Facebook API

Contents

Introduction

This PDF manual is designed for you to use as a reference to the different Facebook API demo objects and scripts, and as such does not contain tutorials on how to set up the Facebook API in your games. If you wish information on setting up, general use, etc… then please see the following YoYo Games Knowledge Base article:

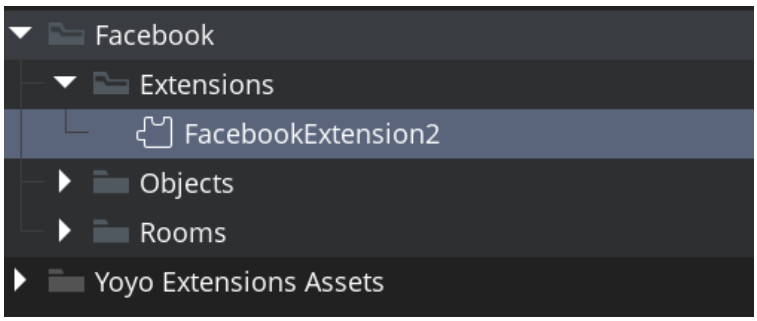
- iOS, Android and HTML5: Integrating Facebook

The rest of this PDF manual is dedicated to the description of the demo, the API, and its objects and scripts.

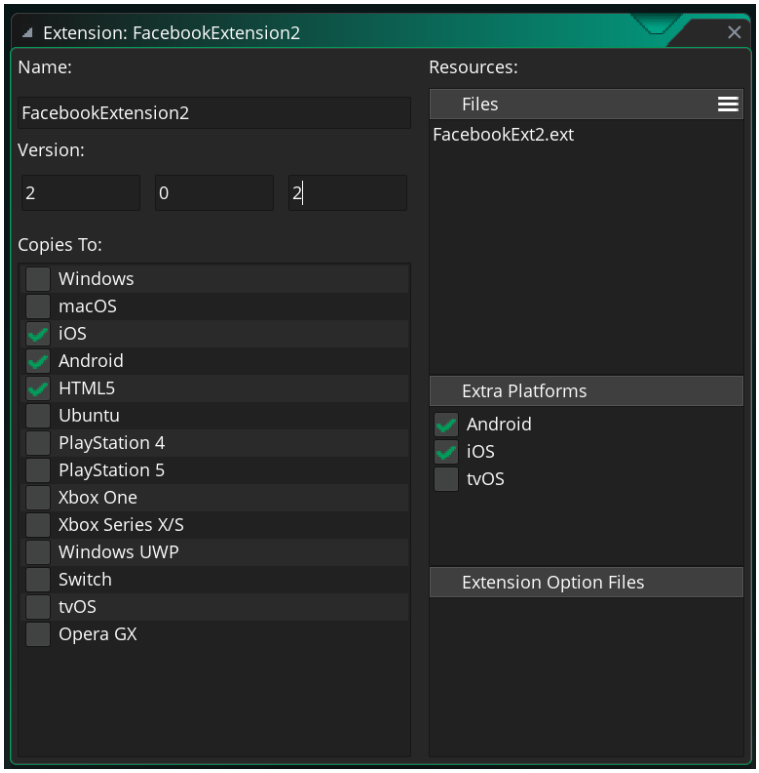
Using the Demo

The Facebook API demo contains both the general API extension, as well as objects and rooms. If you have already set up a game on the Facebook Developer dashboard, or have followed the initial setup guide (see here), then you can start testing the API using the demo.

Since GameMaker Studio v2.3.6, setup for your project is done inside the extension itself and will depend on your target platform Android/iOS. First go into the **FacebookExtension2** options by double clicking the extension on your asset browser:

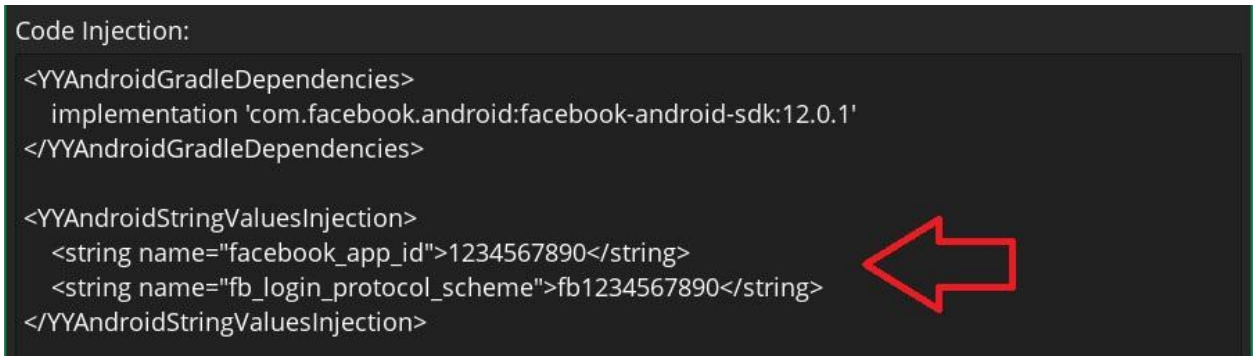


This will bring up the following panel,



from here you need to select Android/iOS from the **Extra Platforms** tab;

If you are targeting **Android** you will have to use your **Facebook App ID** and use it as displayed below:



you need to change both **facebook_app_id** (to your Facebook App ID) and **fb_login_protocol_scheme** (to the Facebook App ID prefixed with "**fb**").

If you are targeting **iOS** you will need to change the panel with the information below:

In the first section you will need your **FacebookAppID**, the **FacebookClientToken** (that you can get from the Facebook dashboard under **Settings -> Advanced -> Client Token**) and the **FacebookDisplayName** which is the name of your dashboard application. In the second section you will need to replace the value there with your Facebook App ID prefixed with "**fb**".

Once you have filled in those details, you are ready to test the demo. Simply plug in your device and set the target platform then run the demo. Once it runs, try logging in then then each of the other buttons and pay attention to the debug output to ensure that everything works correctly.

The Asynchronous Social Event

When using the Facebook extension in your projects, you will be calling different functions that will trigger "callbacks" from the Facebook API. What this means is that certain functions will be run but won't actually give a result until sometime in the future, which could be the next step, or it could be a few seconds later. This result, when it comes, is called the "callback" and is the Facebook API responding to something you've done. This callback is dealt with in the **Asynchronous Social Event**.

This event will always have a DS map in the GML variable `async_load`, and this map can be parsed to get the required information. Each function will generate different callbacks, but they will all have certain key/value pairs in common, which we'll list here:

- **"type"** – This is the event type key and it will hold a string with the type of event that has been triggered. For example, if it's a login event, then the string will be "`facebook_login_result`".

- **"requestId"** – This is the unique ID value of the function request. All functions that generate a callback will return a request ID, which can then be stored in a variable. This request ID value can then be checked in the asynchronous event to make sure that the correct code is being run for the function that triggered it. In general, this is only useful if there is a possibility of multiple calls to the same function, which in turn generates multiple callbacks all of the same "type".

- **"status"** – This can be used to identify whether or not the API call was successful. The value will be one of the following strings:
    - "*success*" – The function call was processed successfully
    - "*error*" – There was an error of some type (eg: network failure, timeout, etc…)
    - "*cancelled*" – The user cancelled the operation before it could be completed (not all function callbacks will have this as a possible status)

The rest of the key/value pairs in the map will depend on the function that triggered the Async Event, and you should check the individual functions listed in the rest of this manual for exact details.

Extension Functions

The rest of this manual is taken up with the different functions and constants that are included as part of the Facebook API Extension. If you are updating from a previous version of GameMaker Studio, then you should see that the new extension functions map easily to the old `facebook_*()` functions.

fb_init

Description

This function should be called to initialise the Facebook API, before using any of the other Facebook functions. This should be done only *once* in the game and can be done at any time, but it is recommended that you call this function right at the game start before anything else. Note too, that even if you use `facebook_logout()` at some point in your game you will not need to use this function again to log in.

> **IMPORTANT!** *This function is called automatically by the extension and you should NOT need to use it unless you have specifically edited the extension to prevent the initialisation function being called.*

Syntax

```
fb_init();
```

Returns

N/A

Example

```
fb_init();
var permissions = ds_list_create();
ds_list_add(permissions,"public_profile", "user_friends");
fb_login(permissions, fb_login_default);
ds_list_destroy(permissions);
```

fb_ready

Description

This function can be called to check and see if the Facebook API has been initialised correctly. It will return `true` if the API has been initialised or `false` otherwise.

Syntax

```
fb_ready();
```

Returns

Bool

Example

```
if !fb_ready()
    {
    fb_init();
    }
```

`fb_status`

Description

With this function you can poll Facebook for the current status of the user login. One of the following strings is returned (note that they are always returned in capital letters):

- "**AUTHORISED**": This means that the user has logged in correctly, but not all permissions may have been granted.
- "**PROCESSING**": This is returned when the login process is currently un-resolved.
- "**IDLE**": This is returned when the connection to the Facebook API has been initialised, but no further action has been taken (ie: no user is logged in).
- "**FAILED**": This means that the user has not logged in correctly, usually due to a connection error.

This function does not continuously check the Facebook API and will only be updated after a Facebook action has been resolved (like logging in, logging out or requesting permissions).

Syntax

```
fb_status();
```

Returns

String

Example

```
switch (fb_status())
    {
    case "AUTHORISED": global.Auth = true; instance_destroy(); break;
    case "FAILED": facebook_login(permissions); alarm[0] = 30; break;
    default: alarm[0] = 30;
    }
```

`fb_login`

Description

With this function GameMaker Studio 2 will connect to Facebook and ask the user to log into their account. You may also supply additional permission requests in the form of a DS list that this function then converts into a JSON array to communicate them to Facebook. There are a wide variety of permissions you may request and they are documented on the Facebook developers pages found here. If you do not wish to ask for any special or extended permissions then you will still need to send a DS list, but this time it will be empty.

> **NOTE**: *You can only use the default read permissions with this function. If you require any write permissions or extended read permissions, then you need to call `fb_request_publish_permissions()` or `fb_request_read_permissions` after logging in. Also note that if your game requests more than the "public_profile", "email" and "user_friends" permissions, it will require review by Facebook before it can be played by people other than the game's developers*.

The function has an extra parameter required for iOS login, which should be one of the constants listed below (other platforms can simply use `fb_login_default` for the argument):

Constant	Description
`FacebookExtension2_LOGIN_TYPE_NATIVE`	This is the default behaviour and indicates logging in through the native Facebook app may be used. The SDK may still use Safari instead.
`FacebookExtension2_LOGIN_TYPE_BROWSER`	Attempts log in through the Safari or SFSafariViewController, if available.
`FacebookExtension2_LOGIN_TYPE_SYSTEM_ACCOUNT`	Attempts log in through the Facebook account currently signed in through the device Settings. If the account is not available this behaviour falls back to `FacebookExtension2_LOGIN_TYPE_NATIVE`.
`FacebookExtension2_LOGIN_TYPE_WEB_VIEW`	Attempts log in through a modal Web View pop up. This behaviour is only available to certain types of apps. Please check the Facebook Platform Policy to verify your app meets the restrictions.

The above listed constants change the login behaviour on iOS, permitting you to select which fallback methods to use (if any) should login fail from through the normal process. For further information please see this Facebook Developer page.

When you call the function, it will return an integer *request ID* value that can be used in the **Social Asynchronous Event** to identify the request that triggered the event. In this Async event, you can then check the built-in DS map `async_load` for the following keys:

Cont.../

`fb_login` **Cont../**

Key	Description	Data Type
"type"	Used for identifying the type of API call that this event was generated by.	String: "facebook_login_result"
"requestId"	Contains the unique request ID returned by the function call that this request was generated by.	Integer
"status"	Used to identify whether or not the API call was successful.	String: "success" "cancelled" "error"
"exception"	Returned if "status" is equal to "error". Provides details on the exception that caused the error.	String
"<permission>" eg: "user_friends"	Each permission requested will be added as a key to this map, with their value being set to either "granted" or "declined", depending on whether the user authorized them or not.	String: "granted" "declined"

Syntax

```
fb_login(permission_list, login_type);
```

Argument	Description	Data Type
`permission_list`	The permissions you want to have from the user logging in.	DS List
`login_type`	The iOS only log in type (see the list of constants below).	Constant

Returns

Integer (request ID)

Example

```
var permissions = ds_list_create();
ds_list_add(permissions,"public_profile", "user_friends");
fb_login(permissions, fb_login_default);
ds_list_destroy(permissions);
```

fb_logout

Description

With this function you can log the user out of Facebook. Note, this does not mean that Facebook is disconnected from the game and there is no need to call fb_init() again if you wish the user to log back in again. It is also worth noting that certain graph and dialogue functions may work even with the user logged out, as they will prompt the user to log in again beforehand.

Syntax

```
fb_logout();
```

Returns

N/A

Example

```
if mouse_check_button_pressed(mb_left)
    {
    fb_logout();
    }
```

fb_user_id

Description

With this function you can poll Facebook for the user id of the currently logged in user. This will be returned as a string, with a format similar to "07778915123" which can then be used with the other Facebook functions.

Syntax

```
fb_user_id();
```

Returns

String

Example

```
if fb_status() == “AUTHORISED”
    {
    global.FB_id = fb_user_id();
    }
```

fb_accesstoken

Description

This function will return the access token that Facebook gives you when you log in to verify that it's a valid log in. An access token is a random string that provides temporary, secure access to the Facebook APIs.

Syntax

```
fb_accesstoken();
```

Returns

String

Example

```
if fb_status() == “AUTHORISED”
    {
    global.FB_token = fb_accesstoken();
    }
```

fb_refresh_accesstoken

Description

This function can be used to refresh the access token for the logged in user. When you call the function, it will return an integer *request ID* value that can be used in the **Social Asynchronous Event** to identify the request that triggered the event. In this Async event, you can then check the built-in DS map `async_load` for the following keys:

Key	Description	Data Type
"type"	Used for identifying the type of API call that this event was generated by.	String: "fb_refresh_accesstoken"
"requestId"	Contains the unique request ID returned by the function call that this request was generated by.	Integer
"status"	Used to identify whether or not the API call was successful.	String: "success" "error"
"exception"	Returned if "status" is equal to "error". Provides details on the exception that caused the error.	String
"access_token"	The new access token for the user	String

Syntax

```
fb_refresh_accesstoken();
```

Returns

Integer (request ID)

Example

```
global.FB_token = fb_accesstoken();

// In the Async Social Event

if async_load[? "requestId"] == global.FB_token
    {
    if async_load[? "status"] == "success"
        {
        global.FB_token = async_load[? "access_token"];
```

```
        }
    }
```

`fb_send_event`

Description

This function will send a Facebook analytics event to the Developer Console. The function takes one of the following constants to define the event ID (type of event):

- `FacebookExtension2_EVENT_ACHIEVED_LEVEL`
- `FacebookExtension2_EVENT_ADDED_PAYMENT_INFO`
- `FacebookExtension2_EVENT_ADDED_TO_CART`
- `FacebookExtension2_EVENT_ADDED_TO_WISHLIST`
- `FacebookExtension2_EVENT_COMPLETED_REGISTRATION`
- `FacebookExtension2_EVENT_COMPLETED_TUTORIAL`
- `FacebookExtension2_EVENT_INITIATED_CHECKOUT`
- `FacebookExtension2_EVENT_RATED`
- `FacebookExtension2_EVENT_SEARCHED`
- `FacebookExtension2_EVENT_SPENT_CREDITS`
- `FacebookExtension2_EVENT_UNLOCKED_ACHIEVEMENT`
- `FacebookExtension2_EVENT_VIEWED_CONTENT`

You then supply a numeric value to be logged for the event as well as an event "object", which is a previously created DS list holding various keys and values. This list should be structured `[key1, value1, key2, value2, ... ]`, where each key is one of the following constants:

- `FacebookExtension2_PARAM_CONTENT_ID`
- `FacebookExtension2_PARAM_CONTENT_TYPE`
- `FacebookExtension2_PARAM_CURRENCY`
- `FacebookExtension2_PARAM_DESCRIPTION`
- `FacebookExtension2_PARAM_LEVEL`
- `FacebookExtension2_PARAM_MAX_RATING_VALUE`
- `FacebookExtension2_PARAM_NUM_ITEMS`
- `FacebookExtension2_PARAM_PAYMENT_INFO_AVAILABLE`
- `FacebookExtension2_PARAM_REGISTRATION_METHOD`
- `FacebookExtension2_PARAM_SEARCH_STRING`
- `FacebookExtension2_PARAM_SUCCESS`

If you do not wish to include an event "object" then simply supply a value of -1 instead of a DS list ID.

You can get more information on the different event types and parameters here.

Cont../

`fb_send_event` **Cont../**

Syntax

```
fb_send_event(event_id, event_value_float, event_value_object);
```

Argument	Description	Data Type
`event_id`	The analytics event ID. see the description above for details of the available constants.	Constant
`event_value_float`	Numeric value to be logged for this event	Real
`event_value_object`	A list of key/value pairs for logging more advanced event parameter data. List format: [key1, value1, key2, value2, ...]. See the description above the possible key constants.	Integer (DS list ID)

Returns

N/A

Example

```
if global.Level == 100
    {
    fb_send_event(FacebookExtension2_EVENT_ACHIEVED_LEVEL, 100, -1);
    }
```

fb_check_permission

Description

With this function you can check the Facebook API to see if the user has granted a specific permission, either read or publish. The function will return `true` if they have the permission or `false` if they do not or there is an error. Note that the user must be logged in for the check to function. See here for a list of available Facebook permissions.

Syntax

```
fb_check_permission(permission);
```

Argument	Description	Data Type
permission	The permission to check	String

Returns

Real

Example

```
var permissions = ds_list_create();
if !facebook_check_permission("user_likes")
   {
   ds_list_add(permissions, "user_likes");
   }
if !facebook_check_permission("user_interests")
   {
   ds_list_add(permissions, "user_interests");
   }
if !ds_list_empty(permissions)
   {
   request_ID = facebook_request_read_permissions(permissions);
   }
ds_list_destroy(permissions);
```

`fb_request_read_permissions`

Description

With this function you can request additional read permissions from the Facebook API. These permissions are added (as strings) to a previously created `ds_list` which is then used in the function to send the request. The user must be successfully logged in (ie. they have called `fb_login()`) and the `fb_status()` should be "AUTHORISED" before you call this function.

When you call the function, it will return an integer *request ID* value that can be used in the **Social Asynchronous Event** to identify the request that triggered the event. In this Async event, you can then check the built-in DS map `async_load` for the following keys:

Key	Description	Data Type
"type"	Used for identifying the type of API call that this event was generated by.	String: "fb_refresh_accesstoken"
"requestId"	Contains the unique request ID returned by the function call that this request was generated by.	Integer
"status"	Used to identify whether or not the API call was successful.	String: "success" "error"
"exception"	Returned if "status" is equal to "error". Provides details on the exception that caused the error.	String
"<permission>"	A permission key. Each permission requested will be added as a separate key.	String: "granted" "declined"

If the function returns -1 then it means that there is an issue with making the request (for example, the user has not logged in).

See here for a list of available Facebook permissions

Cont.../

`fb_request_read_permissions` **Cont.../**

Syntax

```
fb_request_read_permissions(permission_list);
```

Argument	Description	Data Type
`permission_list`	A DS list with the permissions to request	String

Returns

Integer (request ID)

Example

```
var permissions = ds_list_create();
if !facebook_check_permission("user_likes")
   {
   ds_list_add(permissions, "user_likes");
   }
if !facebook_check_permission("user_interests")
   {
   ds_list_add(permissions, "user_interests");
   }
if !ds_list_empty(permissions)
   {
   request_ID = facebook_request_read_permissions(permissions);
   }
ds_list_destroy(permissions);
```

`fb_request_publish_permissions`

Description

With this function you can request additional publish permissions from the Facebook API. These permissions are added (as strings) to a previously created `ds_list` which is then used in the function to send the request. The user must be successfully logged in (ie. they have called `fb_login()`) and the `fb_status()` should be "AUTHORISED" before you call this function.

When you call the function, it will return an integer *request ID* value that can be used in the **Social Asynchronous Event** to identify the request that triggered the event. In this Async event, you can then check the built-in DS map `async_load` for the following keys:

Key	Description	Data Type
"type"	Used for identifying the type of API call that this event was generated by.	String: "fb_refresh_accesstoken"
"requestId"	Contains the unique request ID returned by the function call that this request was generated by.	Integer
"status"	Used to identify whether or not the API call was successful.	String: "success" "error"
"exception"	Returned if "status" is equal to "error". Provides details on the exception that caused the error.	String
"<permission>"	A permission key. Each permission requested will be added as a separate key.	String: "granted" "declined"

If the function returns -1 then it means that there is an issue with making the request (for example, the user has not logged in).

See here for a list of available Facebook permissions

Cont.../

`fb_request_publish_permissions` **Cont.../**

Syntax

```
fb_request_publish_permissions(permission_list);
```

Argument	Description	Data Type
`permission_list`	A DS list with the permissions to request	String

Returns

Integer (request ID)

Example

```
var permissions = ds_list_create();
if !facebook_check_permission("publish_to_groups")
   {
   ds_list_add(permissions, " publish_to_groups");
   }
if !ds_list_empty(permissions)
   {
   request_ID = facebook_request_publish_permissions(permissions);
   }
ds_list_destroy(permissions);
```

`fb_graph_request`

Description

With this function you can get the user to interact with the Facebook Social Graph. The "graph_path" argument is where you define the part of the graph you wish to access, like the current user friends list, or comments on another app, or even an event. The exact path can be defined using the terms outlined in the Publishing section of the Facebook Graph API pages. You then define the http method to use which is usually "POST" or "GET" but Facebook also accepts the "DELETE" method.

The next argument is slightly more complex as it requires you to have created and filled a DS list with the correct information which GameMaker Studio 2 will then convert into json automatically to be sent to the Facebook API. The information that you put in this list will depend very much on which path you choose to use and a complete list of all possible values can be found here. Note that if the graph path requires no extra parameters, then you should supply a value of -1 instead of a DS list ID. The DS list should be formatted with consecutive key value pairs, eg:

```
var _l = ds_list_create("key", "value", "key", "value", etc..);
```

When you call the function, it will return an integer *request ID* value that can be used in the **Social Asynchronous Event** to identify the request that triggered the event. In this Async event, you can then check the built-in DS map `async_load` for the following keys:

Key	Description	Data Type
"type"	Used for identifying the type of API call that this event was generated by.	String: "fb_graph_request"
"requestId"	Contains the unique request ID returned by the function call that this request was generated by.	Integer
"status"	Used to identify whether or not the API call was successful.	String: "success" "error"
"exception"	Returned if "status" is equal to "error". Provides details on the exception that caused the error.	String
"response_text"	The response of this graph API request, encoded into a JSON string.	String (JSON)

If the function returns -1 then it means that there is an issue with making the request (for example, the user has not logged in).

Cont.../

`fb_graph_request` **Cont.../**

Syntax

```
fb_graph_request(graph_path, http_method, ds_list_params);
```

Argument	Description	Data Type
graph_path	The graph API path for the request	string
http_method	The HTTP method to be used for the API request.	
ds_list_params	A DS List containing parameters for the API request. Use -1 if there are no additional parameters to be added	Integer (DS list ID)

Returns

Real

Example

```
mRequestId = fb_graph_request(string("me/permissions"), "DELETE", -1);
fb_logout();
```

fb_dialog

Description

With this function you can open a Facebook dialog with a specific link. The dialog will then be shown to the user and they can edit the contents, add text, etc… The initial contents of the dialog will depend on the link given to the function, as the Facebook API will try and pull a title, an icon image and a descriptive text from the given URL to populate the dialog. You can optimise this process following the Facebook Open Graph Markup Guide.

When you call the function, it will return an integer *request ID* value that can be used in the **Social Asynchronous Event** to identify the request that triggered the event. In this Async event, you can then check the built-in DS map `async_load` for the following keys:

Key	Description	Data Type
"type"	Used for identifying the type of API call that this event was generated by.	String: "fb_graph_request"
"requestId"	Contains the unique request ID returned by the function call that this request was generated by.	Integer
"status"	Used to identify whether or not the API call was successful.	String: "success" "cancelled" "error"
"exception"	Returned if "status" is equal to "error". Provides details on the exception that caused the error.	String

If the function returns -1 then it means that there is an issue with making the request (for example, the user has not logged in).

Syntax

```
fb_dialog(link_url);
```

Argument	Description	Data Type
link_url	URL to the post to be shared	String

Cont.../

fb_dialog **Cont.../**

Returns

Real

Example

```
if fb_status() == “AUTHORISED”
    {
    if mouse_check_button_pressed(mb_left)
        {
        fb_dialog("https://developers.facebook.com/docs/ios/share/");
        }
    }
```