Fcm Documentation

Release 1

Romain Monteil

Feb 23, 2019

Getting Started

Contents:

CHAPTER 1

Introduction

1.1 FCM PHP library

A PHP libray to send push notification with Firebase Cloud Messaging

It currently only supports HTTP protocol for :

- sending a downstream message to one or multiple devices
- managing groups and sending message to a group
- sending topics messages

1.2 Requirements

- PHP >= 7.0

CHAPTER 2

Installation

2.1 Using Composer

To install this plugin, run `composer require kerox/fcm` or add this snippet in your project's `composer.json`.

```
{
    "require": {
        "kerox/fcm": "~1.0"
    }
}
```

CHAPTER 3

Notification Builder

3.1 Introduction

You can build a notification by using the NotificationBuilder class.

3.2 Usage

```
use Kerox\Message\NotificationBuilder;

$notificationBuilder = new NotificationBuilder('Hello World');
$notificationBuilder
    ->setBody('body')
    ->setSound('sound')
    ->setBadge('badge')
    ->setIcon('icon')
    ->setTag('tag')
    ->setColor('#FFFFFF')
    ->setClickAction('click_action')
    ->setBodyLocKey('body_loc_key')
    ->setBodyLocArgs('body_loc_args')
    ->setTitleLocKey('title_loc_key')
    ->setTitleLocArgs('title_loc_args');
```

CHAPTER 4

Data Builder

4.1 Introduction

You can build data by using the DataBuilder class.

4.2 Usage

```
use Kerox\Message\DataBuilder;

$dataBuilder = new DataBuilder();
```

Adding datas

```
$dataBuilder
    ->setData('data-1', 'data-1')
    ->setData('data-2', true)
    ->setData('data-3', 1234);
```

All data passed to the DataBuilder will be converted as `string`.

Retrieving all datas

```
$dataBuilder->getData();
```

Retrieving a specific data

```
$dataBuilder->getData('data-1');
```

Delete all datas

```
$dataBuilder->removeData();
```

Delete a specific data

```
$dataBuilder->removeData('data-1');
```

CHAPTER 5

Options Builder

5.1 Introduction

You can build options for the notification by using the OptionsBuilder class.

5.2 Usage

```
use Kerox\Message\OptionsBuilder;

$optionsBuilder = new OptionsBuilder();
$optionsBuilder
    ->setRestrictedPackageName('foo')
    ->setCollapseKey('Update available')
    ->setPriority('normal')
    ->setTimeToLive(3600)
    ->setContentAvailable(true)
    ->setDryRun(true);
```

CHAPTER 6

Topics Builder

6.1 Introduction

You can build topics by using the TopicsBuilder class.

6.2 Usage

```
use Kerox\Message\TopicsBuilder;

$topicsBuilder = new TopicsBuilder('Topic A');
```

6.3 AND condition

```
$topicsBuilder->andTopic('Topic B');
```

Result: `'Topic A' in topics && 'Topic B' in topics`

6.4 OR condition

```
$topicsBuilder->orTopic('Topic B');
```

Result: `'Topic A' in topics || 'Topic B' in topics`

6.5 Subcondition

```
$topicsBuilder->andTopic(function () {
    return new TopicsBuilder('Topic B')->orTopic('Topic C');
})
```

Result: `'Topic A' in topics && ('Topic B' in topics || 'Topic C' in topics)`

CHAPTER 7

Fcm Sender

7.1 Introduction

You have three methods available to send push notification:

- `sendTo($targets)` to send downstream message.
- `sendToTopic($topic)` to send topic message.
- `sendToGroup($group)` to send group message.

Where:

- `$target` is a string or an array of devices's tokens. (required)
- `$topic` is a topic or conditions of topics. (required)
- `$group` is a group. (required)

7.2 Global

```
use Kerox\Fcm;

$fcm = new Fcm($apiKey);
$fcm->setNotification($notification)
    ->setData($data)
    ->setOptions($options);
```

Where:

- `$apiKey` is your FCM API key.
- `$notification` can be an NotificationBuilder object or an array containing the notification. (optional)
- `$data` can be a DataBuilder object or an array with some data that will be passed. (optional)

- `$options` can be an OptionsBuilder object or an array of options for the payload. (optional)

If you passed arrays to `setNotification()`, `setData()`, or `setOptions`, their will be converted to builder object.

7.3 Downstream Message

```
$fcm->sendTo($target);
```

7.4 Topic Message

```
$fcm->sendToTopic($topic);
```

7.5 Group Message

```
$fcm->sendToGroup($group);
```

For more details on sending message, refer to the FCM documentation

CHAPTER 8

Group Sender

8.1 Introduction

You have three methods available to manage group:

- `createGroup($groupName, $devicesToken)` to create a group.
- `addToGroup($groupName, $notificationKey, $devicesToken)` to add one or more devices to a group.
- `removeFromGroup($groupName, $notificationKey, $devicesToken)` to remove one or more devices from a group.

Where:

- `$groupName` is a string containing the name of the group.
- `$deviceToken` can be a string or an array containing devices's token.
- `$notificationKey` is the notification key return by the request.

A successful request returns a notification_key like the following:

```
{
   "notification_key": "APA91bGHXQBB...9QgnYOEURwm0I3lmyqzk2TXQ"
}
```

8.2 Global

```
use Kerox\FcmGroup;

$fcmGroup = new FcmGroup($apiKey, $senderId);
```

Where:

- `$apiKey` is your FCM API key. (required)
- `$senderId` is your sender ID. (required)

8.3 Creating a group

```
$notificationKey = $fcmGroup->createGroup('myGroup', $deviceToken);
```

8.4 Adding devices to a group

```
$notificationKey = $fcmGroup->addToGroup('myGroup', $notificationKey, $deviceToken)
```

8.5 Removing devices from a group

```
$notificationKey = $fcmGroup->removeFromGroup('myGroup', $notificationKey,
↪$deviceToken)
```

For more details on group in Firebase, refer to the Firebase documentation

CHAPTER 9

Downstream Response

9.1 Introduction

To get the response of a downstream request, just save the return of the `sendTo()` method into a variable.

```
$response = $fcm->sendTo(['1', '2', '3', '4']);
```

You have seven methods available to read downstream response:

- `getNumberSuccess()`: Return the number of messages that were processed without an error.

```
$response->getNumberSuccess();
```

- `getNumberFailure()`: Return the number of messages that could not be processed.

```
$response->getNumberFailure();
```

- `getNumberModify()`: Return the number of results that contain a canonical registration token.

```
$response->getNumberModify();
```

- `getTargetsToDelete()`: Return an array of tokens that you should remove in your database.

```
$response->getTargetsToDelete();
```

- `getTargetsToModify()`: Return an array of tokens (key : old token, value : new token) that you should change in your database.

```
$response->getTargetsToModify();
```

- `getTargetsToRetry()`: Return an array of tokens you should try to resend the message.

```
$response->getTargetsToRetry();
```

- `getTargetsWithError()`: Return an array of tokens that could not be processed with their error.

```
$response->getTargetsWithError();
```

For more details on downstream response, refer to the FCM documentation

CHAPTER 10

Topic Response

10.1 Introduction

To get the response of a topic request, just save the return of the `sendToTopic()` method into a variable.

```
$response = $fcm->sendToTopic($topic);
```

You have three methods available to read downstream response:

- `isSuccess()`: Return `true` if topic was sent with success.

```
$response->isSuccess();
```

- `shouldRetry()`: Return `true` if topic must be resent.

```
$response->shouldRetry();
```

- `getError()`: Return the error message if topic couldn't be sent.

```
$response->getError();
```

For more details on topic response, refer to the FCM documentation

CHAPTER 11

Group Response

11.1 Introduction

To get the response of a group request, just save the return of the `sendToGroup()` method into a variable.

```
$response = $fcm->sendToGroup($group);
```

You have three methods available to read downstream response:

- `getNumberSuccess()`: Return the number of messages that were processed without an error.

```
$response->getNumberSuccess();
```

- `getNumberFailure()`: Return the number of messages that could not be processed.

```
$response->getNumberFailure();
```

- `getTargetsFailed()`: Return a lists of registration tokens that failed to receive the message.

```
$response->getTargetsFailed();
```

For more details on group response, refer to the Device Group Messaging documentation

CHAPTER 12

Basic Examples

12.1 Downstream message

Sending a downstream message from arrays.

```
use Kerox\Fcm;

// Create a downstream message from arrays
$fcm = new Fcm('YOUR_FCM_API_KEY');
$fcm->setNotification([
        'title' => 'Hello World',
        'body' => 'My awesome Hello World!'
    ])
    ->setData([
        'data-1' => 'Lorem ipsum',
        'data-2' => 1234,
        'data-3' => true
    ])
    ->setOptions([
        'dry_run' => true
    ]);

// Send the message and get the response
$response = $fcm->sendTo(['1', '2', '3', '4']);
```

12.2 Topic message

Sending a topic message from arrays.

```
use Kerox\Fcm;
use Kerox\Fcm\Message\TopicBuilder;
```

(continues on next page)

(continued from previous page)

```
$topicBuilder = new TopicBuilder('myTopic');
$topic = $topicBuilder->build();

// Create a downstream message from arrays
$fcm = new Fcm('YOUR_FCM_API_KEY');
$fcm->setNotification([
        'title' => 'Hello World',
        'body' => 'My awesome Hello World!'
    ])
    ->setData([
        'data-1' => 'Lorem ipsum',
        'data-2' => 1234,
        'data-3' => true
    ])
    ->setOptions([
        'dry_run' => true
    ]);

// Send the message and get the response
$response = $fcm->sendToTopic($topic);
```

CHAPTER 13

Advanced Examples

13.1 Downstream message

Sending a downstream message using builders.

```
use Kerox\Fcm\Fcm;
use Kerox\Fcm\Message\DataBuilder;
use Kerox\Fcm\Message\NotificationBuilder;
use Kerox\Fcm\Message\OptionsBuilder;

// Create the notification
$notificationBuilder = new NotificationBuilder('Hello World');
$notificationBuilder
    ->setBody('My awesome Hello World');
    ->setSound('sound')
    ->setBadge('badge')
    ->setIcon('icon')
    ->setTag('tag')
    ->setColor('#FFFFFF')
    ->setClickAction('click_action')
    ->setBodyLocKey('body_loc_key')
    ->setBodyLocArgs('body_loc_args')
    ->setTitleLocKey('title_loc_key')
    ->setTitleLocArgs('title_loc_args')

// Create the data
$dataBuilder = new DataBuilder();
$dataBuilder
    ->setData('data-1', 'data-1')
    ->setData('data-2', true)
    ->setData('data-3', 1234);

// Create the options
$optionsBuilder = new OptionsBuilder();
```

(continues on next page)

(continued from previous page)

```
$optionsBuilder
    ->setCollapseKey('Update available')
    ->setPriority('normal')
    ->setTimeToLive(3600)
    ->setContentAvailable(true)
    ->setDryRun(true);

// Build
$notification = $notificationBuilder->build();
$data = $dataBuilder->build();
$options = $optionsBuilder->build();

$fcm = new Fcm($this->api_key);
$fcm->setNotification($notification)
    ->setData($data)
    ->setOptions($options);

$response = $fcm->sendTo(['1', '2', '3', '4']);
```

Sending a topic message using builders.

```
use Kerox\Fcm\Fcm;
use Kerox\Fcm\Message\DataBuilder;
use Kerox\Fcm\Message\NotificationBuilder;
use Kerox\Fcm\Message\OptionsBuilder;
use Kerox\Fcm\Message\TopicsBuilder;

// Create topics
$topicBuilder = new TopicBuilder('My first topic');
$topicsBuilder->andTopic(function () {
    return new TopicsBuilder('My second topic')->orTopic('My third topic');
})

// Create the notification
$notificationBuilder = new NotificationBuilder('Hello World');
$notificationBuilder
    ->setBody('My awesome Hello World');
    ->setSound('sound')
    ->setBadge('badge')
    ->setIcon('icon')
    ->setTag('tag')
    ->setColor('#FFFFFF')
    ->setClickAction('click_action')
    ->setBodyLocKey('body_loc_key')
    ->setBodyLocArgs('body_loc_args')
    ->setTitleLocKey('title_loc_key')
    ->setTitleLocArgs('title_loc_args')

// Create the data
$dataBuilder = new DataBuilder();
$dataBuilder
    ->setData('data-1', 'data-1')
    ->setData('data-2', true)
    ->setData('data-3', 1234);

// Create the options
$optionsBuilder = new OptionsBuilder();
```

(continues on next page)

(continued from previous page)

```
$optionsBuilder
    ->setCollapseKey('Update available')
    ->setPriority('normal')
    ->setTimeToLive(3600)
    ->setContentAvailable(true)
    ->setDryRun(true);

// Build
$notification = $notificationBuilder->build();
$data = $dataBuilder->build();
$options = $optionsBuilder->build();
$topic = $topicBuilder->build();

$fcm = new Fcm($this->api_key);
$fcm->setNotification($notification)
    ->setData($data)
    ->setOptions($options);

$response = $fcm->sendToTopic($topic);
```

CHAPTER 14

Indices and tables

- genindex
- modindex
- search