Libraries

The Mailgun API is built on HTTP. Our API is RESTful and it:

- Uses predictable, resource-oriented URLs.
- Uses built-in HTTP capabilities for passing parameters and authentication.
- Responds with standard HTTP response codes to indicate errors.
- Returns JSON.

Mailgun has published libraries for various languages. You may use our libraries, or your favorite/suggested HTTP/REST library available for your programming language, to make HTTP calls to Mailgun.

Most code samples in our docs can be viewed in several programming languages by using the language bar at the top. Below are the language-specific notes we feel are useful.

Python

Standard HTTP Library:

Python users love Requests HTTP library. It is simple, elegant and yet very powerful. To install it you would simply run:

```
pip install requests
```

You may also need a MultiDict class to represent HTTP requests with multiple values per key. We recommend WebOb's MultiDict but Werkzeug/Flask also offer MultiDict class.

Note

Our code samples use Requests version 0.7.5. Older versions of Requests don't take multi dictionaries. For post data and query params you could use lists to pass multi-valued keys. Unfortunately it won't work for files.

Ruby

Standard HTTP Library:

Ruby folks recommend rest-client and not without a reason: it is one of the most beautiful REST HTTP libraries we have seen. The library is available as gem, so to install it simply run:

```
gem install rest-client
```

Official Mailgun Ruby Gem:

This is the Mailgun Ruby Library. This library contains methods for easily interacting with the Mailgun API. Fork it on GitHub or read more about the gem here.

```
gem install mailgun-ruby
```

Java

Standard HTTP Library:

Check out jersey REST client if Java is your weapon of choice. It took us 3 jars to get all we needed:

- jersey-client.jar (version ~ 1.17 - 1.18.1)
- jersey-core.jar (version ~ 1.17 - 1.18.1)
- jersey-multipart.jar (version ~ 1.17 - 1.18.1)

C#

Standard HTTP Library:

For C# developers there is RestSharp. And that's it, nothing else is required. Standard .NET makes it easy to make HTTP requests.

However, if you are using mono you will most likely need to allow it to do HTTP requests to external sites first. The easiest way to do that is probably by installing Mozilla certificates, like so:

```
mozroots --import --sync
```

PHP

Mailgun Library:

Our PHP library is robust and provides an excellent interface to easily interact with our API.

Github Repository: mailgun-php

Minimum PHP Version: 5.5.0

To install the library, you will need to be using Composer in your project. If you aren't using Composer yet, it's really simple! Here's how to install composer and the Mailgun library.

```
# Install Composer
curl -sS https://getcomposer.org/installer | php
```

```
# Add Mailgun and Guzzle6 as a dependency (see Github README below for more
info)
php composer.phar require mailgun/mailgun-php php-http/guzzle6-adapter php-
http/message
```

Next, just include Composer's autoloader in your application to automatically load the Mailgun library in your project.

```
require 'vendor/autoload.php';
use Mailgun\Mailgun;
$mailgun = new Mailgun('api_key', new \Http\Adapter\Guzzle6\Client());
```

For additional information, see the Github Repository README file.

Standard HTTP Library:

PHP users can use PHP cURL library.

Below are all the steps needed to install this library from a fresh Ubuntu installation.

Run:

```
sudo aptitude install libmagic-dev
sudo aptitude install php5-dev
```

Then to enable curl support:

```
sudo aptitude install libcurl3
```

Then if you plan to run scripts from CLI:

```
sudo aptitude install php5-cli
```

To install cURL for php which we used for the ability to send put data:

```
sudo aptitude install php5-curl
```

That should be all. Quite a list, isn't it? But firstly, we had only a fresh Ubuntu installation when we started and secondly, once the library is installed, making HTTP requests becomes no more difficult than in any other language.

Node.js

Check out the available node modules from the community.

We also have a step by step tutorial post on sending email with Node.js.

Luvit

Lua and luvit users have two easy options. Either the ‘luvit-curl <https://github.com/dvv/luvit-curl>’_ library or the ‘luvit-request <https://github.com/virgo-agent-toolkit/luvit-request>’_ library.

Due to luvits asynchronous i/o nature, code samples from node.js can be easily retrofitted to work in luvit with luvit libraries.

cURL

curl is a popular command line tool to send HTTP requests. It is very simple and yet quite powerful. With it you could send data using any HTTP method. You could send post data and query params and files in a very consistent and elegant way. An exellent choice to study the API.