php-opencloud Documentation

Release 1.12.1

Jamie Hannaford

March 29, 2016

Contents

CHAPTER 1

Requirements

- PHP 7

CHAPTER 2

Installation

You must install this library through Composer:

```
composer require rackspace/php-opencloud
```

If you do not have Composer installed, please read the Composer installation instructions.

Once you have installed the SDK as a dependency of your project, you will need to load Composer's autoloader (which registers all the required namespaces). To do this, place the following line of PHP code at the top of your application's PHP files:

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

This assumes your application's PHP files are located in the same folder as `vendor/`. If your files are located elsewhere, please supply the path to vendor/autoload.php in the require statement above.

CHAPTER 3

Supported services

3.1 Cloud Servers v2

3.1.1 Servers

Create server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Waiting for builds to complete

Some builds can take minutes to complete. To ensure your code operates on a fully built server, you can wait until it reaches an `ACTIVE` state. This is a blocking operation which continually polls the API:

```
$timeout = 300; // seconds
$server->waitUntilActive($timeout);
```

List servers

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Generators

Each iteration will return a **:apiref:‘Flavor instance <Rackspace/Compute/v2/Models/Flavor.html>‘**.

By default, PHP generators are used to represent collections of resources in the SDK. The benefit of using generators is that it generally improves performance, since objects are not saved in memory as the iteration cycle goes on; instead, each resource is directly output to the user-defined `foreach` loop. For all intents and purposes, you interact with generators like any other Traversable object, but to retain collections in memory, you will need to implement your own logic.

Detailed information

By default only a small subset of information is returned for each server iteration. To enable a more detailed representation, you can pass `true` into the method, like so:

```
$servers = $service->listServers(true);
```

Retrieve details of a server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

List server IP addresses

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Reboot server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Rescue server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Unrescue server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Update server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Note: The only attributes you can update are `name`, `accessIPv4` and `accessIPv6`.

Change admin password

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Resize server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Confirm resize

Once you have resized a server, you need to confirm the resize and transition it to an active state:

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Revert resize

You can revert your changes at any time before confirming like so:

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Delete server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Retrieve metadata

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Merge new metadata with old

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Reset all existing metadata with new

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

3.1.2 Flavors

List flavors

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Each iteration will return a **:apiref:‘Flavor instance <Rackspace/Compute/v2/Models/Flavor.html>‘**.

By default, PHP generators are used to represent collections of resources in the SDK. The benefit of using generators is that it generally improves performance, since objects are not saved in memory as the iteration cycle goes on; instead,

each resource is directly output to the user-defined `foreach` loop. For all intents and purposes, you interact with generators like any other Traversable object, but to retain collections in memory, you will need to implement your own logic.

Retrieve details about a flavor

When retrieving a flavor, sometimes you only want to operate on it. If this is the case, then there is no need to perform an initial GET request to the API:

If, however, you *do* want to retrieve all the details of a remote resource from the API, you just call:

```
$flavor->retrieve();
```

which will update the state of the local object. This gives you an element of control over your app's performance.

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Retrieve extra specifications of a flavor

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

3.1.3 Images

Create image from server

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Retrieve details of an image

When retrieving an image, sometimes you only want to operate on it. If this is the case, then there is no need to perform an initial GET request to the API:

If, however, you *do* want to retrieve all the details of a remote resource from the API, you just call:

```
$image->retrieve();
```

which will update the state of the local object. This gives you an element of control over your app's performance.

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

List all images

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Each iteration will return an **:apiref:'Image instance <Rackspace/Compute/v2/Models/Image.html>'**.

By default, PHP generators are used to represent collections of resources in the SDK. The benefit of using generators is that it generally improves performance, since objects are not saved in memory as the iteration cycle goes on; instead, each resource is directly output to the user-defined `foreach` loop. For all intents and purposes, you interact with

generators like any other Traversable object, but to retain collections in memory, you will need to implement your own logic.

Get image metadata

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Merge new metadata with old

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Reset all existing metadata with new

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Delete image

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

3.1.4 Keypairs

Generate a new keypair

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Import existing keypair

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

List all keypairs

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Each iteration will return a **:apiref:‘Keypair instance <Rackspace/Compute/v2/Models/Keypair.html>‘**.

By default, PHP generators are used to represent collections of resources in the SDK. The benefit of using generators is that it generally improves performance, since objects are not saved in memory as the iteration cycle goes on; instead, each resource is directly output to the user-defined `foreach` loop. For all intents and purposes, you interact with generators like any other Traversable object, but to retain collections in memory, you will need to implement your own logic.

Delete keypair

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

3.1.5 Networks

Create a network

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

List networks

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Retrieve details for a network

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Update a network

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Delete a network

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

3.1.6 Server backups

Enable weekly backups

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Enable daily backups

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Retrieve details of a backup schedule

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Disable a backup schedule

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

3.1.7 Virtual Interfaces

Setup

Since a virtual interface operates on a server, you will need to retrieve the server you want to act on first. To do so, you will need its unique ID:

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Create a virtual interface

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

List all virtual interfaces

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Delete a virtual interface

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

3.1.8 Volume Attachments

Setup

Since a volume attachment operates on a server, you will need to retrieve the server you want to act on first. To do so, you will need its unique ID:

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Attach a volume

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Detach a volume

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

Retrieve details about a volume attachment

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

List volume attachments

To see all the required and optional parameters for this operation, along with their types and descriptions, view the reference documentation.

CHAPTER 4

Help and support

If you have specific problems or bugs with this SDK, please file an issue on our official Github repo. We also have a mailing list, so feel free to join to keep up to date with all the latest changes and announcements to the library.

For general feedback and support requests, send an email to sdk-support@rackspace.com.

You can also find assistance via IRC on #rackspace at freenode.net.

CHAPTER 5

Contributing

If you'd like to contribute to the project, or require help running the unit/integration tests, please view the contributing guidelines.