觸發 **invalid()** 這個 **callback**。此時 plugin 會將儲存的 **token** 這為""，開發者如果預期使用者必須在登入(也就是使用有效 **token**)的狀態下使用 APP 的話。開發者就必須在 **invalid** 中將使用者登出或是讓使用者再登入，以取得有效的 **token**。否則使用者在後續的使用上也許會遭遇到預料外的錯誤(ex：**guest** 無法取得使用者個人資料之類)。

4. progress(progress)

attach 方法會進行檔案的上傳，因此提供 **progress callback** 以回傳上傳進度，傳入的 **progress** 參數為 0-100。

範例

```
var cb = new coim.callback();
cb.success = function(obj) {
  // do your stuff with API results
};
cb.fail = function(err) {
  // do your stuff with err
};
cb.invalid = function() {
  // do your stuff while token invalid
};
cb.progress = function(prog) {
  // do your stuff with UL/DL progress
};
```

B. coim.method

開發者可透過 **coim** 模組提供的方法來使用下述方法與 COIMOTION API 引擎溝通

send: function(relativeURL, params, success, fail, invalid)
send: function(relativeURL, params, coimCallback)

發送 API 至開發者設定的功能元

- **relativeURL** 格式為 **\$WA/\$RS/\$OP[/ \$ID]**。
\$WA 為內容集或模組代碼；\$RS 為資源代碼；\$OP 為功能元代碼；\$ID 則是在需要指定操作目標時給予的目標 ID。
- **params** 為 JS 物件，將要發送的參數以 **key-value** 的方式放入 **params** 中送出，如果沒有參數的話可以使用 **null** 或是不給 **params** 參數。
- **success**、**fail** 與 **invalid** 為 **callback** 函式，分別對應成功、失敗以及 **token** 失效三個事件。
- 另外可使用 **var callback = new coim.callback();**指令產生 **callback** 物件，以第二個形式傳入 **coimPlugin** 函式，產生的 **callback** 物件中，開發者可以自行覆蓋 **success**、**fail**、**invalid** 或 **progress** 方法。

範例

```
coim.send(
  "cms/page/create",
  {"title":"test","summary":"test summary"},
```

```
function(result) {
    // do your stuff with result
},
function(err) {
    // do your stuff with err
},
function() {
    // do your stuff while token invalid
});
```

attach: function(relativeURL, params, files, success, fail, progress)
attach: function(relativeURL, params, files, coimCallback)

上傳檔案至相關功能元(須繼承 cms/page/attach 或 cms/geoLoc/attach) ,
params 請參照相對應的功能元, param 需有 nType 參數(1~5), 定義為:

- 1: icon
- 2: file
- 3: image
- 4: video
- 5: audio

files 為字串陣列, 為要上傳的檔案路徑, 目前限制一次上傳為三個檔案且不大於 1 MB, success、fail 與 progress 為 callback 函式, 分別對應成功、失敗以及上傳進度三個事件; 或是使用 var callback = new coim.callback();指令產生 callback 物件, 以第二個形式傳入 coimPlugin 函式, 產生的 callback 物件中, 開發者可以自行覆蓋 success、fail 或是 progress 方法。

範例

```
coim.attach(
    "cms/page/attach/12345",
    {"title":"test", "nType":"1"},
    files,
    function(result) {
        // do your stuff with result
    },
    function(err) {
        // do your stuff with err
    },
    function(prog) {
        // do your stuff with progress changed
    });
```

login: function(relativeURL, params, success, fail)
login: function(relativeURL, params, coimCallback)

發送登入 API 至開發者設定的功能元(需對應 core/user/login)

- relativeURL 格式為\$WA/\$RS/\$OP[\$ID]。
- params 必需有"accName"與"passwd"。
- success 與 fail 為 callback 函式, 分別對應成功以及失敗兩個事件。
- 或使用 var callback = new coim.callback();指令產生 callback 物件, 以第二個形式傳入。

範例

```
coim.login(
    "core/user/login",
    {"accName":"your_account", "passwd":"your_passwd"},
    function(result) {
        // do your stuff with result
    });
```

```

},
function(err) {
    // do your stuff with err
});

```

```

register: function(params, success, fail)
register: function(params, coimCallback)

```

註冊使用者

- Coimotion 在註冊流程分為兩步，先註冊取得 **actID** 後再啟用該 **actID**，考量到 app 使用者的註冊順暢度，在 SDK 中自動幫註冊的使用者啟用帳號，開發者使用此方法時，如果有成功取得回應即表示該帳號已啟用並取得 **token** 可進行之後的 API 操作。
- **params** 必需有 "accName"、"passwd" 與 "passwd2"。
- **success** 與 **fail** 為 **callback** 函式，分別對應成功以及失敗兩個事件。
- 使用 **var callback = new coim.callback();** 指令產生 **callback** 物件，以第二個形式傳入 **coimPlugin** 函式，產生的 **callback** 物件中，開發者可以自行覆蓋 **success** 或是 **fail** 方法。

範例

```

coim.register(
    {"accName":"your_account", "passwd":"your_passwd", ,
    "passwd2":"confirm_passwd"},
    function(result) {
        // do your stuff with result
    },
    function(err) {
        // do your stuff with err
    });

```

```

updatePasswd: function(relativeURL, params, success, fail)
updatePasswd: function(relativeURL, params, coimCallback)

```

發送密碼修改的 API

- **relativeURL** 格式為 \$WA/\$RS/\$OP[\$ID]。
- **params** 必需有 "oldPasswd"、"passwd" 與 "passwd2"
- **success** 與 **fail** 為 **callback** 函式，分別對應成功以及失敗兩個事件
- 或使用 **var callback = new coim.callback();** 指令產生 **callback** 物件，以第二個形式傳入 **coimPlugin** 函式，產生的 **callback** 物件中，開發者可以自行覆蓋 **success** 或是 **fail** 方法。

範例

```

coim.updatePasswd(
    {"oldPasswd":"old_passwd", "passwd":"new_passwd", ,
    "passwd2":"confirm_passwd"},
    function(result) {
        // do your stuff with result
    },
    function(err) {
        // do your stuff with err
    });

```

```

logout : function(success, fail)
logout : function(coimCallback)

```

將使用者的 **token** 註銷。**success** 與 **fail** 為 **callback** 函式，分別對應成功以及失敗兩個事件；或是使用 `var callback = new coim.callback();`指令產生 **callback** 物件，以第二個形式傳入 **coimPlugin** 函式，產生的 **callback** 物件中，開發者可以自行覆蓋 **success** 或 **fail** 方法。

範例

```
coim.logout(  
  function(result) {  
    // do your stuff with result  
  },  
  function(err) {  
    // do your stuff with err  
  });
```

checkNetwork: function(success, fail)
checkNetwork : function(coimCallback)

檢查網路狀態的方法，如果網路正常，**success** 會回傳 **true**，反之回傳 **false**。如果有其它錯誤的話，會以 **fail** 回傳錯誤訊息。

範例

```
coim.checkNetwork(function(hasNetwork){  
  if(hasNetwork)  
    alert("network available");  
  else  
    alert("no network");  
},  
function(err){  
  alert(err);  
});
```

※在 **iOS** 與 **Android SDK** 中，有關傳送密碼的方法，會將使用者的密碼做加密後才傳出。而此 **plugin** 是介接至 **native** 的 **SDK**，因此同樣的也會對密碼作加密，使用者不用再行處理。

C. coim.sws.method

coim.sws 物件提供 **COIMOTION** 的 **sws** 模組對操作，使用下述方法與 **COIMOTION SWS API** 溝通，使用前需進行 **facebook/google+** 的相關設定。

checkFB: function(scope, success, fail)
checkFB: function(scope, coimCallback)

檢查是否有登入 **Facebook** 以及是否有取得特定權限，有登入且有同意權限的話會成功取得 **access_token** 並進行 **COIMOTION** 的登入 (**sws/fb/setLogin**)；反之會回傳“**FB check fail**”。

scope 為 **facebook** 權限(可不給)，如果沒給的話會預設為“**public_profile**”，權限請參考：

<https://developers.facebook.com/docs/facebook-login/permissions/v2.0#reference>

範例

```
coim.sws.checkFB(  
    function(result){  
        alert(result.message);  
    },  
    function(err){  
        alert(err);  
    });
```

```
loginFB: function(scope, success, fail)  
loginFB: function(scope, coimCallback)
```

開啟 dialog 或 viewController 以 webview 進行 Facebook 的登入互動，如果有登入且同意成當下的 scope 的話會自動取得 access_token 進行 COIMOTION 的登入；如果有新增權限的話，會開啟權限同意的頁面進行互動，同意或略過後才會取得 access_token 進行 COIMOTION 的登入。按 cancel/back 關閉頁面會回傳失敗訊息：“FB login cancel”。

scope 為 facebook 權限(可不給)，如果沒給的話會預設為“public_profile”，權限請參考：

<https://developers.facebook.com/docs/facebook-login/permissions/v2.0#reference>

範例

```
coim.sws.loginFB(  
    function(result){  
        alert(result.message);  
    },  
    function(err){  
        alert(err);  
    });
```

```
FBPostMessage: function(message, success, fail)  
FBPostMessage: function(message, coimCallback)
```

張貼訊息至使用者的 Facebook 塗鴉牆。

- message 為要張貼的訊息。

範例

```
coim.sws.postMessage(  
    "test message",  
    function(result){  
        alert(result.message);  
    },  
    function(err){  
        alert(err);  
    });
```

```
FBPostPhoto: function(params, message, coimCallback)
```

張貼圖片(以及訊息)至使用者的 Facebook 塗鴉牆。

- params 為要 js 物件，以 key-value 存放參數，主要包含下列：
 - url 為要張貼的圖片網址。
 - source 為上傳的圖片內容經 base64 編碼後的字串。如果要給檔案路徑的話，請在路徑前加上“file://”字串。(與 url 同時存在的

話，優先處理 url)

- message 為圖片的相關訊息。message 為圖片的相關訊息(可不給)。

範例

```
coim.sws.postPhoto(  
  "http://25.media.tumblr.com/tumblr_m9cbntJ6Da1qk6hcco1_500.png",  
  "test message",  
  function(result){  
    alert(result.message);  
  },  
  function(err){  
    alert(err);  
  });
```

FBGraph: function(params, success, fail)

FBGraph: function(params, coimCallback)

發送 Facebook Graph API。

- params 為要傳入的參數，格式為 js 物件。根據 Graph API 的文件給予參數，必要的有：
 - method: GET/POST/DELETE
 - endp: end point (請參考 <https://developers.facebook.com/docs/graph-api/reference/v2.1/>)
- API 所需參數請以相同 key-value pair 放入 params。

範例

```
coim.sws.FBGraph(  
  {"method": "GET", "endp": "/me"},  
  function(result){  
    alert(result.message);  
  },  
  function(err){  
    alert(err);  
  });
```

checkGL: function(scope, success, fail)

checkGL: function(scope, coimCallback)

檢查是否有登入 Google 以及是否有取得特定權限，有登入且有同意權限的話會成功取得 code 並進行 COIMOTION 的登入 (sws/google/setLogin)；反之會回傳” GL check fail”。

- scope 為 Google+ OAuth2 權限，如果沒給的話會預設為”<https://www.googleapis.com/auth/plus.login>”，權限請參考：<https://developers.google.com/+/api/oauth#login-scopes>

範例

```
coim.sws.checkGL(  
  function(result){  
    alert(result.message);  
  },  
  function(err){  
    alert(err);  
  });
```

```
});
```

```
loginGL: function(scope, success, fail)
```

```
loginGL: function(scope, coimCallback)
```

開啟 dialog/viewController 以 webview 進行 Google 的登入，如果有登入且同意成當下的 scope 的話會自動取得 access_token 進行 COIMOTION 的登入(sws/google/setLogin)；如果有新增權限的話，會開啟權限同意的頁面進行互動，同意或略過後才會取得 code 進行 COIMOTION 的登入(sws/google/setLogin)。按 cancel/back 關閉頁面會回傳失敗，訊息為” GL login cancel”。

scope 為 Google+ API 權限，如果沒給的話會預設為”

https://www.googleapis.com/auth/plus.login”，權限請參考：

<https://developers.google.com/+/api/oauth#login-scopes>

範例

```
coim.sws.loginGL(  
    function(result){  
        alert(result.message);  
    },  
    function(err){  
        alert(err);  
    });
```

```
googlePlus: function(params, success, fail)
```

```
googlePlus: function(params, coimCallback)
```

發送 Google+ API。

- params 為要傳入的參數，格式為 js 物件。根據 Google+ API 的文件給予參數，必要的有：
 - method: GET/POST/DELETE
 - endp: end point (請參考 <https://developers.google.com/+/api/latest/>)
- API 所需參數請以相同 key-value pair 放入 mapParam。

範例

```
coim.sws.googlePlus(  
    {"method": "GET", "endp": "activities", "query": "exile"},  
    function(result){  
        alert(result.message);  
    },  
    function(err){  
        alert(err);  
    });
```

```
checkLogin: function(success, fail)
```

```
checkLogin: function(coimCallback)
```

檢查登入狀態，以 core/user/profile 檢查現有 token 是否有效。有效的話進一步檢查是否為 Facebook 或 Google+ 登入，是的話以 checkFB/checkGL 以取得有效的 access_token 進行 COIMOTION 登入；不是的話則開發者可讓使用者直接進入 App 主畫面。

回傳的訊息(message 欄位)如果是” Login OK.”，的話表示使用者為合法使用者，開發者可將其導至 app 主畫面或繼續操作；否則為未登入狀態，請導至登入流程。

範例

```
coim.sws.checkLogin(  
  function(result){  
    alert(result.message);  
  },  
  function(err){  
    alert(err);  
  });
```

IV. SWS 相關注意事項

A. 登入

checkLogin 方法提供開發者可在 App 開啟或是返回 App 時判斷使用者是使用哪個方式登入 App，並更新 FB/GL 的 access_token。由於 access_token 有可能會過期，因此開發者必須在使用者重新開啟或返回 App 時更新。

B. 權限

張貼圖文至 FB、呼叫 graph API 以及 Google+ API 時也許會需要不同的權限。因此除非開發者確定已有該對應權限，否則請在進行上述動作前以 loginGL/loginFB 進行權限確認後再發送相關 API。

V. 如有問題的話，可到 **COIMOTION** 網站留言

vi. Appendix

Release note

版號	修改	日期
0.9.6.3	增加 SWS 物件	
0.9.6.2	<ol style="list-style-type: none">1. tag(page/tag 或 geoLoc/tag)現在可接受 Array 物件，開發者可以將 Array 的物件放入 js Object 作為 tag 的值。2. attach (page/attach 或 geoLoc/attach) 如果 files 傳入為空陣列的話，會檢查參數是否帶有 dataURI(外部網址)參數，有的話會以該 dataURI 做為檔案的網址發送 API。	2014/08/05
0.9.6	<ol style="list-style-type: none">3. Timeout 機制 開發者可設定 API 呼叫的 timeout，未設定的話預設為 30 秒。4. checkNetwork 方法 checkNetwork 方法，開發者可在發送 API 之前檢查網路狀況，而不用等到 API 無法發送再處理	2014/07/08
0.9.5	<ol style="list-style-type: none">1. 改提供 coim 模組，安裝後直接可使用 coim.method 來呼叫以及 callback 物件生成改由模組提供(new coim.callback())。2. 增加傳入參數的判斷，send 時不需要參數時，param 可不給或是給 null。3. 增加 debug mode，開發者可在 AndroidManifest 或是 plist 中加入 coim_debug 的旗標，為真的話，console.log 中會輸出 debug 訊息(前置 [DEBUG] -)	2014/06/23
0.9.4	重整 callback 定義，現在所有由 COIMOTION API 回傳的結果將會傳回至 success 或是 coimCallback.success，開發者需進一步判斷 errCode 為 0 才是成功完成 API 呼叫並取得預期結果。	2014/06/13
0.9.3	<ol style="list-style-type: none">1. 對應 native SDK 改版2. 修改 api 網址為".coimapi.tw"3. register API 新增 _loc 參數，以 os 語系為準，目前分中英文，中文以外的語言預設為英文。	2014/06/10

	4. success 在 errCode>=0 時會被呼叫，原先只在 errCode=0 時被呼叫	
0.9.2.1	修正：使用 coimCallback 物件時，在 cordova.exec 中 return false 時對應的 fail 無法被呼叫。	2014/06/09
0.9.2	建立	2014/06/03