PHP-FCM Documentation

Release 1.2.0

Edwin Hoksberg

Jan 19, 2021

Contents

PHP-FCM is a PHP HTTP client that makes it easy to send push notifications, manage groups of devices and message topics using Google Firebase Cloud Messaging.

```php
<?php

// Load composer
require 'vendor/autoload.php';

// Instantiate the client with the project api_token and sender_id.
$client = new \Fcm\FcmClient($apiToken, $senderId);

// Instantiate the push notification request object.
$notification = new \Fcm\Push\Notification();

// Enhance the notification object with our custom options.
$notification
    ->addRecipient($deviceId)
    ->setTitle('Hello from php-fcm!')
    ->setBody('Notification body')
    ->setColor('#20F037')
    ->setSound("default")
    ->setIcon("myIcon.png")
    ->addData('key', 'value');

// custom sound and custom icon must be in app package
//     - custom sound file must be in /res/raw/
//     - custom icon file must be in drawable resource, if not set, FCM displays␣
↪launcher icon in app manifest

// Send the notification to the Firebase servers for further handling.
$client->send($notification);
```

Before installing, read the *Overview* for more information about this project. Read the *Quickstart* for installation and using the project.

CHAPTER 1

Overview

1.1 Requirements

1. PHP 7.0
2. A free Firebase account, read the *Quickstart* for more information.

1.2 Running the tests

This project uses PHPUnit for running its unit tests. Run the tests using the vendor provided phpunit binary:

```
vendor/bin/phpunit -c phpunit.dist.xml
```

1.3 License

Licensed using the MIT license.

```
Copyright (c) 2018 Edwin Hoksberg <https://github.com/Edwinhoksberg>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
```

(continues on next page)

(continued from previous page)

```
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER 2

Quickstart

2.1 Installing this package

This package is available on packagist, and can be installed with composer:

```
composer require edwinhoksberg/php-fcm
```

2.2 Configuring a Firebase account

This project uses Google Firebase for sending notifications, so we need to signup for an free account. A free account can store up to ~20 million messages, device groups and topics.

1. Click here to go the firebase console page.
2. Click on the Create Project button. Enter a project name and select your country/region.
3. Once your project is created, click on continue to go to the project dashboard.
4. Click on the gear icon in the top left, next to the Project Overview link, and click on Project Settings.
5. Go to the Cloud Messaging tab.
6. On this page, you should see several authentication tokens. Copy and save the *Server key* and *Sender ID* tokens somewhere. These will be used in the project configuration.

2.3 Configuring the project

When you instantiate a new Firebase Cloud Messaging object, pass the *Server Key* and *Sender ID* you retrieved from the previous steps:

```php
<?php

// Load composer
require 'vendor/autoload.php';

$client = new \Fcm\FcmClient(/* Server Key */, /* Sender ID */);
```

CHAPTER 3

App integration

3.1 Android Integration

The easiest way to start is using the offcial example messaging app which can be found here. It is also documented on how to register a device and how to capture the device id, which will be used for sending notifications or managing device groups/topics.

Official documentation: https://firebase.google.com/docs/cloud-messaging/android/client.

3.2 iOS Integration

An example application was also made available for iOS, which can be found on the official github repository.

Official documentation: https://firebase.google.com/docs/cloud-messaging/ios/client.

CHAPTER 4

Device Information

There is only 1 device call Firebase exposes, and it can be used to retrieve information about a single device, such as registration date and subscribed topics.

4.1 Device Info

```php
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

// Remove the second parameter for more basic device information
$info = new \Fcm\Device\Info($deviceId, true);

// Shortcut function:
// $info = $client->deviceInfo($deviceId, true);

$client->send($info);
```

Example response:

```
array(10) {
  'applicationVersion' =>
  string(1) "1"
  'connectDate' =>
  string(10) "2018-08-07"
  'attestStatus' =>
  string(6) "ROOTED"
  'application' =>
  string(27) "com.google.application"
  'scope' =>
  string(1) "*"
  'authorizedEntity' =>
  string(12) "347372151029"
```

(continues on next page)

(continued from previous page)

```
  'rel' =>
  array(1) {
    'topics' =>
    array(1) {
      'news' =>
      array(1) {
        'addDate' =>
        string(10) "2018-08-07"
      }
    }
  }
  'connectionType' =>
  string(4) "WIFI"
  'appSigner' =>
  string(40) "c5abd4420a7b4844c034fe9c47fcb42234bbf5fe"
  'platform' =>
  string(7) "ANDROID"
}
```

4.2 Quirks

This plugin uses the underlying package *guzzlehttp* - this package is designed to error out if a URL/Link returns a *404 Not Found*. However, this can cause the above *DeviceInfo* to error out at as well. When looking up a deviceID via FCM, if the deviceID does not exist the Google FCM API returns a *404 Not Found* with the JSON value *{"error":"No information found about this instance id."}*. The 404 causes *guzzlehttp* to throw a Fatal Error and exit the script. Obviously, this is a major issue if you are scripting mass deviceID lookups - the first not found ID will exit your script.

To get around this, you can use the *http_errors* option, see *Package Options*.

Additionally, once the above modification is applied, you will need your processing script to check for JSON key *error* to process the proper error message. IE:

```
$info = new \Fcm\Device\Info($deviceID, true);
$response =  $client->send($info);
if (array_key_exists('error',$response)) {
    // process error info here
} else if (array_key_exists('rel',$response)) {
    // process returned info here
} else {
    // ID exists but is not registered to any topics/groups/etc
}
```

CHAPTER 5

Sending Messages

There are two different types of messages Firebase can support, *Notification messages* and *Data messages*. Read here for more information.

5.1 Notification message to deviceIDs <to> <registration_ids> <deviceGroupID>

FCM automatically displays the message to end-user devices on behalf of the client app. Notification messages have a predefined set of user-visible keys and an optional data payload of custom key-value pairs.

```
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

$notification = new \Fcm\Push\Notification();
$notification
    ->addRecipient($deviceId1)
    ->addRecipient($deviceGroupID)
    ->addRecipient($arrayIDs)
    ->setTitle('Hello from php-fcm!')
    ->setColor('#20F037')
    ->setSound("default")
    ->setBadge(11)
    ->addData("key","value");

// Shortcut function:
// $notification = $client->pushNotification('The title', 'The body', $deviceId);

$response = $client->send($notification);
```

Example deviceID <to> <registration_ids> <deviceGroupID> response:

```
array(5) {
  'multicast_id' =>
  int(9014353506250345342)
  'success' =>
  int(1)   // how many deviceIDs were successfully sent messages
  'failure' =>
  int(1)   // how many deviceIDs messages failed to send to
  'canonical_ids' =>
  int(0)
  'results' =>
  array(2) {
    [0] =>  // each deviceID message success or failure
    array(1) {
      'message_id' =>  //first ID success
      string(35) "0:154231004164960%c5f39c08c5f39c543"
    }
    [1] =>  // each deviceID message success or failure
    array(1) {
      'error' =>  // second ID failure
      string(19) "InvalidRegistration"
    }
  }
}
```

5.2 Notification message <topics>

FCM automatically displays the message to end-user devices on behalf of the client app. Notification messages have a predefined set of user-visible keys and an optional data payload of custom key-value pairs.

```php
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

$notification = new \Fcm\Push\Notification();
$notification
    ->addRecipient('/topics/myTopicName')
    ->setTitle('Hello myTopicName Members')
    ->setColor('#20F037')
    ->setSound("default")
    ->setIcon("myIcon.png")
    ->addDataArray($myObjArray);

// Shortcut function:
// $notification = $client->pushNotification('The title', 'The body', $deviceId);

$response = $client->send($notification);
```

Example <topics> response:

```
array(1) {
    'message_id' =>  // this is a successful response to a topic notification
    int(154231004164960%c5f39c08c5f39c543)
  }

array(1) {
```

(continues on next page)

(continued from previous page)

```
    'error' =>  // this is an error response to a topic notification
    string(19) "InvalidRegistration"
  }
```

5.3 Notification sending options

```
* addRecipient
     // recipient can be ONE of four types
     // deviceID (string)
     // devicegroupID (string)
     // registeredIDs (array_of_IDs)
     // topicID ('/topics/myTopicID')

     // note: deviceID/deviceGroupID/registerIDs can be mixed/matched in same␣
↪notification
     // note: topicID can not be mixed/matched with other IDs types in same␣
↪notification
```

5.4 Notification options <topics> <deviceID> <registered_ids>

iOS, Android currently Supported options for notifications

```
 iOS only:
 * setBbadge (int)
 * setSubtitle (string)

 Android only:
 * setTag (string)
 * setColor (string (hex #rrggbb color format))
     // In Android 6 and lower - this the background color of the icon image when you␣
↪pull down on the status bar messages
     // In Android 7 and greater - this is the color of the icon itself when you pull␣
↪down on the status bar messages
 * setIcon (string)
     // custom icon file must be in app itself
     // icon must be drawable resource, if not set, FCM displays launcher icon in app␣
↪manifest
     // for consistency across Android OS versions(5.0 - 10.0), use a material design,
↪ transparent icon
     // if using icon in `drawable-XYdi` folders, use icon name without file␣
↪extension, ie: ->setIcon('myIcon')
     // if using icon from other location, you must specificy the file extension, ie:␣
↪->setIcon('www/images/thisIcon.png')
     // for more info, see: https://github.com/arnesson/cordova-plugin-firebase/
↪issues/764

*** future: android_channel_id

 Both:
 * setTitle (string)
 * setBody (string)
```

(continues on next page)

(continued from previous page)

```
 * setSound (string)
     // custom sound must be in the app itself
     // custom sound file must be in /res/raw/

 *** future: click_action
 *** future: body_loc_key
 *** future: body_loc_args
 *** future: title_loc_key
 *** future: title_loc_args

 //NOTE: You can mix and match iOS and Android only options in same notification
 //NOTE: if iOS/Android don't recognize an option used in the other platform, that␣
↪option is simply discarded/ignored.
```

5.5 Notification DATA options <topics> <deviceID> <registered_ids>

Client app is responsible for processing data messages. Data messages require custom key-value pairs that your app will understand.

```
* addData("key","value") - add data key/values one at a time
* addDataArray(array_of_keyValues) - add data as a prebuilt object array
    //     $fcmData = array(
    //          'action' => 2,
    //          'dataTitle' => "This is my subject line",
    //          'dataMsg' => "This is the body of my message

    // Example, In a cordova based app using `cordova-plugin-firebase`
    window.FirebasePlugin.onNotificationOpen(function(payload) {
        // if there is a payload it will be in payload object
        if (payload.action == 1) { // email verification confirmation
          setDB("user_emailVerify",payload.user_emailVerify) ;
          alert("Thank you, your email address has now been verified") ;
        } else if (payload.action == 2) {  // display gen message
          alert(payload.dataTitle+ "\n" +payload.dataMsg) ;
        }
    }, function(error) {
      console.error(error);
    }) ;

    // NOTE: you can mix/use ->addDataArray(array()) and ->addData("key","value") in␣
↪same notification
    // NOTE: pass in preset array, then add a few extra custom key/values.
```

5.6 Data Only message

Client app is responsible for processing data messages. Data messages have only custom key-value pairs.

```
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);
```

(continues on next page)

(continued from previous page)

```php
$notification = new \Fcm\Push\Data();
$notification
    ->addData('test', '123');
    ->addRecipient($deviceId)

// Shortcut function:
// $notification = $client->pushData(['key' => 'value'], $deviceId);

$response = $client->send($notification);
```

Example response:

```
array(5) {
  'multicast_id' =>
  int(7676235924847328062 2)
  'success' =>
  int(1)
  'failure' =>
  int(0)
  'canonical_ids' =>
  int(0)
  'results' =>
  array(1) {
    [0] =>
    array(1) {
      'message_id' =>
      string(35) "0:1524927061384248%c5f39c08f9fd7ecd"
    }
  }
}
```

CHAPTER 6

Managing Device Groups

With device group messaging, you can send a single message to multiple instances of an app running on devices belonging to a group. Typically, "group" refers a set of different devices that belong to a single user. All devices in a group share a common notification key, which is the token that FCM uses to fan out messages to all devices in the group. Read more on at official documentation.

6.1 Creating a device group

When creating a new group, the response will contain a *notification key*, which will be used when adding or removing devices from the group, and sending messages to it.

```php
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

$newGroup = new \Fcm\DeviceGroup\Create('phones');
$newGroup->addDevice($deviceId);

// Shortcut function:
// $client->deviceGroupCreate('phones', $deviceId);

$client->send($newGroup);
```

Example response:

```
array(1) {
  'notification_key' =>
  string(119)
↪"APA91bE8asD44A2gjSUJqRp8Ym4pe7TlrlrSLVkKRBdvkoWOFmusdc87934ASDURl8xaUbXXdKC5DRkUssYtkOl_
↪lnWXT7gF0vO9E666XeL1qJs02FsunJ4"
}
```

6.2 Adding devices to a group

```php
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

$group = new \Fcm\DeviceGroup\Update('phones', $notificationKey);
$group->addDevice($deviceId);

// Shortcut function:
// $client->deviceGroupUpdate('phones', $notification_key, $deviceId);

$client->send($group);
```

Example response:

```
array(1) {
  'notification_key' =>
  string(119)
↪"APA91bE8asD44A2gjSUJqRp8Ym4pe7TlrlrSLVkKRBdvkoWOFmusdc87934ASDURl8xaUbXXdKC5DRkUssYtkOl_
↪lnWXT7gF0vO9E666XeL1qJs02FsunJ4"
}
```

6.3 Removing devices from a group

```php
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

$group = new \Fcm\DeviceGroup\Remove('phones', $notificationKey);
$group->addDevice($deviceId);

// Shortcut function:
// $client->deviceGroupRemove('phones', $notification_key, $deviceId);

$client->send($group);
```

Example response:

```
array(1) {
  'notification_key' =>
  string(119)
↪"APA91bE8asD44A2gjSUJqRp8Ym4pe7TlrlrSLVkKRBdvkoWOFmusdc87934ASDURl8xaUbXXdKC5DRkUssYtkOl_
↪lnWXT7gF0vO9E666XeL1qJs02FsunJ4"
}
```

CHAPTER 7

Managing Notification Topics

Based on the publish/subscribe model, FCM topic messaging allows you to send a message to multiple devices that have opted in to a particular topic. You compose topic messages as needed, and FCM handles routing and delivering the message reliably to the right devices.

For example, users of a local weather forecasting app could opt in to a "severe weather alerts" topic and receive notifications of storms threatening specified areas. Users of a sports app could subscribe to automatic updates in live game scores for their favorite teams.

Read here for more information.

7.1 Subscribing to a topic

When a topic does not exist, you can still subscribe to it, and the topic will be automaticly created.

```
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

$subscribe = new \Fcm\Topic\Subscribe('my_topic_name');
$subscribe->addDevice($deviceId);

// Shortcut function:
// $client->topicSubscribe('my_topic_name', $deviceId);

$client->send($subscribe);
```

Example response:

```
// When an error occurs, this will be filled with the message.
array(1) {
  'results' =>
  array(1) {
```

(continues on next page)

(continued from previous page)

```
    [0] =>
    array(0) {
    }
  }
}
```

7.2 Unsubscribing from a topic

Just like creating a topic, a topic will be automaticly deleted once all devices are unsubscribed from it.

```php
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

$unsubscribe = new \Fcm\Topic\Unsubscribe('my_topic_name');
$unsubscribe->addDevice($deviceId);

// Shortcut function:
// $client->topicUnsubscribe('my_topic_name', $deviceId);

$client->send($unsubscribe);
```

Example response:

```
// When an error occurs, this will be filled with the message.
array(1) {
  'results' =>
  array(1) {
    [0] =>
    array(0) {
    }
  }
}
```

CHAPTER 8

Package Options

Package options allow you to set options used within calls in the package. Package options can be configured in two ways.

8.1 Options

Options are set as an array. Options can be set on client instantiation or on an already created client.

```
<?php
$options = Array("http_errors" => true);
?>
```

Currently only one option is supported.

For any other sugestions for options, please add an issue to https://github.com/EdwinHoksberg/php-fcm/issues describing the new option.

8.2 Setting on instantiation

Settings options on instance creation is done as follows.

```
<?php

$options = Array("http_errors" => true);
$client = new \Fcm\FcmClient($serverKey, $senderId, $options);
```

8.3 Setting after instantiation

Options can be set after instantiation using the *setOptions* function passing in an options array.

```php
<?php

$client = new \Fcm\FcmClient($serverKey, $senderId);

# some code using the client

$options = Array("http_errors" => true);
$client->setOptions($options);

# some more code using the client

$options = Array("http_errors" => false);
$client->setOptions($options);
```

8.4 http_errors option

The *http_errors* option exposes the Guzzle *http_errors* parameter. By default, a HTTP error in Guzzle will cause an exception, so will terminate your code. By setting *http_errors* to *false* a HTTP error will not result in an exception.

An example of setting the options array might look like this.

```php
<?php

$options = Array("http_errors" => true);
```

Guzzle is the underlying library used by php-fcm, see http://docs.guzzlephp.org/en/stable/ for more details.

The *http-errors* option is described at http://docs.guzzlephp.org/en/stable/request-options.html#http-errors