Facebook Backend 2014

COMPREHENSIVE GUIDE TO USE THE MODULE

Introduction

With this guide you'll get the propper instructions to use the "fbHelper" module, according with the specifications delivered via mail. The purpose of this guide is not to teach you how to configure the facebook-side of the functions and actions, but just to show you how to perform this actions in the client side.

This guide assume that you know how to setup facebook achievements, actions (needed to create and post "stories"), and every other thing not-facebook related.

Please note that this plugin is as much generic as possible.

All the documentation of this guide refers to the "fbHelper.lua" module.
All the examples refers to the demo-app provided with this guide.

Also note that you need to specify the facebook's app id in the "bulid.settings" twice. Please see the "facebook backend" demo app to see an example of the configuration.

You can see an example app in the project folder. Please note that this example app assumes that you have all the things properly configurated in the facebook side.

What's new?

With the Graph 2.0 API, Facebook have introduced a wide list of changes that affect on how apps manage information and user's data. The filosophy behind the changes is to preserve users privacy and to promote the Facebook's website.

To preserve users privacy, now **all the calls to "friends_ ... " are deprecated and can't be used**, and the "user_friends" permission **is not enabled by default**. In addition of that, an app have all the permission calls disabled by default (except for the default public profile and user friends permissions), and **your app will need to be reviewed by Facebook** (in the same manner Apple do with his Appstore), **in order to allow you to ask for permissions other than the default**.

In order to promote the Facebook's website, now **you can't get a full list of user's friends if your app doesn't have a CANVAS version** (HTML5 or Flash, hosted on your servers). This is a "timid" step to avoid those games that are taking advantage of Facebook's integration without having a canvas version (and that causes Facebook doesn't take advantage of having more games published on his platform). So, **if you want to have a customized UI to invite friends to play your game, you'll need to publish a Canvas version of the game**.

It's still possible to invite friends using the standard feed dialog as always. (In fact, now is mandatory to use the feed dialog even if you have a Canvas version to take the final step and effectively invite friends). More on this in the following documentation.

Functions changed

fbHelper.getFriends now retrieve the friends that are playing your game ONLY (not the full list of friends).

fbHelper.postScore, now your app needs to be reviewed by the Facebook team in order to let you request for the permission required (publish_actions) and pos scores.

All the permissions are now updated to minimize the risk of requesting a permission that requires app review to be requested.

Facebook is now a plugin for iOs on Corona, you need to propperly get it on "bulid.settings".

New Functions

fbHelper.getInvitableFriends that allows you to get a JSON table with all the user's invitable friends, and construct a custom interface with their pictures, names and tokens.

fbHelper.openInviteDialog, that allows you to invite specifically the friends selected on a custom interface created via fbHelper.getInvitableFriends

Initial setup

In order to let the plugin working properly, you need to configure and get some information directly from facebook before using this module. Concretely, you need to get:

- **The "Facebook APP ID"**: mandatory and needed for absolutely all the functions of this module. **THIS IS A STRING**.
- **The "appToken"**: mandatory, and needed for some functions. Please don't confuse this with the "user token" or "event.token" delivered by Corona. This is a different token that is generated only once and lives for all the life of your app. More info about how to get this token here: https://developers.facebook.com/docs/opengraph/howtos/publishing-with-app-token/ .**THIS IS A STRING**
- **The "appNameSpace"**: needed for some functions of this module, you can get this information in the "apps" section of your facebook developer account. **STRING**.
- **defaultText**: the text that will be used for the footer of the messages posted in the user's wall. **THIS IS A STRING**.
- **defaultImg**: the filename, including the extension, of the default image to post within the messages (just in case you don't specify an image when calling the proper function). **THIS IS A STIRNG**
- **defaultSrc**: the default directory where to search the image. **A CORONA API CALL**.
- **defaultLink**: the default URL to share when posting a message with URL, just in case you don't specify an URL. **THIS IS A STRING**.

ALL THE FIELDS ARE REQUIRED and needs to be configured in the "initial setup" section of the fbHelper module

Please note that you can experiment and get some information (such like the "appToken") using the Facebook's Graph API Explorer:

https://developers.facebook.com/tools/explorer/

Calling functions

All functions can be called using an "options" table that contains the parameters needed to let the function work properly.

The last parameter for all the functions is the "listener", a function that will catch the result of the operation called. In the following instructions you'll know the name of the function to call, the parameters that you need to pass via the "options" table, and the result you'll get vie the "event" table (just like the "event listeners" works in Corona).

In case of error, all functions return an "event.isError" flag with "true" value, and an "event.errorMessage" with the error string.

In case of success, all functions returns an "event.success" flag with the "true" value, and the proper information requested.

In order to use the module, please require it using the former call for LUA 5.0 an above. All the exampes of this guide assume that you're requiring the module using: `fbHelper = require( "fbHelper" )`

fbHelper.login(options)

This function is very simple, just login and authorize the app to publish and perform actions. With this function we get the user's name and facebook ID.

Options:

- options.listener: the name of the listener function that will be called on error or success. Optional.

Returns:

event.success = true (if the action is performed successfully)
event.id = the user's facebook unique ID
event.name = the user's facebook former name
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function loginStatus( event )
	if event.success then
		print( "The name of the user is " .. event.name )
		print( "The ID   of the user is " .. event.id )
	elseif event.isError then
```

```
            native.showPopup( "Error", event.errorMessage, {"OK"} )
      end
end

local options = {}
options.listener = loginStatus

fbHelper.login( options )
```

fbHelper.getFriends(options) - Updated!

With this function you can get a table containing the names and unique IDs of the user **friends playing your game**. Also, this function will automatically download the friends profile pictures into the Temporary Directory, using the ID as name and with "jpg" extension. This function is smart, and only downloads the pictures that are not downloaded yet. (Expect that the first time this function is called it delays a bit while downloads all the images. I suggest a spinner or "please wait" dialog while this occur).

Options:

- options.listener: the name of the listener function that will be called on error or success. Optional.

Returns:

event.success = true (if the action is performed successfully)
event.data = the table containing friend names and IDs
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function response( event )
      if event.success then
            for i = 1, #event.data do
                  print( "The name of the friend is " .. event.data[i].name )
                  print( "The ID   of the friend is " .. event.data[i].id )
                  print( "The friend's pic name is " .. event.data[i].id .. ".jpg" )
            end
      elseif event.isError then
            native.showPopup( "Error", event.errorMessage, {"OK"} )
      end
end

local options = {}
options.listener = response

fbHelper.getFriends( options )
```

fbHelper.getInvitableFriends(options) - New!

This function let you show the invitable friends using a custom interface, and it's the only way to get the full user's friend list.

WARNING: Please note that you need to have a CANVAS version of your game in order to use this function. A Canvas version is a HTML5 or Flash version of your game published directly on Facebook (and hosted on your own servers). Please note that Corona will have the ability to publish an app directly on HTML5 soon :)

WARNING: You will NEED to use this function in combination with **fbHelper.openInviteDialog** to effectively invite friends. In other words: you can't invite friends directly, but only using the feed dialog.

Options:

- options.listener: the name of the listener function that will be called on error or success. Optional.
- options.downloadFriendPictures: a boolean value (true or false) to specify if you want to automatically download the friend's profile pictures. PLEASE NOTE: that pictures will be downloaded using the "id" (also known as token) as name, followed by the ".jpg" extension.

Returns:

event.success = true (if the action is performed successfully)
event.data = the table containing friend names, IDs and scores. Inside of this table the're several sub-tables containing important user info:
event.data[index].id = **This is the TOKEN needed to invite a friend** (string)
event.data[index].name = The friend's name (string)
event.data[index].picture.data.url = The friend's profile picture URL (string)
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function response( event )
	if event.success then
		for i = 1, #event.data do
			print( "The name of the friend is " .. event.data[i].name )
			print( "The TOKEN of the friend is " .. event.data[i].id )
			print( "The picture url is " ..  event.data[i].picture.data.url )
		end
	elseif event.isError then
		native.showPopup( "Error", event.errorMessage, {"OK"} )
	end
end

local options = {}
options.listener = response

fbHelper.getInvitableFriends( options )
```

fbHelper.openInviteDialog(options) - New!

This function opens a Facebook Dialog with the friends picked up from the previous function (**fbHelper.getInvitableFriends**). In other words, opens a feed dialog with a direct invitation to play your game, that will be delivered only to the friend's tokens selected in the getInvitable friends function. So, you will need to call this function (openInviteDialog) ONLY AFTER you have successfully called the **fbHelper.getInvitableFriends** function.

Options:

- options.message: the message that will appear in the friend's wall.
- options.tokens: a string with all the invitable friend's tokens, separated by commas.

Returns:

nothing

Example of usage:

```
local options = {}
options.message = "Play my game!!"
options.tokens = "INVITE_TOKEN_1, INVITE_TOKEN_2, INVITE_TOKEN3"


fbHelper.openInviteDialog( options )
```

MORE INFORMATION: Please refer to the official Facebook documentation to get more information on how and why to use the new Graph API 2.0 functions:

https://developers.facebook.com/docs/games/requests/v2.0

fbHelper.getFriendScores(options)

This function is similar to the "getFriends" one, except that will return a table with the friends names, ids and Scores.

Options:

- options.listener: the name of the listener function that will be called on error or success. Optional.

Returns:

event.success = true (if the action is performed successfully)
event.data = the table containing friend names, IDs and scores. Inside of this table the're is a sub-table containing the user info (event.data[i].user)
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function response( event )
	if event.success then
		for i = 1, #event.data do
			print( "The name of the friend is " .. event.data[i].user.name )
			print( "The ID   of the friend is " .. event.data[i].user.id )
			print( "The friend's score is " .. event.data.score )
		end
	elseif event.isError then
		native.showPopup( "Error", event.errorMessage, {"OK"} )
	end
end

local options = {}
options.listener = response

fbHelper.getFriendScores( options )
```

fbHelper.postScore(options) - Updated!

This function is very simple, just post a new user score on Facebook for your app.

WARNING!: This function needs the "**publish_actions**" permission, which request is not allowed until your app have been reviewed by the Facebook team. Please consider that a review can take up to 7 business days.

Options:

- options.score: the score to post. Must be a number. Mandatory.
- options.listener: the name of the listener function that will be called on error or success. Optional.

Returns:

event.success = true (if the action is performed successfully)
event.scorePosted = the score posted in Facebook
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function response( event )
	if event.success then
		print( "Score posted is: " .. event.scorePosted )
	elseif event.isError then
		native.showPopup( "Error", event.errorMessage, {"OK"} )
	end
end

local options = {}
options.score    = 200
options.listener = response

fbHelper.postScore( options )
```

fbHelper.postAchievement(options)

This function just post a new Achievement on Facebook for your app.

Options:

- options.achievURL: the URL of the achievement. Mandatory.
- options.listener: the listener function that called on error or success. Optional.

Returns:

event.success = true (if the action is performed successfully)
event.id = the ID of the achievement posted in Facebook
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function response( event )
	if event.success then
		print( "Achievement ID is: " .. event.id )
	elseif event.isError then
		native.showPopup( "Error", event.errorMessage, {"OK"} )
	end
end

local options = {}
options.achievURL = "http://your.achievement.com"
options.listener  = response

fbHelper.postAchievement( options )
```

fbHelper.postStory(options)

This is maybe the most complex function to call. In this function you need to pass two tables of parameters. The first one is the "options" table, as usual. The second, inserted into the options table, is a "params" table that contains the information about the object. Note that, to construct a story, you need two values, an action and an object (Example "eat an apple", where "eat" is the action and "apple" is the object).

Options:

- options.action: the name of the action. Mandatory.
- options.params: the table that contains the object details. At least you need to specify the name of the object as key-value, and the URL of the object (but you can pass as much params as you need). Mandatory.
- options.listener: the name of the listener function that will be called on error or success. Optional.

Returns:

event.success = true (if the action is performed successfully)
event.id = the id of the story posted in Facebook
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function response( event )
	if event.success then
		print( "Story ID is: " .. event.id )
	elseif event.isError then
		native.showPopup( "Error", event.errorMessage, {"OK"} )
	end
end

local options = {}
options.params    = {}
options.nameOfTheObject = "URL of the object
options.action = "nameOfTHeAction"
options.listener = response

fbHelper.postStory( options )
```

IMPORTANT: in the instructions you specify that I need to let this as open as possible. I understand that you know that the actions needs to be defined in the "apps" section on developer.facebook.com, and that needs to be defined per-app. More information about how to setup actions for your app here: https://developers.facebook.com/docs/opengraph/creating-action-types/

fbHelper.postMessage(options)

This is a very complete function that allows you to post a message or image, using (or not) an UI where the user can edit the message.

Options:

- options.showUI: true or false, specifies if show or not the UI (where the user can edit the message). If set to false, the message/image is directly posted. Mandatory.
- options.message: the message to post. If "showUI" is set to true, this is the message that the user will see and will be able to edit. Optional.
- options.picture: the name of the picture to post, including the extension. Optional.
- options.source: the source directory of the picture, used as usual (in other words: this is NOT a string). Optional.
- options.linkURL: the URL of the link. If not specified, it will catch the default value.
- options.caption: the caption for the link. Optional.
- options.listener: the name of the listener function that will be called on error or success. Optional.

Returns:

event.success = true (if the action is performed successfully)
event.id = the id of the message posted in Facebook
event.isError = true (if the call is not successfull)
event.errorMessage = string with the error message

Example of usage:

```
local function response( event )
	if event.success then
		print( "MESSAGE ID is: " .. event.id  )
	elseif event.isError then
		native.showPopup( "Error", event.errorMessage, {"OK"} )
	end
end

local options = {}
options.showUI = true
options.message = "Hello from here!!"
options.picture = nil
options.source  = system.ResourceDirectory
options.caption = "Download my app for FREE now!"
options.linkURL = "http://www.tocatoca.es"
options.listener = response

fbHelper.postMessage( options )
```

fbHelper.openFeedDialog(options)

This function opens a Facebook Dialog with the list of friends to invite to play your app. Please note that this is the only method to perform this operation: direct posting from the app in the friends wall is no more allowed.

Options:

- options.message: the message that will appear in the friend's wall.
- options.linkURL: the URL of the link. If not specified, it will catch the default value.

Returns:

nothing

Example of usage:

```
local options = {}
options.message = "Play my game!!"
options.linkURL = "http://www.tocatoca.es"


fbHelper.openFeedDialog( options )
```

fbHelper.logOut( )

This function directly logs out from facebook.

Options:

nothing

Returns:

nothing

Example of usage:

```
fbHelper.logOut()
```