Testing out your Pipl API integration

The following guide describes how to test your Pipl API integration and usage without incurring any or minimal charges.

Free $50 credit

When signing up for a new Pipl API account and completing the short survey, you'll automatically get a $50 development credit allowance valid for a month from signup. This credit will be activated as soon as you specify your payment method and appear as a credit on your next invoice.

The free Clark Kent email address search

If you query the API with the email address '**clark.kent@example.com** ' a full example Person response will be returned. You will not be charged for this response when using any of your production or demo keys.

Simply including the 'clark.kent@example.com' email address as an additional input to any query lets the system know you are testing, and you will not be charged for the response. This also works when you do a person object search with multiple email address as long as clark.kent@example.com is included in the query, just be sure to add the clark.kent@example.com as the first email in the array.

Examples in different code libraries are available in the API reference.

Caching responses and testing response types

Since the clark.kent@example.com returns a Person response, you may wish to test your application with a Possible Person response, a non matching response, or a response that returns warnings and/or errors.

For testing other response types it is recommended that you build a cache system. This way, when in development mode your application would read from the cache instead of calling the Pipl API, and you can test your system for different response scenarios. Examples of JSON output have been provided below for your convenience.

Possible Person response

To get a possible person response from the API is as simple as searching a common name in a certain location such as

- first_name=“James”
- last_name=”Smith”
- country=”US”
- state=”ID”

200 no match response (no data found)

Here the API was queried with the email address “thereisnosuchemailaddress@example.com ”

```
{
    "@http_status_code": 200,
    "@visible_sources": 0,
    "@available_sources": 0,
    "@persons_count": 0,
    "@search_id": "1803212112122292586844880443055099531",
    "query": {
        "emails": [
            {
                "address": "thereisnosuchemailaddress@example.com",
                "address_md5": "f65a71d57980d5791b41c49d72075adc"
            }
        ]
    } }
```

200 no match response (match criteria not met)

Here the API was queried with the email address “roy.kfir@pipl.com ” and match_requirements “phone.home_phone”

```
{
    "@http_status_code": 200,
"@visible_sources": 0,
    "@available_sources": 93,
    "@persons_count": 1,
    "@search_id": "1804021428292512890210981053256463098",
    "match_requirements": "phone.home_phone",
    "query": {
        "emails": [
            {
                "address": "roy.kfir@pipl.com",
                "address_md5": "3016cb849c9ec8819490e6a61d7bc868"
            }
        ]
    },
    "available_data": {
        "premium": {
            "relationships": 15,
            "usernames": 3,
"jobs": 6,
            "addresses": 2,
            "phones": 3,
            "mobile_phones": 3,
            "educations": 3,
            "languages": 2,
            "user_ids": 7,
            "social_profiles": 7,
            "names": 1,
            "images": 7,
            "genders": 1,
            "emails": 3,
            "origin_countries": 1
        }
    } }
```

pipl

400, 403 and 500 response

Examples of errors (400, 403, 500) and warnings are available in the API reference documentation.

Unit test examples with the Clark Kent email

When integrating with Pipl's API you may want to test different combinations of data field inputs in your application logic.

You may have the following data field input combinations to test:

- First Name + Last Name
- First Name + Middle Name + Last Name
- First Name + Last Name + Address
- First Name + Middle Name + Last Name + Address
- Email Address
- First Name + Last Name + Email Address
- First Name + Middle Name + Last Name + Email Address

We recommend having your application run in a test mode which automatically adds clark.kent@example.com to the query.

So in the above examples, the test function would simply add the email as follows:

- First Name + Last Name + *clark.kent@example.com*
- First Name + Middle Name + Last Name + *clark.kent@example.com*
- First Name + Last Name + Address + *clark.kent@example.com*
- First Name + Middle Name + Last Name + Location + *clark.kent@example.com*

With the combinations that included email, you could replace the email with the Clark Kent one:

- *clark.kent@example.com*
- First Name + Last Name + *clark.kent@example.com*
- First Name + Middle Name + Last Name + *clark.kent@example.com*

When using person search requests with multiple data fields of the same type, please note that you must include the clark.kent@example.com email address as the first email in the array:

- (*clark.kent@example.com* + other Email Address)
- (*clark.kent@example.com* + other Email Address) + First Name + Last Name
- (*clark.kent@example.com* + other Email Address) + First Name + Middle Name + Last Name

e.g:

person=`{"emails":[{"address":"clark.kent@example.com"},{"address":"yourtestemail@example.com"}]}`

and URL encode this parameter value:
%7B%22emails%22%3A%5B%7B%22address%22%3A+%22clark.kent%40example.com%22%7D%2C%7B%22address%22%3A%22yourtestemail%40example.com%22%7D%5D%7D

So a query will look like:
http://api.pipl.com/search/?person=%7B%22emails%22%3A%5B%7B%22address%22%3A+%22clark.kent%40example.com%22%7D%2C%7B%22address%22%3A%22yourtestemail%40ex ample.com%22%7D%5D%7D&key=<YOUR_KEY_HERE>