

# Mobile Social Plugin

Anyone can comment

---

[Short Overview](#)

[How to update](#)

[Getting Started](#)

[Setup](#)

[Twitter](#)

[Setup](#)

[API References](#)

[Coding Guidelines](#)

[Facebook](#)

[Setup](#)

[API References](#)

[Coding Guidelines](#)

[More Social Networks](#)

[Instagram](#)

[Platform Native Sharing API](#)

[Platform Behavior differences](#)

[Frequently Asked Questions](#)

[PlayMaker](#)

[Example Scenes](#)

[How to get support](#)

# Short Overview

---

Plugin provides the functionality to use Twitter and Facebook api on IOS and Android devices. You can write code once and be sure that it will work the same on your IOS and Android app. However some API actions may looks different, on IOS / Android even if it do absolutely the same for example posting/authorization. You can find out more about differences in API References.

Twitter API includes:

- User Authentication
- Loading User Data
  - id
  - name
  - description
  - screen\_name
  - location
  - lang
  - status
  - profile images urls
  - profile images
  - and much more
- Posting
- Posting with image

Facebook API:

Provided facebook api as wrapper class around [Unity Facebook SDK](#). Which means all Unity Facebook SDK will be completely available for you.

Wrapper include API for:

- User Authentication
- Loading User Data
  - id
  - name
  - first\_name
  - last\_name
  - email

- location
- locale
- profile images urls
- profile images
- and much more
- Posting
- Posting with image
- Loading friends information

#### Instagram API

- Share Image
- Share image with text

# How to update

---

## 1. Version Notes

With every new update I make try to make plugin better. Add new features, improve stability, usability and code base structure.

When new version is available, you can find out what's new in the version and version history by pressing version number on [Asset Store Plugin Page](#):

### Mobile Social Plugin

Category: Scripting/Integration  
Publisher: Stan's Assets  
Rating: ★★★★★ (17)  
Price: \$15  
Buy \$15.00

Requires Unity 4.3.0 or higher.

Plugin provides the functionality to use Twitter and Facebook API on IOS and Android devices. You can write code once and be sure that it will work the same on your IOS and Android app.

However some API actions may looks different, on IOS / Android even if it do absolutely the same for example posting/authorization. You can find out more about differences in Documentation

Online Documentation | Forum Thread

Source Code is Open! (eclipse project included)  
Fullv compatible with:

Version: 2.5 (May 29, 2014) Size: 23.0 MB

Support E-mail Support Website Visit Publisher's Website

## 2. Avoiding conflicts

Sometimes in order to implement new feature or improve code structure I have to change some of plugin files / folder or method names.

It will be of course described in version notes. But if you simple click update in Asset Store version, you may get duplicated or conflicted files.

**Warning:** Check the Version Notes before update. If version notes contains

**Code Refactor:** version section. It means that plugin structure is changed. Some files can be removed or function names changed. If files was removed, you should remove it by your self from the project. Example or Remove notes:

Removed:

Assets/Extensions/AndroidNative/Other/Twitter

All Scene under xExample/Scenes Moved to corresponding folder  
only Preview Scene should be under xExample/Scenes

**Note:** If you own another plugins with also have `GooglePlayCommon` folder (this folder is shared between few plugins in order to supply compatibility of android plugins) I also recommend update those plugins too. To avoid conflicts

### ***3. Saving Plugins settings***

Plugin setting that was specified in editor GUI earlier will be overridden. So just backup your settings data with stored in files:

Assets/Extensions/GooglePlayCommon/Resources/SocialSettings

And replace plugin files with your backup after update.

# *Getting Started*

Some of plugin function working out of the box, more advanced function requires additional setup action. I recommend to read Coding Guidelines before you will start to implement social network support to your project.

If you have any difficulties or unexpected errors, feel free to contact [support team](#)

# Setup

Plugin may require ew small set up action in order to compile in your device with no issue.

Copy files from

**Assets/Plugins/StansAssets** to → **Assets/Plugins**

If you already have some files under **Assets/Plugins/Android** it mean that you may already have some android plugin installed. In this case, please read this sections with will help you to merge the plugins:

[Can I use this plugin with other Android Plugins from Asset Store](#)

[How to compile androidnative.jar from eclipse project](#)

[How to merge manifest with another android plugin](#)

If the plugin has conflict with the IOS / WP8 / Android (with not overrides main activity)  
Please [contact support team](#).

## IOS

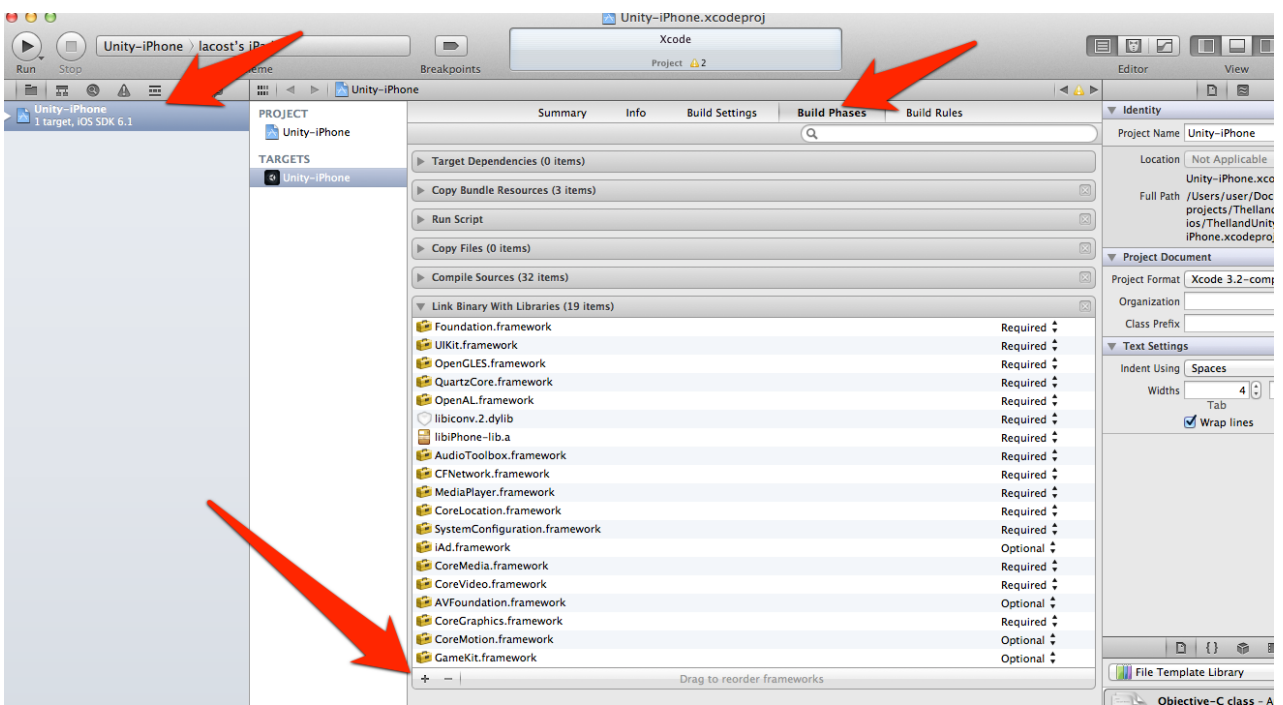
You should include following frameworks to your IOS XCode project

**Accounts.framework**

**Social.framework**

Here is instruction how to do that.

Open your project. Click on The Solution Name (project setting will open). Click on  
**"Build Phase Tab", C**



click on "+" button on "Link Binary".



# Twitter

## Twitter Setup

In order to implement twitter oAuth in your application you need twitter **consumer key** and **consumer secret** which are used to make twitter API calls. So register a new twitter application and get the keys

1. Go to <https://dev.twitter.com/apps/new> and register new application. Fill application name, description and website.

2. Give some **dummy url in the callback url field** to make the app as browser app. (If you leave it as blank it will act as Desktop app which won't work in mobile device)

3. Under the **settings** tab upload icon and change the access type to **Read and Write**.

Application details

**Name \***

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

**Description \***

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

**Website \***

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens. (If you don't have a URL yet, just put a placeholder here but remember to change it later.)

**Callback URL**


Where should we return after successfully authenticating? [OAuth 1.0a](#) applications should explicitly specify their oauth\_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

**any url, we not going to use it**

4. Copy **Consumer Key & Consumer Secret key**

## Application settings

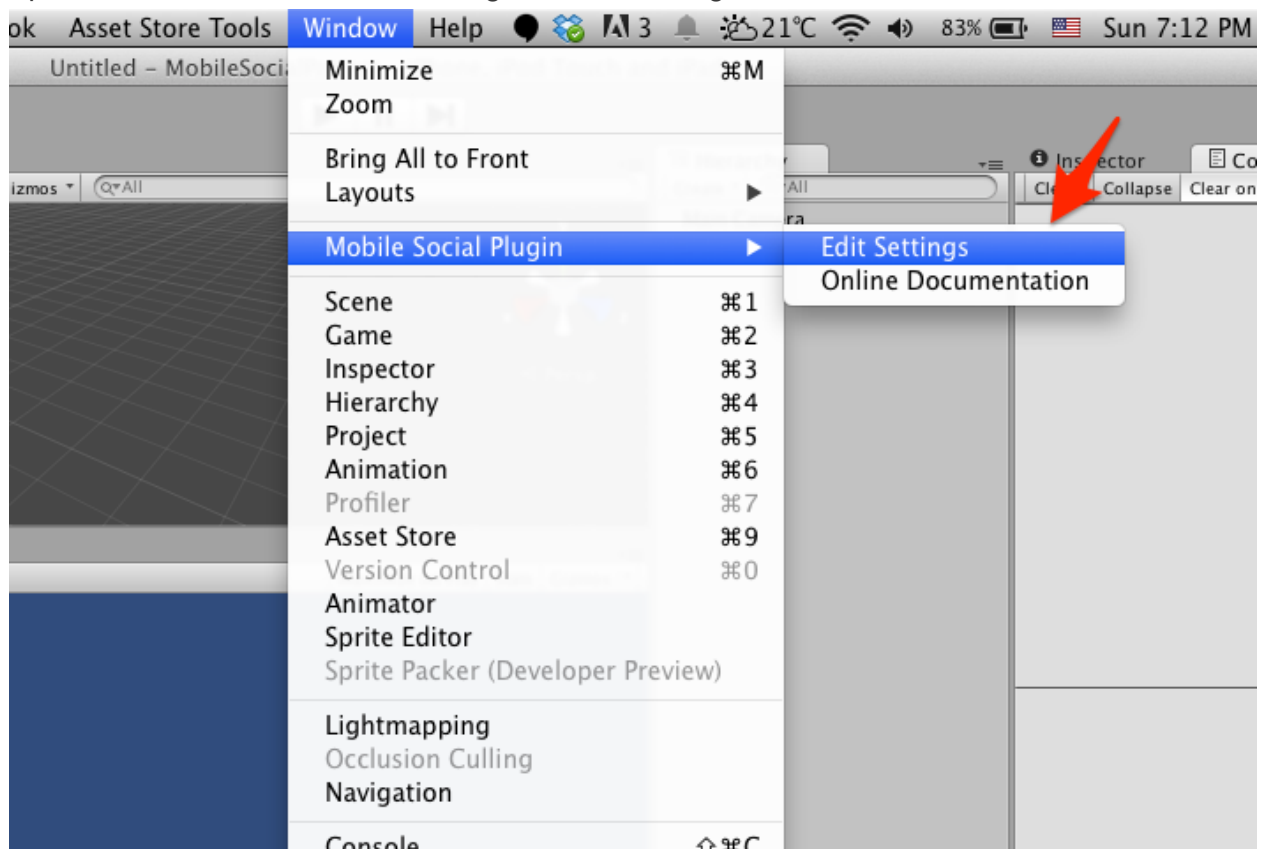
*Keep the "API secret" a secret. This key should never be human-readable in your application.*

API key	wEvDyAUr2QabVAsWPDiGwg
API secret	igRxZbOrkLQPNLSvibNC3mdNJ5tOIVOPH3HNNKDY0
Access level	Read, write, and direct messages ( <a href="#">modify app permissions</a> )
Owner	LacostSt
Owner ID	

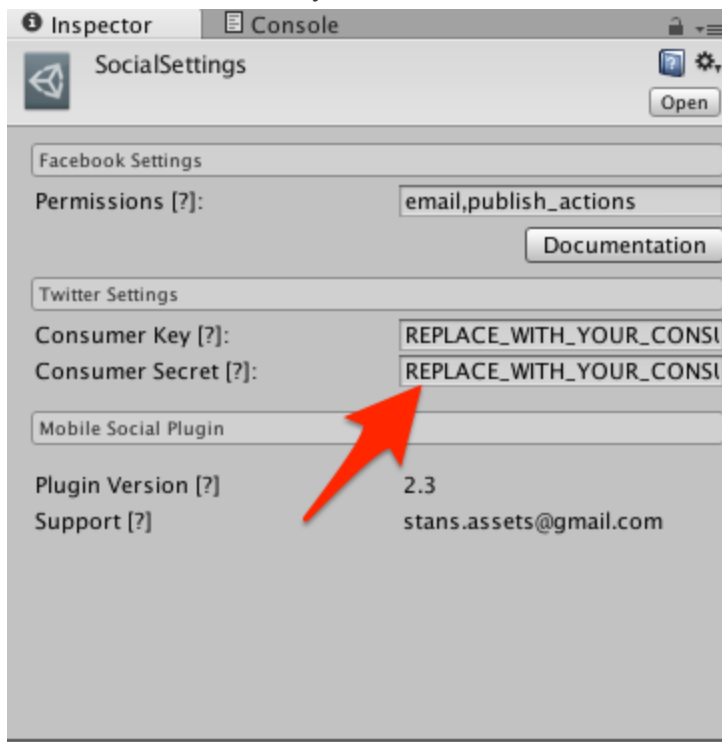
You done. Now you can start using Twitter API in your project using copied secret keys.

Last step is to specify consumer(API) key and consumer(API) secret in the plugin setting. As option you can also do this using coding, but is much easier to use GUI.

Open Windows → Mobile Social Plugin → Edit Settings



And fill in Consumer key and secret.



The image shows a screenshot of the 'SocialSettings' Inspector window in a development environment. The window has a title bar with 'Inspector' and 'Console' tabs. Below the title bar, there's a 'SocialSettings' header with a settings icon and an 'Open' button. The main content area is divided into sections for different social media settings:

- Facebook Settings:** Includes a 'Permissions [?]:' field with the value 'email,publish\_actions' and a 'Documentation' button.
- Twitter Settings:** Includes 'Consumer Key [?]:' and 'Consumer Secret [?]:' fields, both containing the placeholder text 'REPLACE\_WITH\_YOUR\_CONSI'.
- Mobile Social Plugin:** Includes a 'Plugin Version [?]' field with the value '2.3' and a 'Support [?]' field with the value 'stans.assets@gmail.com'.

A red arrow points to the 'Consumer Secret [?]:' field, highlighting the placeholder text.

## Twitter API References

SPTwitter : Singleton<SPTwitter> class.

---

### API methods:

*Init twitter consumer key and secret will be used from plugin setting.*

*You may modify settings in plugin windows:*

*Windows → Mobile Social Plugin → Edit Settings*

*Triggers TwitterEvents.TWITTER\_INITED event.*

`public static void Init()`

*Init twitter controller using consumer key and secret. Triggers*

*TwitterEvents.TWITTER\_INITED event.*

`public static void Init(string consumer_key, string consumer_secret)`

*Start user authentication. Triggers TwitterEvents.AUTHENTICATION\_SUCCEEDED or TwitterEvents.AUTHENTICATION\_FAILED Note, that after initialization user can already be authed, so use this method only if after initialization user not authed.*

`public static void AuthenticateUser()`

*Loads user data. Triggers TwitterEvents.USER\_DATA\_LOADED and*

*TwitterEvents.USER\_DATA\_FAILED\_TO\_LOAD events.*

`public static void LoadUserData()`

*Post a Tweet*

`public static void Post(string status)`

`public static void Post(string status, Texture2D texture )`

*Authenticate a user and then Post using Twitter API.*

`public TwitterPostingTask PostWithAuthCheck(string status)`

`public TwitterPostingTask PostWithAuthCheck(string status Texture2D texture)`

*Resets the user authorization*

`public static void LogOut()`

### Get / Set:

*Returns twitter controller interface*  
`public static` TwitterManagerInterface twitter

## TwitterManagerInterface

---

### Get / Set:

*True if user authorized*  
`bool` IsAuthed

*Returns user info if it was successfully loaded*  
TwitterUserInfo userInfo

## TwitterUserInfo : EventArgs

---

### API methods:

*Loads Profile Image. Triggers PROFILE\_IMAGE\_LOADED event*  
`public void` LoadProfileImage()

*Loads Profile Background Image. Triggers PROFILE\_BACKGROUND\_LOADED event*  
`public void` LoadBackgroundImage()

### Get / Set:

*User id*  
`public string` id;

*Description*  
`public string` description;

*User name*  
`public string` name;

*User Login*

`public string screen_name;`

*Current Location*

`public string location;`

*Language code*

`public string lang;`

*full user info in JSON format*

`public string rawJSON;`

*http and https url to user profile image*

`public string profile_image_url;`

`public string profile_image_url_https;`

*http and https url to user profile background*

`public string profile_background_image_url;`

`public string profile_background_image_url_https;`

*contains user profile images if was previously loaded*

`public Texture2D profile_image = null;`

*contains user profile background images if was previously loaded*

`public Texture2D profile_background = null;`

*twitter page background color*

`public Color profile_background_color = Color.clear;`

*twitter page text color*

`public Color profile_text_color = Color.clear;`

*friends count*

`public int friends_count;`

*tweets count*

`public int statuses_count;`

*current user status*

`public TwitterStatus _status`

## TwitterStatus

---

### Get / Set

*current status text*

`public string` text;

*Location*

`public string` geo;

*unparsed status response from twitter*

`public string` rawJSON;

## TwitterPostingTask

---

### Events

*Events fired when the posting task is complete, Event contains [TWResul](#) as event data.*

BaseEvent.[COMPLETE](#)

## TWResult

---

### Get / Set

*true if action was successful*

`public bool` IsSucceeded

*contains result response data*

`public string` data;



## Twitter Coding Guidelines

There is two general ways you can go with twitter integration.

1) If you want to implement only posting, and you do not need to know if a user posted a message or cancelled the posting sequence

In this case you can use native platform API.

### Advantages:

- You do not need to create your Twitter app
- You do not need to prompt user to authenticate with your app with requested permissions before you can use Twitter API
- User will not leave your app for Twitter authentication.
- Posting looks natively to platform

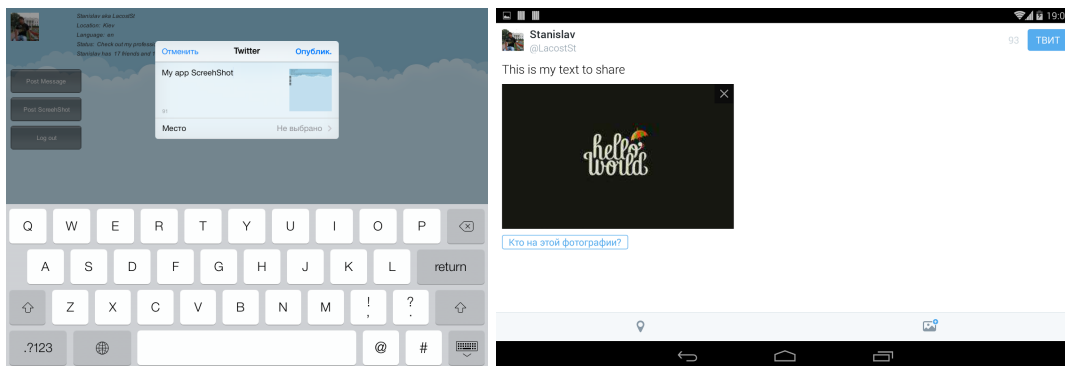
### Disadvantages

- You can only prompt user to post messages / images to twitter.
- You will not know or be able to respond if the user cancels their post from the twitter app.

If you decide to go this way, you may use the following API calls from anywhere in your project to prompt the user to tweet a message or image, using Twitter's UI:

```
SPShareUtility.TwitterShare("This is my text to share");  
SPShareUtility.TwitterShare("This is my text to share", ImageToShare);
```

Result of this api calls would be similar to screenshot below:



You may read more about native shared behavior on different platforms at this [section](#).

2) If you need a lot more, and you want to use all plugin power.

### Advantages:

- You will be able use all feature offered by the plugin
- More permissions from user
- Full control of user actions

### Disadvantages

- You have to create app on twitter.
- Extra work with rendering message windows, since we are calling api directly

If you decide to go this way. Then you need to complete [Setup Steps](#) before you begin to use Twitter API.

I would recommend to create singleton class which will be responsible for all your game twitter interactions.

At the game start you should subscribe to twitter event you interested in and init twitter API by calling:

```
SPTwitter.instance.Init();
```

And example how you can subscribe to the events.

```
SPTwitter.instance.addListener(TwitterEvents.TWITTER_INITED, OnInit);  
SPTwitter.instance.addListener(TwitterEvents.AUTHENTICATION_SUCCEEDED, OnAuth);  
SPTwitter.instance.addListener(TwitterEvents.AUTHENTICATION_FAILED, OnAuthFailed);
```

```
SPTwitter.instance.addListener(TwitterEvents.POST_SUCCEEDED, OnPost);  
SPTwitter.instance.addListener(TwitterEvents.POST_FAILED, OnPostFailed);
```

```
SPTwitter.instance.addListener(TwitterEvents.USER_DATA_LOADED, OnUserDataLoaded);  
SPTwitter.instance.addListener(TwitterEvents.USER_DATA_FAILED_TO_LOAD,  
OnUserDataLoadFailed);
```

After Twitter API is initied, you can check if user is already authed (he may complete the auth in previous game sessions.) Here is example how to do this in OnInit Function.

```
private void OnInit() {  
    if(SPTwitter.instance.IsAuthed) {  
        //user is authed  
        //we can use API calls any time  
    } else {  
        // user not authed, we need to use authenticate user first if we want to  
        use API  
    }  
}
```

```
}  
}
```

If user is Authenticated after init call, feel free to use any API Methods described in [API References](#). If not, then you should Authenticate before you can call API methods. To authenticate the user you should call:

```
SPTwitter.instance.AuthenticateUser();
```

Example how to subscribe to authentication events:

```
SPTwitter.instance.addEventListener(TwitterEvents.AUTHENTICATION_SUCCEEDED, OnAuth);  
SPTwitter.instance.addEventListener(TwitterEvents.AUTHENTICATION_FAILED, OnAuthFailed);
```

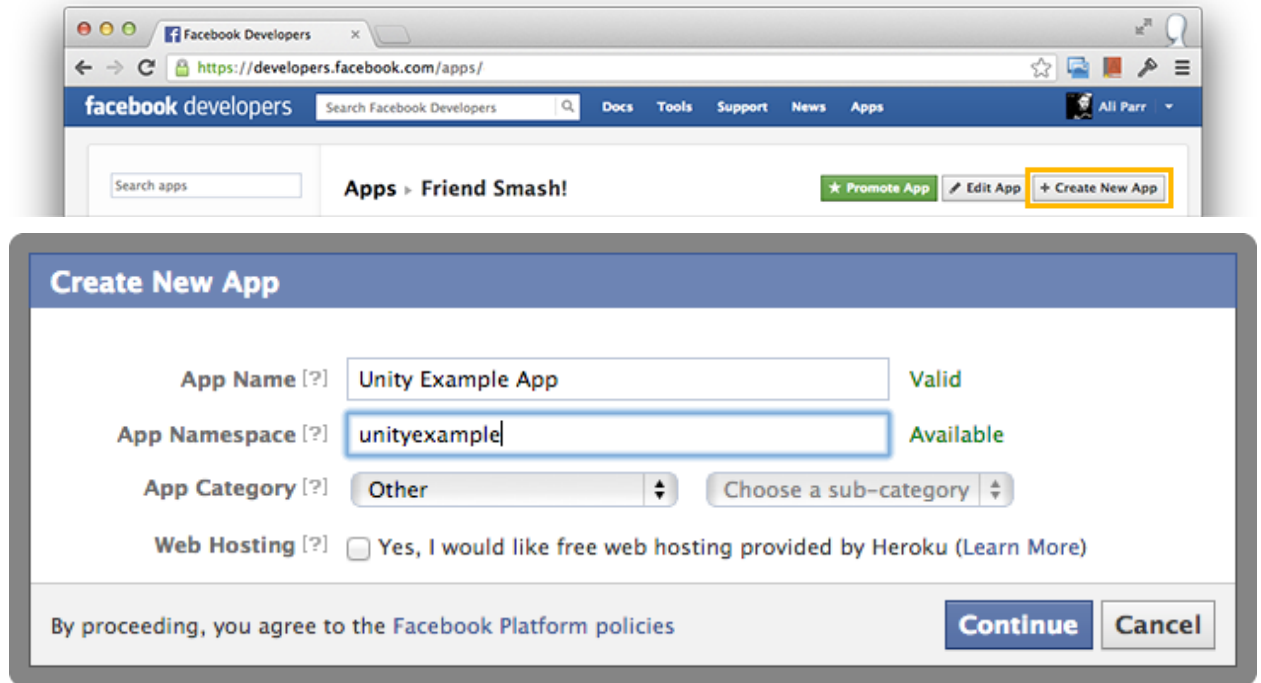
As soon as you done with auth, feel free to use [Twitter API](#).

Do not forget to read about [Platform Behavior Differences](#). Since we using direct API calls we can not force Android for example to draw twitter native post window, all posting will be done in background, so as developer you should render posting window using your game style by your self.

# Facebook

## Facebook Setup

**Step 1:** Navigate to your [App Dashboard](#) and click '+ Create New App'. In the popup dialog, give your new app a name and optionally a unique namespace, click 'Continue' and follow the subsequent instructions.



The screenshot shows a web browser window with the Facebook Developers App Dashboard. The URL is <https://developers.facebook.com/apps/>. The page title is "facebook developers" and the search bar contains "Search Facebook Developers". The main content area shows "Apps > Friend Smash!" with buttons for "Promote App", "Edit App", and "+ Create New App". The "+ Create New App" button is highlighted with a yellow border. Below the main content area, a "Create New App" dialog is open. The dialog has a blue header with the text "Create New App". It contains the following fields and options:

- App Name** [?]: "Unity Example App" (Valid)
- App Namespace** [?]: "unityexample" (Available)
- App Category** [?]: "Other" (Choose a sub-category)
- Web Hosting** [?]: ☐ Yes, I would like free web hosting provided by Heroku ([Learn More](#))

At the bottom of the dialog, there is a text line: "By proceeding, you agree to the [Facebook Platform policies](#)". To the right of this text are two buttons: "Continue" and "Cancel".

**Step 2:** Once completed, you'll see your app's Basic Settings page. Here you can find your **App ID**, which is required within your Unity game's configuration, in order to integrate with Facebook.

## Apps ▸ Unity Example App ▸ Basic



**Unity Example App**  
**App ID:** 412806198828612  
**App Secret:** 39102f9d731c6f33ca636575793e25b2 (reset)  
**This app is in Sandbox Mode** (Only visible to Admins, Developers and Testers)

**Basic info**

**Display Name:** [?]

Unity Example App

**Namespace:** [?]

unityexample

**Contact Email:** [?]

**App Domains:** [?]

Enter your site domains and press enter

**Hosting URL:** [?]

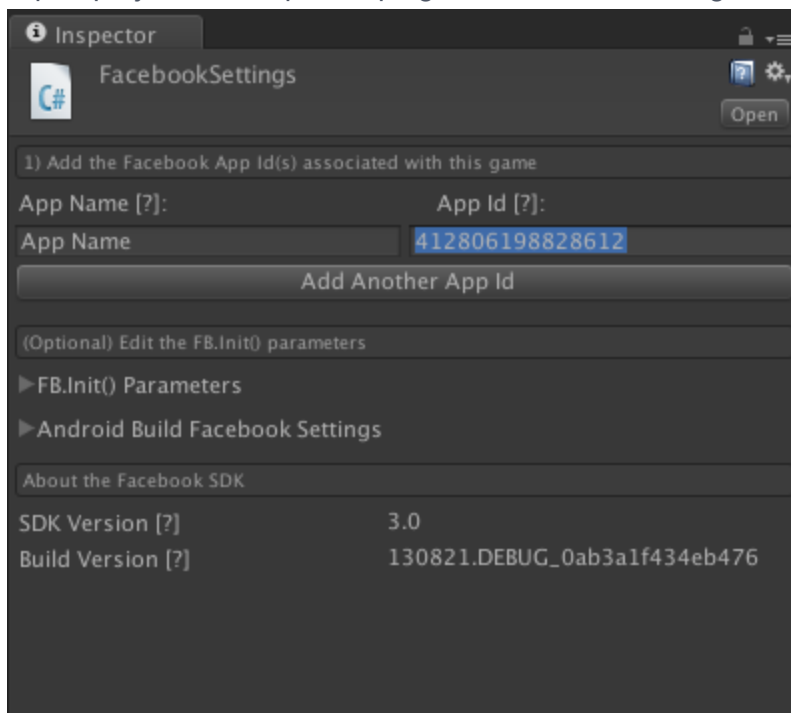
You have not generated a URL through one of our partners ([Get one](#))

**Sandbox Mode:** [?]

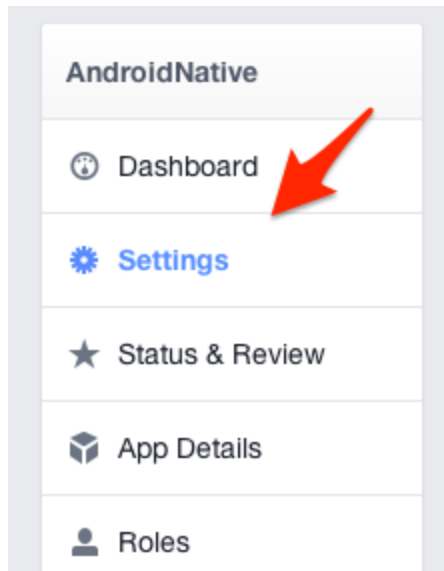
☒ Enabled ☐ Disabled

### Step 3:

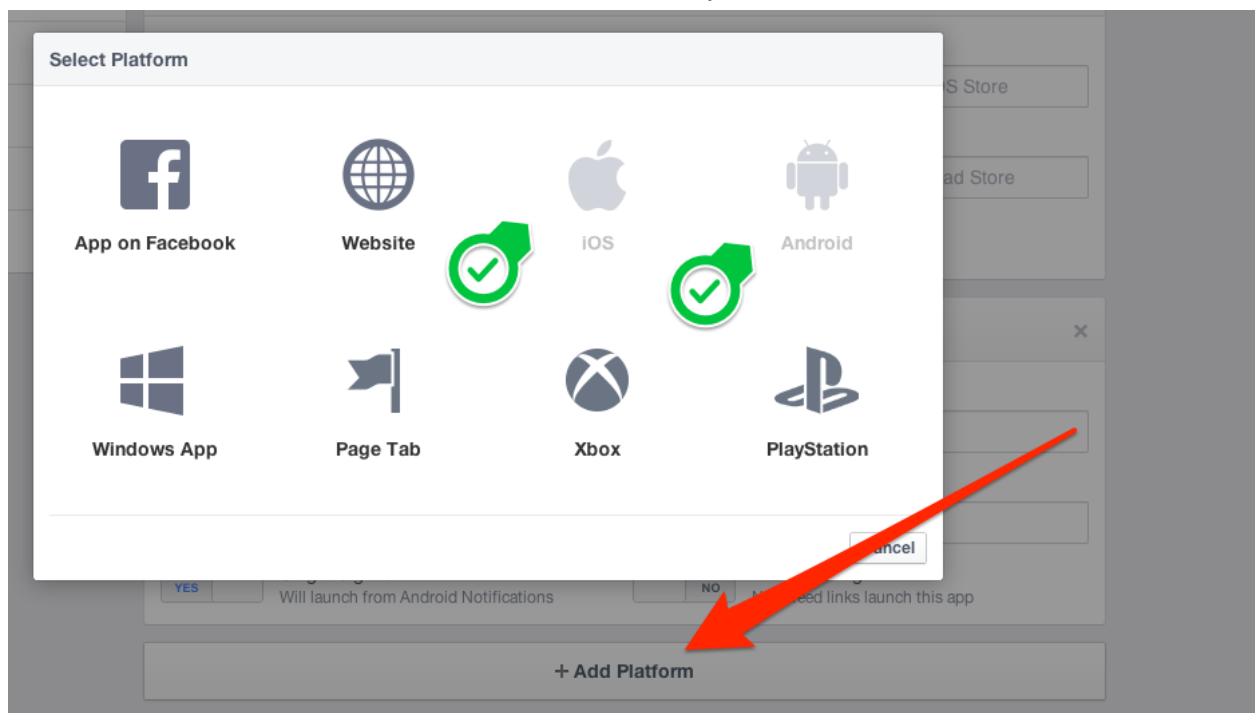
Open project with imported plugin. Select 'Edit Settings' from the 'Facebook' menu.



Go to your app settings on facebook



Click on Add Platforms and add IOS and Android platform



IOS Set up Part:

Set your app bundle Id. You can also specify iPhone and iPad Store ID if you have one.

1. Download the openssl for windows [here](#)
2. now unzip to c drive

3. open cmd prompt
4. type `cd C:\Program Files\Java\jdk1.6.0_26\bin`
5. then type only `keytool -export -alias myAlias -keystore C:\Users\your user name\.android\myKeyStore | C:\openssl-0.9.8k_WIN32\bin\openssl sha1 -binary | C:\openssl-0.9.8k_WIN32\bin\openssl enc -a -e`
6. Done

password: **android.**

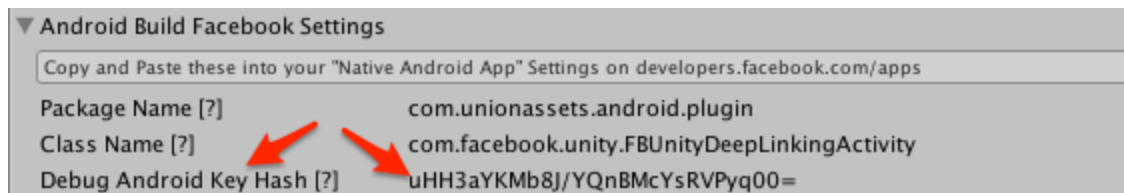
If you already have your keystore, use this to get Key Hash:

```
keytool -exportcert -alias <alias_name> -keystore <path_to_keystore> | openssl sha1  
-binary | openssl base64
```

If you don't you can use [this instruction](#) to create your app keystore.

**Note:** you can create your keystore, in any way you like. Just make sure you specified Key Hash FB app setting accordingly to keystore you building with

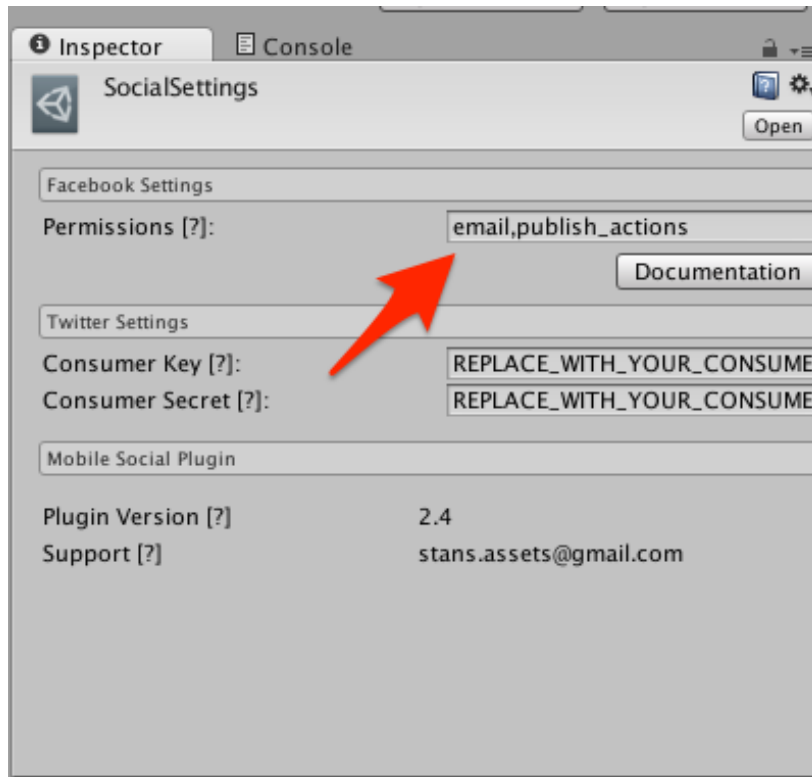
**Warning:** Key Hash you see from Facebook → Edit Settings window. Is Key Hash for debug (unsigned) app. This is not your Key Hash of you building using keystore





## Step 4: Specify login scopes

Open plugin setting window. Window → Mobile Social Plugin → Edit Settings



You can leave default permissions, or change if you need more. You can see the [here](#) all available scopes and it descriptions.

You can also do this with Login Function. Here is example

```
SPFacebook.instance.Login("email,publish_actions");
```

If you using function without parameters:

```
SPFacebook.instance.Login();
```

Plugins will use scopes (permissions) specified in plugin settings window.

**Note:** Including **publish\_actions** triggers a Facebook review.

The basic permissions that don't require a review are

*public\_profile, email, user\_friends.*

But without **publish\_actions** you will not be able to use

`SPFacebook.instance.Post` function.

That's it you can now go to [API Reference](#) section to find out how to use Facebook and API in your app.

## Facebook API References

SPFacebook : Singleton<SPFacebook> class.

---

### API methods:

*Init facebook Triggers FacebookEvents.FACEBOOK\_INITED event.*

```
public void Init()
```

*Start user authentication with scopes. Triggers FacebookEvents.AUTHENTICATION\_SUCCEEDED or FacebookEvents.AUTHENTICATION\_FAILED Note, that after initialization user can already be authed, so use this method only if after initialization user not authed.*

*If you using methods without parameters, Plugin will use scopes from [plugin settings](#).*

```
public void Login()
```

```
public void Login(string scopes)
```

*Loads user data. Triggers FacebookEvents.USER\_DATA\_LOADED and FacebookEvents.USER\_DATA\_FAILED\_TO\_LOAD events.*

```
public static void LoadUserData()
```

*Post to facebook*

```
public void Post (
```

```
    string told = "",  
    string link = "",  
    string linkName = "",  
    string linkCaption = "",  
    string linkDescription = "",  
    string picture = "",  
    string actionName = "",  
    string actionLink = "",  
    string reference = ""
```

```
)
```

```
public void PostImage(string caption, Texture2D image)
```

*Authenticate a user and then Post using FB API.*

```
public FBPostingTask PostWithAuthCheck(  
    string told = "",
```

```
string link = "",
string linkName = "",
string linkCaption = "",
string linkDescription = "",
string picture = "",
string actionName = "",
string actionLink = "",
string reference = ""
)
```

*Sends application app request. Similar to [Unity Facebook API method](#).*  
`public void` AppRequest(...)

*Resets the user authorization*  
`public static void` Logout()

## Get / Set

*True if user authorized*  
`public bool` IsLoggedIn

*User id*  
`public string` UserId

*Session access token*  
`public string` AccessToken

*Returns user info if it was successfully loaded*  
`public` FacebookUserInfo userInfo

*Returns friends data if it was successfully loaded*  
`public` Dictionary<`string`, FacebookUserInfo> friends  
`public` List<`string`> friendIds  
`public` List<FacebookUserInfo> friendsList

## FacebookUserInfo : EventDispatcherBase

---

### API methods:

*Returns profile image url.*

`public string` GetProfileUrl(FacebookProfileImageSize size)

*Returns profile image if it was previously loaded*

`public Texture2D` GetProfileImage(FacebookProfileImageSize size)

*Loads Profile Image. Triggers PROFILE\_IMAGE\_LOADED event*

`public void` LoadProfileImage(FacebookProfileImageSize size)

### Get / Set:

*User id*

`public string` id;

*User name*

`public string` name;

*User first name*

`public string` first\_name;

*User Last name*

`public string` \_last\_name;

*Username*

`public string` \_username;

*Facebook profile url*

`public string` \_profile\_url;

*Location*

`public string` location;

*Language code*

`public string` \_locale;

*full user info in JSON format*

`public string` rawJSON;

*gender*

`private` GoogleGenger \_gender

## FBPostingTask

---

### Events

*Events is fired when the posting task is complete, Event contains [FBResult](#) as event data.*

BaseEvent.[COMPLETE](#)

## ***Facebook Coding Guidelines***

The Facebook API is implemented using [Unity Official Facebook SDK](#)

### ***What does it mean?***

First of all Unity Facebook SDK will be imported in your project with the plugin. And most of facebook plugin functionality is a wrapper around Unity Facebook implementation. It also mean that Unity Facebook SDK integrated with the plugin and has no conflicts, which makes you able to use it (if for some reason you do not like the way facebook stuff is implemented in this plugin). Or use function from Unity Facebook SDK. Like for example direct API call using [FB.AP](#) function.

And of course [all other stuff](#) offered by Unity SDK is available for you.

### ***What the difference between Unity SDK and Mobile Social Plugin?***

There is couple difference:

- Another workflow.
- Few extra features based on Unity SDK like for example:
  - Posting screenshot in background
  - Posting with auto authentication check
- Few feature not available from Unity SDK like for example:
  - Platform Native Image Posting
  - Platform Native Text Posting
- Ability to use [PlayMaker Actions](#) instead coding.
- And all other feature you got with the plugin instead Facebook.

There is two general ways you can go with Facebook integration. Pretty much same as Twitter

1) If you want to implement only posting, and you do not need to know if a user posted the message or cancelled the posting sequence.

In this case you can use native platform API.

#### **Advantages:**

- You do not need to create your Facebook app
- You do not need to prompt user to authenticate with your app with requested permissions before you can use Facebook API
- User will not leave your app for Facebook authentication.

- Posting looks native to platform

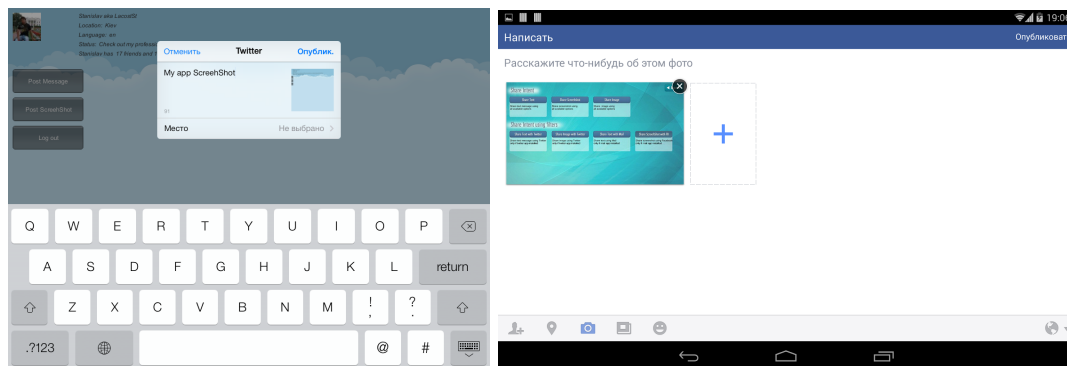
### Disadvantages

- You can only prompt user to post messages / images to facebook.
- You will not know or be able to respond if the user cancels their post from the facebook app
- You will not able to post text on Android, due to facebook restrictions.

If you decide to go this way, you may use the following API calls from anywhere in your project to prompt the user to tweet a message or image, using Facebook's UI:

```
SPShareUtility.FacebookShare("This is my text to share");
SPShareUtility.FacebookShare("This is my text to share", ImageToShare);
```

Result of this api calls would be similar to screenshot below:



You may read more about native shared behavior on different platforms at this [section](#).

2) If you need a lot more, and you want to use all plugin power.

### Advantages:

- You will be able use all feature offered by the plugin
- More permissions from user
- Full control of user actions

### Disadvantages

- You have to create app on twitter.
- Extra work with rendering message windows, since we are calling api directly

If you decide to go this way. Then you need to complete [Setup Steps](#) before you begin to use Facebook API.

I would recommend to create singleton class which will be responsible for all your game Facebook interactions.

At the game start you should subscribe to Facebook event you interested in and init Facebook API by calling:

```
SPFacebook.instance.Init();
```

And example how you can subscribe to the events.

```
SPFacebook.instance.addListener(FacebookEvents.FACEBOOK_INITED, OnInit);
SPFacebook.instance.addListener(FacebookEvents.AUTHENTICATION_SUCCEEDED, OnAuth);
SPFacebook.instance.addListener(FacebookEvents.AUTHENTICATION_FAILED, OnAuthFailed);

SPFacebook.instance.addListener(FacebookEvents.USER_DATA_LOADED, OnUserDataLoaded);
SPFacebook.instance.addListener(FacebookEvents.USER_DATA_FAILED_TO_LOAD, OnUserDataLoadFailed);

SPFacebook.instance.addListener(FacebookEvents.FRIENDS_DATA_LOADED, OnFriendsDataLoaded);
SPFacebook.instance.addListener(FacebookEvents.FRIENDS_FAILED_TO_LOAD, OnFriendDataLoadFailed);

SPFacebook.instance.addListener(FacebookEvents.POST_FAILED, OnPostFailed);
SPFacebook.instance.addListener(FacebookEvents.POST_SUCCEEDED, OnPost);

SPFacebook.instance.addListener(FacebookEvents.GAME_FOCUS_CHANGED, OnFocusChanged);
```

After the Facebook API is initied, you can check if user is already authed (he may complete the auth in previous game sessions.) Here is example how to do this in OnInit Function.

```
private void OnInit() {
    if (SPFacebook.instance.IsLoggedIn) {
        //user is authed
        //we can use API calls any time
    } else {
        // user not authed, we need to use authenticate user first if we want to
        use API
    }
}
```



If user is Authenticated after init call, feel free to use any API Methods described in [API References](#). If no, then you should Authenticate before you can call API methods. To authenticate the user you should call:

```
SPFacebook.instance.Login();
```

Example how to subscribe to authentication events:

```
SPFacebook.instance.addListener(FacebookEvents.AUTHENTICATION_SUCCEEDED, OnAuth);  
SPFacebook.instance.addListener(FacebookEvents.AUTHENTICATION_FAILED, OnAuthFailed);
```

As soon as you done with auth, feel free to use [Facebook API](#).

Do not forget to read about [Platform Behavior Differences](#). uld render posting window using your game style by your self.

# More Social Networks

## Instagram

---

Instagram implementation based on platform API.

As result:

- 1) It works out of the box, no extra work from your side.
- 2) API Methods will only work if instagram app is installed on device. Otherwise corresponded error code will be returned.

To Share texture or message with texture, you can call this API from any place of your app:

```
SPInstagram.Share(imageForPosting);  
SPInstagram.Share(imageForPosting, "I am posting from my app");
```

You may also listen for the posting result. You need to subscribe for posting events. Here is example how to do this:

```
SPInstagram.addListener(InstagramEvents.POST_SUCCEEDED, OnPost);  
SPInstagram.addListener(InstagramEvents.POST_FAILED, OnPostFailed);
```

**Note:** POST\_SUCCEEDED event will be raised even if user cancelled posting inside Instagram app.

If you got Failed Posting event, you can find out reason why. Fail event will contain one of following constants as error code:

```
public enum InstaErrorCode {  
    NO_APPLICATION_INSTALLED,  
    USER_CANCELLED,  
    SYSTEM_VERSION_ERROR,  
    INTERNAL_EXCEPTION  
}
```

Example of OnPostFailed method implementation:

```
private void OnPostFailed(CEvent e) {  
    InstaErrorCode error = e.data as InstaErrorCode;  
    Debug.Log("Posting failed with error code " + error.ToString())  
}
```

# Native Sharing API References

SPShareUtility :

---

## API methods:

*Share message to Twitter using native platform API*

```
public static void TwitterShare(string status)
```

*Share message and texture to Twitter using native platform API*

```
public static void TwitterShare(string status, Texture2D texture)
```

*Share message to Facebook using native platform API*

```
public static void FacebookShare(string message)
```

*Share message and texture to Facebook using native platform API*

```
public static void FacebookShare(string message, Texture2D texture)
```

*Share message using all available apps (user will be prompted to create one)*

```
public static void ShareMedia(string caption, string message)
```

*Share message and texture using all available apps (user will be prompted to create one)*

```
public static void ShareMedia(string caption, string message, Texture2D texture)
```

# Platform Behavior differences

---

At this chapter I will describe how plugin function will behave on different platforms.

## **Facebook**

`public void Login()`

### **IOS:**

User will be redirect to the web browsed or installed official facebook app in order to give access to your application

### **Android:**

Same as IOS

Note: After initialization, you do not have to use Login function if the user is already logged in.

`public void Post (...)`

### **IOS:**

Facebook style pop-up will appear before posting. User will have ability to modify text before post and cancel/accept post.

### **Android:**

Same as IOS

`public void PostImage(string caption, Texture2D image)`

### **IOS:**

All posting happens in background. User will not see any pop-ups.

### **Android:**

Same as IOS

## ***Twitter***

```
public static void AuthenticateUser()
```

### ***IOS:***

Native dialog will appear to request user permission to use his Twitter account specified in device settings.

### ***Android:***

User will be redirect to the web browsed in order to give access to your application.

```
public static void Post(string status)
```

### ***IOS:***

IOS style pop-up will appear before posting. User will have ability to modify text before post and cancel/accept post.

### ***Android:***

Posting goes in background. User will see preload until posting is in process.

```
public static void Post(string status, Texture2D texture )
```

### ***IOS:***

IOS style pop-up will appear before posting. User will have ability to modify text before post and cancel/accept post.

### ***Android:***

Posting goes in background.

## ***Native Sharing***

### ***IOS:***

Sharing to Facebook / Twitter only possible if user specified Facebook / Twitter account info in the device settings. If there is not info, sharing pop up will be created any way, but user will be prompted to fill in account info before sharing.

### ***Android:***

Facebook message will be ignored (how ever you steel can share image) due to Facebook policy.

# *Example scenes*

---

Basic example scene elements.

This is active button. You can press it to run plugin API example. Button name and button description should explain what exactly will happen after you press it.

The inactive button. The function is locked due to some reason. In most cases it is waiting for service initialization.

The status bar, is displaying what is currently happening and status of the current API process. It may also contain additional explanation.

Under this chapter you will find small description and use instruction of example scene you will find in the plugin.

First of all, the example scenes are located under.

This is preview scene with only one purpose - allow you to switch between plugin example scene.

For example you can do all necessary plugin setup, and then just compile with all example scene included, and make sure the example scene has 0 index to be the first loaded scene.

The twitter example scene will show you basic twitter api you can use with the plugin.

The platform API calls work out of the box and do not require any set-up from you.

The twitter OAuth API works out of the box also, but it uses the plugin's twitter app. If you want to use your own app, you have to finish [Twitter Setup Guide](#) and update your **Consumer (API) Key** and **Consumer(API) Secret** in the example script.

The facebook example scene will show you basic facebook api you can use with the plugin.



The platform API calls works out of the box and do not require any set-up from you.  
The facebook oAuth API requires finishing of [Facebook Setup](#) section

# *PlayMaker Actions*

---

The plugin now contains playmaker actions.

The actions scripts can be found in the rar archive at:

**Assets/Extensions/MobileSocialPlugin/Addons/PlayMakerActions.zip**

You can simply un - zip it to the same folder and IOS Native action will appear under playmaker actions menu. You always welcome on the [PlayMaker Actions Forum Thread](#) to request new actions or report a bug

The current actions list is:

- **Facebook**
  - MSPFacebookLogIn
  - MSPFacebookLogOut
  - MSPFacebookPost
  - MSPFacebookPostTexture
- **Twitter**
  - MSPTwitterLogIn
  - MSPTwitterPost
  - MSPTwitterPostTexture
  - MSTwitterLogOut
- **Instagram**
  - MSPInstaPostTexture

# Example Scenes

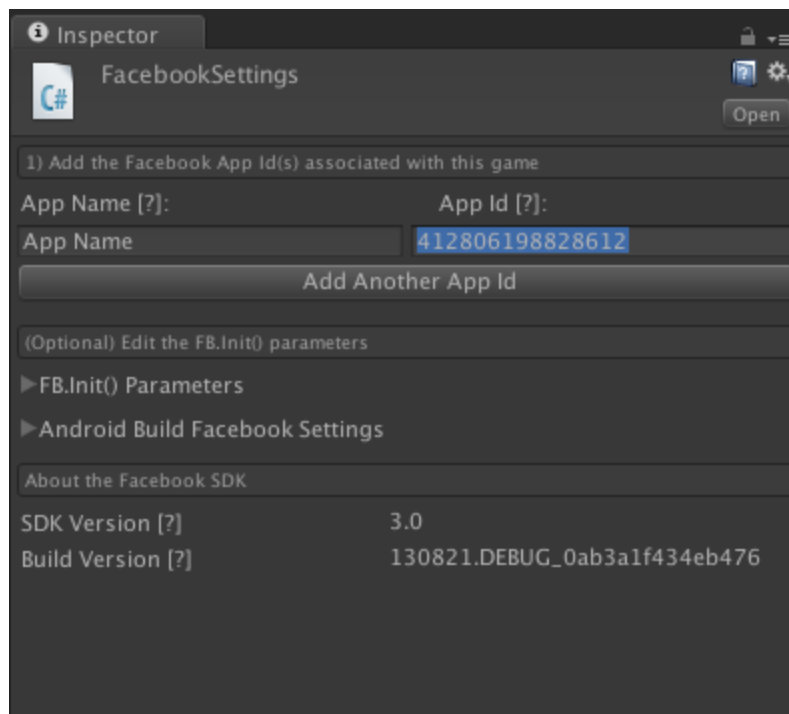
---

## FacebookExample

This example scene can be found at Assets/Extensions/MobileSocialPlugin/Example/Scenes/FacebookExample. It will describe how you can use facebook API. The controller script MSPFacebookUseExample is attached to the \_Controller gameobject and provides example for API calls.

## FacebookExample[Setup]

You need complete facebook [setup](#). Copy **app\_name** and **app\_id** from FB developers and enter to Facebook Menu -> Edit settings:



Two ways of using key hash:

If you chose the way of getting the connection to facebook via the debug key hash, you should FB and in Unity to specify the debug key hash. If simple key, then do the same manipulations

---

## ***TwitterExample***

This example scene can be found at Assets/Extensions/MobileSocialPlugin/xExample/Scenes/TwitterExample. It will describe how you can use twitter API. The controller script MSPTwitterUseExample is attached to the \_Controller gameobject and provides example for API calls.

### ***TwitterExample[Setup]***

You need complete twitter app [setup](#). Enter to MSPTwitterUseExample your app\_key and app\_secret\_key:

```
private static string TWITTER_CONSUMER_KEY = "YOUR_KEY";  
private static string TWITTER_CONSUMER_SECRET = "YOUR_SECRET_KEY";
```

---

## ***MoreExamples***

This example scene can be found at Assets/Extensions/MobileSocialPlugin/xExample/Scenes/MoreExamples. It will describe how you can use instagram API. The controller script MSPMoreActionsExample is attached to the \_Controller gameobject and provides example for API calls.

### ***MoreExamples[Setup]***

You need complete instagram [setup](#). Before that, naturally set himself instagram on your device.

---

## ***MSP\_Preview***

This example scene can be found at Assets/Extensions/MobileSocialPlugin/xExample/Scenes/MSP\_Preview. It will describe how you can use instagram, twitter and facebook API. The controller script MSPMoreActionsExample is attached to the \_Controller gameobject and provides example for API calls.

### ***MSP\_Preview[Setup]***

On this stage is a review of two social API(Facebook and Twitter), their documentation, examples, and the Transition to the stage to use instagram.