

I. 簡介

提本 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**。

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



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，並切換至"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)

III. Plugin 文件

本 plugin 提供與 coimSDK 的介接方法與 callback 的物件，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 直接使用，其中包含一建立 callback 物件的方法可以讓開發者使用，其中提供了 success、fail、invalid 以及 progress 四個方法，對應 JS 介接方法所需要的 callback 函式，因此開發者使用 plugin 時需要實作下敘方法或是以下列方法來覆蓋(override) coimCallback 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**，開發者便可採取對應的處理方式(ex: 將使用者登出)。

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 coimCallback();**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(
  "workshop/beef/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 coimCallback();**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.attch(
  "workshop/page/attach/12345",
  {"title":"test","nType":"1"},
  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 coimCallback();**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 coimCallback();**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 coimCallback();**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 coimCallback();**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
  });
```

※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。

IV. 如有問題的話，可到 **COIMOTION** 網站留言

v. Appendix

Release note

版號	改修	日期
0.9.5	<ol style="list-style-type: none">1. 改提供 coim 模組，安裝後直接可使用 <code>coim.method</code> 來呼叫以及 <code>callback</code> 物件生成改由模組提供(<code>new coim.callback()</code>)。2. 增加傳入參數的判斷，<code>send</code> 時不需要參數時，<code>param</code> 可不給或是給 <code>null</code>。3. 增加 <code>debug mode</code>，開發者可在 <code>AndroidManifest</code> 或是 <code>plist</code> 中加入 <code>coim_debug</code> 的旗標，為真的話，<code>console.log</code> 中會輸出 <code>debug</code> 訊息(前置 <code>[DEBUG]</code> -)	2014/06/19
0.9.4	重整 <code>callback</code> 定義，現在所有由 <code>COIMOTION API</code> 回傳的結果將會傳回至 <code>success</code> 或是 <code>coimCallback.success</code> ，開發者需進一步判斷 <code>errCode</code> 為 0 才是成功完成 <code>API</code> 呼叫並取得預期結果。	2014/06/13
0.9.3	<ol style="list-style-type: none">1. 對應 <code>native SDK</code> 改版2. 修改 <code>api</code> 網址為 <code>"*.coimapi.tw"</code>3. <code>register API</code> 新增 <code>_loc</code> 參數，以 <code>os</code> 語系為準，目前分中英文，中文以外的語言預設為英文。4. success 在 <code>errCode>=0</code> 時會被呼叫，原先只在 <code>errCode=0</code> 時被呼叫	2014/06/10
0.9.2.1	修正：使用 <code>coimCallback</code> 物件時，在 <code>cordova.exec</code> 中 <code>return false</code> 時對應的 <code>fail</code> 無法被呼叫。	2014/06/09
0.9.2	建立	2014/06/03