# Firebase PHP SDK

**Feb 24, 2018**

# Contents

A PHP client for the Google Firebase Realtime Database

**Note:** This is a 3rd party SDK and not maintained or supported by Firebase or Google.

```php
use Kreait\Firebase\Configuration;
use Kreait\Firebase\Firebase;

$config = new Configuration();
$config->setAuthConfigFile('/path/to/google-service-account.json');

$firebase = new Firebase('https://my-app.firebaseio.com', $config);

$firebase->set(['key' => 'value'], 'my/data');
$firebase->set('new value', 'my/data/key');

print_r($firebase->get('my/data'));

$firebase->delete('my/data');
```

CHAPTER 1

# User Guide

## 1.1 Overview

### 1.1.1 Requirements

1. PHP >5.5
2. A Firebase Project at https://firebase.google.com
3. A Google Service Account JSON config file or a database secret

### 1.1.2 Installation

The recommended way to install the Firebase PHP SDK is with Composer.

```
composer require kreait/firebase-php ^1.0
```

After installing, you need to require Composer's autoloader:

```
require 'vendor/autoload.php';
```

You can find out more on how to install Composer, configure autoloading, and other best-practices for defining dependencies at getcomposer.org.

## 1.2 Authentication

### 1.2.1 With a Google Service Account

To use this authentication method, you must first create a Service account as described in the official documentation.

**Note:** The Service Account needs at least the *Editor* role to be able to modify data.

Put the downloaded credentials JSON file in a path of your project and configure the Firebase SDK to use it:

```
use Kreait\Firebase\Configuration;
use Kreait\Firebase\Firebase;

$config = new Configuration();
$config->setAuthConfigFile(__DIR__.'google-service-account.json');

$firebase = new Firebase('https://my-app.firebaseio.com', $config);
```

All requests to the Firebase Realtime Database will now be performed with the permissions of this Service Account.

### 1.2.2 With a database secret

**Warning:** Using the database secret is considered a legacy method. It is not sure how long Google will support this authentication method.

After retrieving a database secret from the project settings of your Firebase application, you can configure the Firebase SDK with it:

```
use Kreait\Firebase\Configuration;
use Kreait\Firebase\Firebase;

$config = new Configuration();
$config->setFirebaseSecret('...');

$firebase = new Firebase('https://my-app.firebaseio.com', $config);
```

All requests to the Firebase Realtime Database will now be performed with full read and write access.

### 1.2.3 Authentication overrides

You can override the main authentication to impersonate users of your application by providing a UID and optionally some claims to the Firebase object:

```
$uid = 'some-unique-user-id';
$claims = ['premiumUser' => true];

$firebase->setAuthOverride($uid, $claims);

$data = $firebase->get('/premium-content');

$firebase->removeAuthOverride();
```

## 1.3 Retrieving data

### 1.3.1 Direct access

```
$result = $firebase->get('my/data')
```

### 1.3.2 Using a Reference

```
$reference = $firebase->getReference('my/data');
// or
$reference = $firebase->my->data;

$data = $reference->getData();
```

### 1.3.3 Queries

Queries can be executed on the Firebase object or a Reference.

```
use Kreait\Firebase\Firebase;
use Kreait\Firebase\Query;

$firebase = new Firebase(...);
$query = new Query();

$firebase->query($query);
// or
$firebase->my->data->query($query);
```

Query methods for sorting and filtering can be chained:

```
$query
    ->orderByKey()
    ->startAt('a')
    ->endAt('d');
```

#### Sorting results

```
$query->orderByChildKey($key);
$query->orderByKey();
$query->orderByPriority();
```

#### Filtering results

```
$query->limitToFirst($limit);
$query->limitToLast($limit);
$query->startAt($start);
$query->endAt($end);
$query->equalTo($value);
$query->shallow();
```

## 1.4 Writing data

The following code examples operate on the following data set:

```
$data = [
    'key_1' => [
        'key_1_1' => 'value',
        'key_1_2' => 'value',
    ],
    'key_2' => [
        'key_2_1' => 'value',
        'key_2_2' => 'value',
    ],
    'key_3' => [
        'key_3_1' => 'value',
        'key_3_2' => 'value',
    ],
];
```

### 1.4.1 Write/Replace

```
$firebase->set([
    'key_1' => 'value_1',
    'key_2' => 'value_2',
    ], 'my/data');

$firebase->set('my/data/key_1', 'value_1');
```

### 1.4.2 Update

```
$firebase->update('value_1_1', 'key_1/key_1_1');

$firebase->update([
    'key_1/key_1_1' => 'value_1_1',
    'key_2/key_2_2' => 'value_2_2',
], 'my/data');
```

### 1.4.3 Add children

```
$newKey = $firebase->push([
    'child_key_1' => 'value_1',
    'child_key_2' => 'value_2',
], 'my/data');
```

`$newKey` will be populated with the name of the new child, which is generated by the Firebase database, e.g. `-KMkDDGJQ9vuooQapvdv`.

### 1.4.4 Delete

```
$firebase->delete('my/data');
```

## 1.5 Configuration

```
use Kreait\Firebase\Configuration;
use Kreait\Firebase\Firebase;

$config = new Configuration();

$firebase = new Firebase('https://my-app.firebaseio.com', $config);
```

### 1.5.1 Logging

Any PSR-3 compliant logger can be used for logging. The following example uses Monolog:

```
use Monolog\Logger;
use Monolog\Handler\StreamHandler;

$logger = new Logger('firebase-php');
$logger->pushHandler(new StreamHandler('path/to/your.log', Logger::DEBUG));

$config->setLogger($logger);
```

### 1.5.2 Custom HTTP Adapter

By default, the library tries to find an already existing instance of an HTTP adapter, and if none is found, it will create a new one. You can override the used HTTP adapter by setting it in the configuration like this:

```
use Ivory\HttpAdapter\FopenHttpAdapter;

$http = new FopenHttpAdapter(); // or any other available adapter

$config->setHttpAdapter($http);
```

## 1.6 About this project

### 1.6.1 License

Licensed using the MIT license.

> Copyright (c) 2015-2016 Jérôme Gamez <https://github.com/jeromegamez>
>
> 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:

### 1.6.2 Acknowledgments

- This library would not be possible without the continuous support of kreait, the company I work for. Give us a shout on Twitter, or join us at jobs.kreait.com!