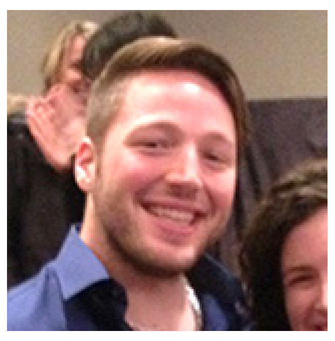

Sammy Kaye Powers

- About
- Speaking
- Contact

Upgrade Guide: Facebook PHP SDK v4.0 to v5

Upgrading the Facebook PHP SDK from v4.0 to v5

Jul 15, 2015

On **July 9th, 2015**, Facebook tagged the latest stable version of the Facebook PHP SDK, version 5.

This guide will help you upgrade from version 4.0 to version v5 of the official Facebook PHP SDK.

> **Wait - what happened to version 4.1?** Before the decision to adopt Semantic Versioning, the official Facebook PHP SDK was going to be released as v4.1. But now v4.1 is just an alias to v5.0.0. So any references to 4.1 can be safely aliased to v5.0.0.

Now with better docs!

Each major version of the PHP SDK at Facebook have notoriously been poorly documented. With version 5, the official docs have been much improved... or at least I think they have since I was the primary author. But I know they can still be improved even more! You can contribute to them on the official GitHub repo.

You need PHP 5.4 or greater

Version 5 of the SDK runs on PHP 5.4 or greater with the mbstring extension installed. If you're still on PHP 5.3, you really need to upgrade since 5.3 reached EOL and is no longer getting security updates. While you're at it, why not just install PHP 5.6!

Installation with Composer

The recommended way to install the Facebook PHP SDK v5 is with Composer. Just add the following to your `composer.json` and run `composer update` from the command-line.

```
{
  "require" : {
    "facebook/php-sdk-v4": "^5.0"
  }
}
```

No, that's not a type-o. The package is named `php-sdk-v4`... yes a `-v4` suffix to install v5... that was an oversight that will eventually get addressed.

> **A note on SemVer:** The Facebook PHP SDK v5 follows Semantic Versioning (SemVer) so it's safe to use the carrot (^) in the composer require line.

Don't forget to include the Composer autoloader in your script if you haven't already done that.

Manual Installation

You don't use Composer? You should definitely start using it. But if you're needing a quick-and-dirty install, follow the official manual installation guide.

Introduction & Comparison

Lots has changed in v5.0. Before you start upgrading all the things it might be best to familiarize yourself with the new API in the v5 getting started guide.

Configuration

There are a number of configuration options that the PHP SDK needs to know about in order to run properly. The three most important are:

1. App ID
2. App secret
3. Graph API version

App ID & secret

Version 4.0 of the PHP SDK, focused a lot of attention on the `FacebookSession` class. Contrary to what you might assume at first glance, this class has nothing to do with a native PHP session and everything to do with a user access token and the app configuration.

```
# The v4.0 way to configure your app settings
use Facebook\FacebookSession;

// Eeeew. Statics.
FacebookSession::setDefaultApplication('{app-id}', '{app-secret}');
```

In version 5 you inject your app settings via an array in the `Facebook\Facebook` service class which ties all the components of the SDK together in a nice little API.

```
# The v5 new-hotness way
$fb = new Facebook\Facebook([
  'app_id'     => '{app-id}',
  'app_secret' => '{app-secret}',
  'default_graph_version' => 'v2.4',
  ]);
```

The default Graph API version

In v4.0 of the PHP SDK, there were a number of methods that allowed you to define the Graph API version. Unfortunately this functionality was inconsistent across all the methods and forced developers to rely on the default Graph API version fallback in some cases.

Since the Graph API is subject to a versioning schedule with breaking changes in each version, every time the default Graph API version was updated, apps broke when the developer updated their version of the SDK with the latest minor/patch version. Doh!

In version 5, you can set the default Graph API version when you instantiate the main `Facebook\Facebook` service with the `default_graph_version` option.

```
$fb = new Facebook\Facebook([
  // ...
  'default_graph_version' => 'v2.4',
  ]);
```

> **Important!** You should always set the `default_graph_version` option! Don't rely on the fallback version unless you want your app to break when the version of Graph API gets updated in the SDK.
>
> At the time of writing, v2.4 is the latest version of Graph. You should make sure you're using the latest version.

Helpers have moved

The "helpers" in v4.0 allowed you to obtain an access token or signed request from a number of contexts. The helpers still exist in v5, but they have moved to the `Facebook\Helpers` namespace.

On top of that, two of the helpers got renamed.

Helper name in v4.0	Helper name in v5
`FacebookJavaScriptLoginHelper`	`FacebookJavaScriptHelper`
`FacebookCanvasLoginHelper`	`FacebookCanvasHelper`

Facebook Login (getting an access token)

Access tokens could be obtained using the "helpers" in v4.0. You would instantiate the helpers directly in v4.0 but in v5 we have factories on the `Facebook\Facebook` service class to generate new helpers as we will see in a bit.

Getting an access token from redirect

You can implement Facebook Login (OAuth 2.0) with the PHP SDK by using the `FacebookRedirectLoginHelper`. An instance of this helper can be obtained from the `Facebook` super service by using the `getRedirectLoginHelper()` method.

Compare how you generate an authentication URL in v4.0 vs v5:

```
# v4.0
Facebook\FacebookSession::setDefaultApplication('{app-id}', '{app-secret}');

$permissions = ['email', 'user_posts']; // optional
$callback = 'http://example.com/fb-login-callback.php';
$helper = new Facebook\FacebookRedirectLoginHelper($callback);
$loginUrl = $helper->getLoginUrl($permissions);

echo '<a href="'.$loginUrl.'">Log in with Facebook!</a>';

# v5
$fb = new Facebook\Facebook([/* . . . */]);
$helper = $fb->getRedirectLoginHelper();

$permissions = ['email', 'user_posts']; // optional
$callback = 'http://example.com/fb-login-callback.php';
$loginUrl = $helper->getLoginUrl($callback, $permissions);

echo '<a href="'.$loginUrl.'">Log in with Facebook!</a>';
```

Then in your callback script, `fb-login-callback.php`, you grab an instance of the `FacebookRedirectLoginHelper` again and call the `getAccessToken()` method.

Compare how you handle the callback URL in v4.0 vs v5:

```
# v4.0 (fb-login-callback.php)
Facebook\FacebookSession::setDefaultApplication('{app-id}', '{app-secret}');
$helper = new Facebook\FacebookRedirectLoginHelper($redirect_url);

try {
    $session = $helper->getSessionFromRedirect();
} catch(Facebook\FacebookSDKException $e) {
    // There was an error communicating with Graph
    echo $e->getMessage();
    exit;
}

if ($session) {
    // User authenticated your app!
    // Save the access token to a session and redirect
    $_SESSION['facebook_access_token'] = $session->getToken();
    // Log them into your web framework here . . .
    // Redirect here . . .
    exit;
} else {
    // User denied or some other problem
    // You have to parse the GET params
    // to figure out what went wrong
}
```

Version 5 gives us a bit more bullet-proofing in the callback:

```
# v5 (fb-login-callback.php)
$fb = new Facebook\Facebook([/* . . . */]);
$helper = $fb->getRedirectLoginHelper();

try {
  $accessToken = $helper->getAccessToken();
} catch(Facebook\Exceptions\FacebookSDKException $e) {
  // There was an error communicating with Graph
  echo $e->getMessage();
  exit;
}

if (isset($accessToken)) {
  // User authenticated your app!
  // Save the access token to a session and redirect
  $_SESSION['facebook_access_token'] = (string) $accessToken;
  // Log them into your web framework here . . .
  // Redirect here . . .
  exit;
} elseif ($helper->getError()) {
  // The user denied the request
  // You could log this data . . .
  var_dump($helper->getError());
  var_dump($helper->getErrorCode());
  var_dump($helper->getErrorReason());
  var_dump($helper->getErrorDescription());
  // You could display a message to the user
  // being all like, "What? You don't like me?"
  exit;
}

// If they've gotten this far, they shouldn't be here
http_response_code(400);
exit;
```

Getting Access Tokens From App Canvas/Page Tabs

When you have an app that runs in app canvas or a Page tab, an access token can be obtained from the signed request that is `POST`ed to your app. If an access token exists in the signed request, it can be obtained using the `FacebookCanvasHelper` *(note the name change)*.

```
# v4.0
Facebook\FacebookSession::setDefaultApplication('{app-id}', '{app-secret}');
$helper = new Facebook\FacebookCanvasLoginHelper();

try {
  $session = $helper->getSession();
} catch(Facebook\FacebookSDKException $e) {
  // There was an error communicating with Graph
  // Or there was a problem validating the signed request
  echo $e->getMessage();
  exit;
}

if ($session) {
  // Logged in
  $_SESSION['facebook_access_token'] = $session->getToken();
}
```

In version 5, an instance of the `FacebookCanvasHelper` can be generated from the `Facebook` service by using the `getCanvasHelper()` method.

```
# v5
$fb = new Facebook\Facebook([/* . . . */]);
$helper = $fb->getCanvasHelper();

try {
  $accessToken = $helper->getAccessToken();
} catch(Facebook\Exceptions\FacebookSDKException $e) {
  // There was an error communicating with Graph
  // Or there was a problem validating the signed request
  echo $e->getMessage();
  exit;
}

if ($accessToken) {
  // Logged in.
  $_SESSION['facebook_access_token'] = (string) $accessToken;
}
```

For Page tabs you use the `FacebookPageTabHelper` that helps you interface with the components available to your Page tab.

```
# v4.0
Facebook\FacebookSession::setDefaultApplication('{app-id}', '{app-secret}');
$helper = new Facebook\FacebookPageTabHelper();

// Obtain a FacebookSession the same way
// as the FacebookCanvasLoginHelper example above

# v5
$fb = new Facebook\Facebook([/* . . . */]);
$helper = $fb->getPageTabHelper();

// Obtain an access token the same way
// as the FacebookCanvasHelper example above
```

Getting Access Tokens From The JavaScript SDK

If you're using the JavaScript SDK for Facebook Login, make sure in your `FB.init()` method, you enable the cookie option with `{cookie:true}`. This tells the JavaScript SDK to set a cookie on your domain that contains a signed request with information about the authenticated user.

```
FB.init({
  appId   : '{app-id}',
  cookie  : true,
  version : 'v2.4'
});
```

In version v4.0 you could obtain a `FacebookSession` from the JavaScript SDK cookie via the `FacebookJavaScriptLoginHelper`.

```
# v4.0
Facebook\FacebookSession::setDefaultApplication('{app-id}', '{app-secret}');
$helper = new Facebook\FacebookJavaScriptLoginHelper();

try {
  $session = $helper->getSession();
} catch(Facebook\FacebookSDKException $e) {
  // There was an error communicating with Graph
  // Or there was a problem validating the signed request
  echo $e->getMessage();
  exit;
}

if ($session) {
  // Logged in
  $_SESSION['facebook_access_token'] = $session->getToken();
}
```

In version v5 an instance of the `FacebookJavaScriptHelper` *(note the name change)* can be generated from the `Facebook` service by using the `getJavaScriptHelper()` method.

```
# v5
$fb = new Facebook\Facebook([/* . . . */]);
$helper = $fb->getJavaScriptHelper();

try {
  $accessToken = $helper->getAccessToken();
} catch(Facebook\Exceptions\FacebookSDKException $e) {
  // There was an error communicating with Graph
  // Or there was a problem validating the signed request
  echo $e->getMessage();
  exit;
}

if ($accessToken) {
  // Logged in.
  $_SESSION['facebook_access_token'] = (string) $accessToken;
}
```

The AccessToken entity

Version 4.0 stored the access token in the `FacebookSession` object (which had nothing to do with PHP sessions). The primary way of sending authenticated requests to the Graph API was by injecting an instance of `FacebookSession` into a `FacebookRequest`. The access tokens were stored as `AccessToken` entities within `FacebookSession`.

In v5, we interface with `AccessToken` entities directly and `FacebookSession` has been removed completely.

Getting a long-lived access token

In v4.0, you could exchange a short-lived access token for a long-lived access token using the `getLongLivedSession()` method on the `FacebookSession` class.

```
# v4.0 example
// Returns the default short-lived access token in a FacebookSession
$session = $helper->getSession();

try {
  // Returns a long-lived access token
  $longSession = $session->getLongLivedSession();
} catch(Facebook\Exceptions\FacebookSDKException $e) {
  // There was an error communicating with Graph
  echo $e->getMessage();
  exit;
}

if (isset($longSession)) {
  $_SESSION['facebook_access_token'] = $longSession->getToken();
}
```

In v5 you can get a long-lived access token by using an instance of the `OAuth2Client`. You can generate an instance of the `OAuth2Client` by using the `getOAuth2Client()` method on the `Facebook` service.

```
# v5
$client = $fb->getOAuth2Client();

try {
  // Returns a long-lived access token
  $accessToken = $client->getLongLivedAccessToken('{short-lived-token}');
} catch(Facebook\Exceptions\FacebookSDKException $e) {
  // There was an error communicating with Graph
  echo $e->getMessage();
  exit;
}

if (isset($accessToken)) {
  // Logged in.
  $_SESSION['facebook_access_token'] = (string) $accessToken;
}
```

Requests

Version 5 of the SDK got a major API overhaul when it comes to making requests to the Graph API. The primary way of making requests to the Graph API in v4.0 was via the `FacebookRequest` class directly.

The Graph API supports `GET`, `POST` and `DELETE` HTTP verbs. Here's an example of each HTTP request in v4.0:

```
$session = new Facebook\FacebookSession('{access-token}');

# GET request in v4.0
$request = new Facebook\FacebookRequest($session, 'GET', '/me');
$response = $request->execute();

# POST request in v4.0
$request = new Facebook\FacebookRequest($session, 'POST', '/me/feed', $data);
$response = $request->execute();
```

```
# DELETE request in v4.0
$request = new Facebook\FacebookRequest($session, 'DELETE', '/123');
$response = $request->execute();
```

In v5, each of these HTTP verbs get their own corresponding method on the `Facebook\Facebook` service class.

```
$fb = new Facebook\Facebook([/* . . . */]);

# GET request in v5
$response = $fb->get('/me', '{access-token}');

# POST request in v5
$response = $fb->post('/me/feed', $data, '{access-token}');

# DELETE request in v5
$response = $fb->delete('/123', $data, '{access-token}');
```

If you don't want to have to send in `'{access-token}'` with every method call in v5, you can use the `setDefaultAccessToken()` method.

```
# v5 with default access token fallback
$fb = new Facebook\Facebook([/* . . . */]);

$fb->setDefaultAccessToken('{access-token}');

# These will fall back to the default access token
$response = $fb->get('/me');
$response = $fb->post('/me/feed', $data);
$response = $fb->delete('/123', $data);
```

Responses

Responses from the `execute()` method in v4.0 would return a `FacebookResponse`. The same is true for v5.

```
# v4.0 response example
$session = new Facebook\FacebookSession('{access-token}');
$request = new Facebook\FacebookRequest($session, 'GET', '/me');
$response = $request->execute();

// Get the response as an array
var_dump($response->getResponse());

// array(10) { ...
```

Instead of `getResponse()`, you call the `getDecodedBody()` method on the `FacebookResponse` entity to get an array in v5.

```
# v5 response example
$fb = new Facebook\Facebook([/* . . . */]);

$response = $fb->get('/me');

var_dump($response->getDecodedBody());

// array(10) { ...
```

GraphObjects are GraphNodes

If you're getting your data back in the form of `GraphObject`'s, in v5 they have been renamed to `GraphNode`'s.

```
# v4.0 GraphObject example
$response = $request->execute();

$object = $response->getGraphObject();

var_dump($object->getProperty('id'));
// string(3) "123"

# v5 GraphNode example
$response = $fb->get('/me');

$node = $response->getGraphNode();

var_dump($node->getField('id'));
// string(3) "123"
```

GraphNode subtypes

The `GraphObject` subtypes now extend from `GraphNode` and are easier to obtain from a `FacebookResponse`.

```
# v4.0 GraphObject subtype example
$response = $request->execute();

$userObject = $response->getGraphObject(Facebook\GraphUser::className());

var_dump($userObject->getId());
// string(3) "123"

# v5 GraphNode subtype example
$response = $fb->get('/me');

$userNode = $response->getGraphUser();

var_dump($userNode->getId());
// string(3) "123"
```

There are a number of new `GraphNode` subtypes in v5.

Exceptions have moved

All the exceptions have moved under the `Facebook\Exceptions` namespace.

```
# v4.0 exception example
try {
  // Do something
} catch (Facebook\FacebookSDKException $e) {
  die($e->getMessage());
}

# v5 exception example
try {
  // Do something
} catch (Facebook\Exceptions\FacebookSDKException $e) {
  die($e->getMessage());
}
```

File Uploads

The file upload support in v4.0 was very pretty sketch and would only work properly if you had the cURL extension installed.

```
# v4.0 photo upload example
$session = new Facebook\FacebookSession('{access-token}');

$data = [
  'source' => new CURLFile('path/to/file.png', 'image/png'),
  'message' => 'My file!',
  ];

$request = new Facebook\FacebookRequest($session, 'POST', '/me/photos', $data);
$response = $request->execute();
```

In v5, uploading files is much simpler and doens't require using `CURLFile`.

```
# v5 photo upload example
$fb = new Facebook\Facebook([/* . . . */]);
```

```
$data = [
  'source' => $fb->fileToUpload('/path/to/photo.jpg'),
  'message' => 'My file!',
  ];

$response = $fb->post('/me/photos', $data, '{access-token}');
```

Not only can you upload photos, but you can also upload videos. And what's more, file uploads are supported in batch requests.

PHP sessions (persistent data) & the FacebookRedirectLoginHelper

One of the biggest pitfalls people would run into when using v4.0 for the first time surrounded the `FacebookRedirectLoginHelper`. The helper would try to make use of native PHP sessions behind the scenes to store the CSRF token generated during the Facebook Login process. Since the PHP session stuff wasn't documented, people got errors and didn't know why.

If you didn't use native PHP sessions to store your persistent data, you needed to extend the `FacebookRedirectLoginHelper` and overwrite the `storeState()` and `loadState()` methods.

Version 5 of the SDK will still try to store a CSRF token in native PHP session, but there's a better way to overwrite the session-storing functionality; you code to the `Facebook\PersistentData\PersistentDataInterface` and inject the implementation into the `Facebook\Facebook` service constructor with the `persistent_data_handler` option.

See an example of how to overwrite the persistent data handler with CodeIgniter or with Laravel.

Custom HTTP clients

Since HTTP client implementations (mainly cURL) can be pretty hard to get right across myriad server environments, the PHP SDK v4.0 allowed you to inject your own HTTP implementation.

In version 5 you can also inject a custom HTTP client, but you'll need to code your implementation to the `Facebook\HttpClients\FacebookHttpClientInterface` and then inject it with the `http_client_handler` option in the `Facebook\Facebook` service constructor.

> ~~**@TODO:** Write a blog about injecting a custom HTTP client in v5.~~ Done!

New Features in v5

Version 5 comes with lots of stability improvements, better security and loads of new features including:

- Batch requests support
- Easy-as-pie pagination
- Injectable URL-detection logic
- Injectable CSPRNG

Final Thoughts

The Facebook PHP SDK has come a very long way since its inception but there's still a ton of room for improvement. There's plenty of discussion going on about what can be improved in version 6.0. And the project is alive and well!

So if you've been looking for an open source project to get involved with, certainly join us in contributing to the Facebook PHP SDK. The water's fine! :)

Good luck!

If you found this guide helpful, say, "Hi" on twitter! I'd love to hear from you. :)

Share

1. Twitter
2. Facebook

Need more help?

I'm available for one-on-one help!

---

1. Home
2. About
3. Speaking
4. Contact