Email Address Verification API

Release 2.0.1

Email Hippo Ltd.

May 16, 2017

Contents

CHAPTER 1

Latest Uptime Statistics

Live Uptime Report

Report shows functional requests. Functional requests are queries containing real email addresses for validation.

Live Response Time Report

Report shows response times from the functional *API* endpoint.

CHAPTER 2

Documentation Introduction

This document will show you how to get up and running with the email verification *API*. You will have the basics of the *API* up and running in 15 minutes or less.

Use:

- Signup for a free account.
- Verify email addresses from inside your portal.
- Integrate email validation with your application using the *RESTful API*.

Engage:

- Create a support ticket
- Twitter
- Facebook
- Google+
- LinkedIn

Quick Start

Quick Start

This quick start guide is designed to get you up and running as fast as possible.

Please follow the steps below in sequence:

1) Create Account

Create your account.

2) Login

Login to your customer portal to retrieve your *API License Key*.

3) Try it

Plug your license key into the following

```
https://api1.27hub.com/api/emh/a/v2?k=INSERTYOURLICENSEKEY&e=john.doe@gmail.com
```

Paste the url above into your browser and watch the response come back as follows:

```
{
        "result": "Bad",
        "reason": "MailboxDoesNotExist",
        "role": false,
        "free": true,
        "disposable": false,
        "email": "john.doe@gmail.com",
        "domain": "gmail.com",
        "user": "john.doe",
        "mailServerLocation": "US",
        "duration": 723
}
```

Note: Internet Explorer may prompt to download the file instead of simply displaying it on screen. This is a quirk of Internet Explorer and not an issue with the *API*. We do not recommend Internet Explorer for testing with the *API*. Instead, use Chrome or Firefox - both will display the results on screen correctly!

Data Dictionary

Data Dictionary For API V2

A response is a message consisting of a standard *HTTP* header and body. The body of the message contains the detail of the message (e.g. the *JSON* data with email verification detail). The header of the message contains general *HTTP* information such as *HTTP* status codes.

Response Body Content

Field Name	Type	Description	Example Data
result	string	*Main Status Response Codes*	Bad
reason	string	*Additional Status Codes*	MailboxDoesNotExist
role	bool	Is a *Role Address*?	false
free	bool	Is a *Free Mail*?	true
disposable	bool	Is a *DEA*?	false
email	string	The email address supplied to the request.	john.doe@gmail.com
domain	string	The domain of an email address.	gmail.com
user	string	The user of an email address.	john.doe
mailServerLocation	string	Server location. *ISO 3166* country codes.	US
duration	integer	The elapsed time(*ms*) to execute the query.	649

Main Status Response Codes

Ok Verification passes all checks including Syntax, *DNS*, *MX*, Mailbox, Deep Server Configuration, *Grey Listing*

Bad Verification fails checks for definitive reasons (e.g. mailbox does not exist)

RetryLater Conclusive verification result cannot be achieved at this time. Please try again later. - This is ShutDowns, IPBlock, TimeOuts

Unverifiable Conclusive verification result cannot be achieved due to mail server configuration or anti-spam measures. See table "Additional Status Codes".

Additional Status Codes

None No additional information is available.

This status differs from a TransientNetworkFault as it should not be retried (the result will not change).

There are a few known reasons for this status code for example the target mx record uses *Office 365* or a mail provider implementing custom mailbox shutdowns.

AtSignNotFound The required '@' sign is not found in email address.

DomainIsInexistent The domain (i.e. the bit after the '@' character) defined in the email address does not exist, according to *DNS* records.

A domain that does not exist cannot have email boxes. A domain that does not exist cannot have email boxes.

DomainIsWellKnownDea The domain is a well known Disposable Email Address *DEA*.

There are many services available that permit users to use a one-time only email address. Typically, these email addresses are used by individuals wishing to gain access to content or services requiring registration of email addresses but same individuals not wishing to divulge their true identities (e.g. permanent email addresses).

DEA addresses should not be regarded as valid for email send purposes as it is unlikely that messages sent to DEA (Disposable Email Address) addresses will ever be read.

GreyListing *Grey Listing* is in operation. It is not possible to validate email boxes in real-time where grey listing is in operation.

MailboxFull The mailbox is full.

Mailboxes that are full are unable to receive any further email messages until such time as the user empties the mail box or the system administrator grants extra storage quota.

Most full mailboxes usually indicate accounts that have been abandoned by users and will therefore never be looked at again.

We do not recommend sending emails to email addresses identified as *full*.

MailboxDoesNotExist The mailbox does not exist.

100% confidence that the mail box does not exist.

NoMxServersFound There are no mail servers defined for this domain, according to *DNS*.

Email addresses cannot be valid if there are no email servers defined in *DNS* for the domain.

ServerDoesNotSupportInternationalMailboxes The server does not support international mailboxes.

International email boxes are those that use international character sets such as Chinese / Kanji etc.

International email boxes require systems in place for *Punycode* translation.

Where these systems are not in place, email verification or delivery is not possible.

For further information see *Punycode*.

ServerIsCatchAll The server is configured for *catch all* and responds to all email verifications with a status of *Ok*.

Mail servers can be configured with a policy known as *Catch All*. Catch all redirects any email address sent to a particular domain to a central email box for manual inspection. Catch all configured servers cannot respond to requests for email address verification.

Success Successful verification.

100% confidence that the mailbox exists.

TooManyAtSignsFound Too many '@' signs found in email address.

Only one '@' character is allowed in email addresses.

Unknown The reason for the verification result is unknown.

TransientNetworkFault A temporary network fault occurred during verification. Please try again later.

Verification operations on remote mail servers can sometimes fail for a number of reasons such as loss of network connection, remote servers timing out etc.

These conditions are usually temporary. Retrying verification at a later time will usually result in a positive response from mail servers.

Please note that setting an infinite retry policy around this status code is inadvisable as there is no way of knowing when the issue will be resolved within the target domain or the grey listing resolved, and this may affect your daily quota.

PossibleSpamTrapDetected A possible spam trap email address or domain has been detected.

Spam traps are email addresses or domains deliberately placed on-line in order to capture and flag potential spam based operations.

Our advanced detection heuristics are capable of detecting likely spam trap addresses or domains known to be associated with spam trap techniques.

We do not recommend sending emails to addresses identified as associated with known spam trap behaviour.

Sending emails to known spam traps or domains will result in your *ESP* being subjected to email blocks from a *DNS Block List*.

An *ESP* cannot tolerate entries in a *Block List* (as it adversely affects email deliverability for all customers) and will actively refuse to send emails on behalf of customers with a history of generating entries in a *Block List*.

Response Header

HTTP Status Codes

In additional to the application level codes (see *Main Status Response Codes* and *Additional Status Codes*) returned in the *HTTP* message body, *HTTP* status codes are returned in the *HTTP* header.

200 Call successful.

400 Bad request. The server could not understand the request. Perhaps missing a license key or an email to check? Conditions that lead to this error are: No license key supplied, no email address supplied, email address > 255 characters, license key in incorrect format.

401 Possible reasons: The provided license key is not valid, the provided license key has expired, you have reached your quota capacity for this account, this account has been disabled.

50x An error occurred on the server. Possible reasons are: license key validation failed or a general server fault.

Sandbox

Testing Sandbox

A sandbox environment is available to assist customers with testing, evaluation and integration. The sandbox url is:

```
https://api1.27hub.com/api/emh/a/v2/sandbox
```

There is no charge for use and your live quota is not affected. No emails are verified in the sandbox and responses are hard coded.

For a full list of hard coded test cases, please see here.

Features

Features

> 99.9% Service Availability

Fully load balanced and automatic fail-over systems dispersed across multiple data centers in multiple regions deliver enterprise grade resilience.

See *Service Reliability* for more information on availability and *SLA*.

Fanatical Service Quality Management (SQM)

Email Hippo operational staff obsessively monitor services to ensure the best possible uptime and coverage.

Uptime and functional correctness is actively monitored on a minute by minute basis from multiple data centers dispersed across North America, Europe and Asia.

Fast, Transparent Response Times

Every query response includes stopwatch data that shows the time taken to execute the request.

Multi Factor Verification

Progressive verification using multiple verification processes including:

- Syntax checking
- DNS checking
- Mailbox checking

Unrivalled Coverage

With more than 5 years experience and the benefit of owning our own software stack, Email Hippo has evolved its services to provide good coverage not only of the easier *B2B* domains but also the more technically challenging *B2C* domains including:

- Hotmail
- Yahoo
- AOL
- Yandex

Spam Trap Detection

After many years R&D, Email Hippo has developed technology that can effectively identify any probable *Spam Trap*.

Disposable Email Address Detection

Identify *DEA* mail boxes.

Unrivalled Performance

Strategic data centers in North America, Europe and Asia, aggressive caching and cloud based auto-scaling deliver outstanding performance. Typical queries are answered between 0.2 to 1.5 seconds.

Note: See *Technical Specification*

On Screen Reporting

Every account comes with a secure online portal for customers to view their current and historic usage via simple but powerful, user friendly charts and reports.

Thoughtful Versioning

Endpoints are "versioned". This means that Email Hippo can continue to release new functionality without "breaking" existing clients committed to integrating with our systems on legacy endpoints.

What it does

Email Hippo is used to check email addresses in real-time. Not only are syntax and domain checked, but that the user mailbox is available too. This is the only way to know for sure if an email address is valid.

Additionally identified as part of the email verification process is extra information including:

- *DEA*.
- *Spam Trap*.

How it works

Email addresses are verified using various filters and processes. As a high level overview, an email address submitted for verification goes thorough the following filters:

Syntax A basic inspection of the syntax of the email address to see if it looks valid. Work is done only using server CPU (Central Processing Unit) based on simple pattern matching algorithms.

DNS A Verifies a domain exists in *DNS*. Domains that do not exist in *DNS* cannot have mail servers or email boxes.

DNS checks are performed over the network.

DNS MX Verify *MX* records using *DNS*. Domains that do not have *MX* records, have no mail servers and therefore no valid email boxes.

MX checks are performed over the network.

MailBox Verify email boxes with *SMTP* checks.

Connect to mail server and perform *SMTP* protocol to verify if mailbox exists.

This is the deepest level of verification. It is performed over the network.

Reliability

Service Reliability

Reliability of your systems is important to you and your clients. You can be sure that we won't let you down when you use our services in your business applications.

By using the latest, distributed cloud based systems, we give deliver fast response times together with enterprise grade uptime of more than 99.9%.

About Our Infrastructure

We operate five data centers geographically dispersed as follows:

- Europe (Netherlands)
- United Kingdom (Ireland)
- United Kingdom (London)
- North America (Oregon)
- Asia (Singapore)

Data centers are located to provide you with the best response times as well as providing automatic fail over to another region in the event of a failure in one or more of the data centers.

Your closest data center is automatically selected based on your *IP Address*. In the event of a fault in your closest data center, requests are handled by the next closest installation.

Service Level Agreement

Our *API* has a stated *SLA* that ensures that we provide you with more than 99.9% uptime for our services.

Download our full Service Level Agreement for further infomation.

Real Time Monitoring

We use a third party service to monitor all of our endpoints for availability, function and response times.

Live Uptime Report

Report shows functional requests. Functional requests are queries containing real email addresses for validation.

Live Response Time Report

Report shows response times from the functional *API* endpoint.

Full Monitoring Statistics

See our Pingdom site for more information.

Technical Specification

Technical Specification

Infrastructure

Manufacturer	emailhippo.com
Uptime	> 99.9%
Response time	>0.2seconds < 8 seconds. Typical response time 0.7 seconds.
Throughput and concurrency	> 100 TPS (Transactions Per Second)
Integration	RESTful GET over HTTP(S)
Authentication	License key
Infrastructure	Geographically dispersed cloud data centers, auto load balance / failover

Application

SLA included	yes
Syntax checking?	yes
DNS A checking?	yes
DNS MX checking	yes
Role Address checking	yes
Free Mail detection	yes
DEA detection	yes
Geo-location detection	yes
Execution timer	yes
Mailbox checking	yes
Enhanced mailbox detection reason codes	yes
Reporting / charts?	yes
Versioning supported?	yes
B2B coverage?	yes
Hotmail coverage?	yes
Yahoo coverage?	yes
AOL coverage?	yes
Yandex coverage?	yes
Secure?	yes. HTTPS supported.
Spam trap detection?	partial
Reporting charts	yes
Server to server supported?	yes

FAQs

Frequently Asked Questions

How can I get a key?

Click here to signup.

How do I call the API?

Make a simple GET request to the endpoint. For example, to query email address *john.doe@gmail.com* with license key *ABCD1234* call:

```
https://api1.27hub.com/api/emh/a/v2?k=ABCD1234&e=john.doe@gmail.com
```

What comes back from the API?

A *JSON* based response similar to:

```
{
        "result": "Bad",
        "reason": "MailboxDoesNotExist",
        "role": false,
        "free": true,
        "disposable": false,
```

```
        "email": "john.doe@gmail.com",
        "domain": "gmail.com",
        "user": "john.doe",
        "mailServerLocation": "US",
        "duration": 723
}
```

Note: For a detailed explanation of the response, see *Data Dictionary For API V2*.

How reliable is the API?

> 99.9% average availability with a defined *SLA*. See *Service Reliability*

Does the system get slower when it's busy?

No. All infrastructure is hosted in cloud based platforms with automatic scaling enabled. Automatic scaling kicks in at busy times to provide more hardware resources to meet demand.

Can it do Hotmail?

Yes.

Can it do Yahoo?

Yes.

Can it find spam traps?

Partially.

A *Spam Trap* is a moving target. In theory (and indeed in practice) anyone can setup a *Block List* and start putting spam traps into the wild.

Email Hippo has *Spam Trap* detection capabilities that covers several of the well known block lists. Whilst it is not possible to deliver 100% coverage of all spam traps from all block lists, Email Hippo provides the best *Spam Trap* detection capabilities available.

How does it work?

At a basic conceptual level, the process of verifying email addresses is very simple. Google for "Send email using telnet" for a quick and general overview of how it's done. To verify an email address without sending an email, simply go as far as the "RCPT TO" stage and parse the response code. That's the easy bit and can be accomplished in just a couple of dozen lines of a PHP script!

The hard bit is dealing with mail services that are intrinsically configured to work against the process of email verification or any similar SMTP based activity. The reason that any email / *SMTP* process is difficult from a client perspective is that mail services need to protect themselves from an ever increasing landscape of abuse including spam and *DDoS* attacks.

Email Hippo's strength in dealing with the "hard bit" of email verification comes from years of experience in doing email verification together with our complete ownership of our *SMTP* verification software stack together with an extensive cloud based infrastructure. That's why Email Hippo can do the "hard bits" best and offer outstanding coverage on the more difficult domains such as Yahoo and Hotmail.

Can I get blacklisted using this API?

No. It's Email Hippo infrastructure that does the work.

Will anyone know that I am verifying their email address?

No. It's Email Hippo infrastructure that does the work.

Your service says an address is OK and I know it's Bad (or vice versa)?

Email Hippo queries mail servers in real time. Mail servers respond with one of two possible answers for a given email address:

- Yes, the email address exists - SMTP code 2xx
- No, the email address does not exist - SMTP code 5xx

Email Hippo uses the above response codes to determine if an email address is valid or not and reports this back to you.

This method of determining email address validity works in >99% cases. However, nothing is guaranteed. In a small number of cases it is possible for a mail server to report one thing on email verification and do something different on trying to deliver an email to the email address verified.

At the time of verification the mail server would have reported Yes/No, however this may have been due to an error within the target mail server and the opposite may have been true. This is rare, but it can happen. If this was a temporary error within the target mail server, please note that this result may be remembered by our system for a few hours.

For another example, say we take an email address of "this.seems.to.verify@hotmail.com" to send to. We are sending from a fictitious email address "my.sending.account@gmail.com".

"this.seems.to.verify@hotmail.com" reports with status code of "OK" from the email verification *API*. However, when you send an email to "this.seems.to.verify@hotmail.com", the email bounces. Further inspection of the bounced email Non Delivery Report (NDR) headers show something like the following:

```
Delivered-To: my.sending.account@gmail.com
Received: by 10.107.174.134 with SMTP id n6csp24867ioo;
                Sat, 6 Jun 2014 03:57:29 -0800 (PST)
X-Received: by 10.202.4.5 with SMTP id 5mr1335105oie.22.1417867048986;
                Sat, 06 Jun 2014 03:57:28 -0800 (PST)
Return-Path: <>
Received: from SNT004-OMC2S34.hotmail.com (snt004-omc2s34.hotmail.com. [65.55.90.109])
                by mx.google.com with ESMTPS id ws5si21632759obb.102.2014.12.06.03.57.
↪28
                for <my.sending.account@gmail.com>
                (version=TLSv1.2 cipher=ECDHE-RSA-AES128-SHA bits=128/128);
                Fri, 6 Jun 2014 03:57:28 -0800 (PST)
Received-SPF: none (google.com: SNT004-OMC2S34.hotmail.com does not designate␣
↪permitted sender hosts) client-ip=65.55.90.109;
Authentication-Results: mx.google.com;
           spf=none (google.com: SNT004-OMC2S34.hotmail.com does not designate␣
↪permitted sender hosts) smtp.mail=
```

```
Received: from SNT004-MC2F40.hotmail.com ([65.55.90.73]) by SNT004-OMC2S34.hotmail.
↪com over TLS secured channel with Microsoft SMTPSVC(7.5.7601.22751);
         Fri, 6 Jun 2014 03:57:28 -0800
From: postmaster@hotmail.com
To: my.sending.account@gmail.com
Date: Fri, 6 Jun 2014 03:57:28 -0800
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status;
        boundary="9B095B5ADSN=_01D010AABCE2C5CC0008C930SNT004?MC2F40.ho"
X-DSNContext: 335a7efd - 4481 - 00000001 - 80040546
Message-ID: <mjZ7zgTpi00029250@SNT004-MC2F40.hotmail.com>
Subject: Delivery Status Notification (Failure)
Return-Path: <>
X-OriginalArrivalTime: 06 Jun 2014 11:57:28.0142 (UTC) FILETIME=[CEAD2EE0:01D0114B]

This is a MIME-formatted message.
Portions of this message may be unreadable without a MIME-capable mail program.

--9B095B5ADSN=_01D010AABCE2C5CC0008C930SNT004?MC2F40.ho
Content-Type: text/plain; charset=unicode-1-1-utf-7

This is an automatically generated Delivery Status Notification.

Delivery to the following recipients failed.

           this.seems.to.verify@hotmail.com
```

The email header of the *NDR* shows that Hotmail thinks the email address is invalid as far as sending to this address is concerned. However, Hotmail reports that the same email address is valid as far as the email verification activity performed by Email Hippo.

The discrepancy in verification results versus mail send is with the Hotmail infrastructure reporting one thing but doing the exact opposite. This behaviour occasionally (particularly from Hotmail) is seen in a small amount of cases and is attributable to internal Hotmail (or other mail services) system anomalies.

The majority (>99%) of email verification status versus mail send is consistent. However there are some edge cases caused by system faults in the mail service providers themselves. For these small number of cases, there is nothing that can be done at the email verification stage.

Glossary

Glossary

ACL Access Control List.

An ACL is checked against the domain calling the *API*. ACL authentication does not require a license key and is best in situations where the *API* is being called from client side script such as JavaScript, jQuery, Angular etc.

API Application Programmers Interface.

See Wikipedia - API Definition for more information.

B2B Business To(2) Business

Business email hosting services are generally private, enterprise grade hosting services typically hosted in either private data centers or in cloud based infrastructure.

Business to business refers to the activity of businesses sending email to clients using business email addresses.

B2C Business To(2) Consumer

Consumer email hosting providers are generally well known, mostly web based providers such as Hotmail, Yahoo, AOL, Gmail etc.

Business to consumer refers to the activity of businesses sending email to clients using consumer email addresses.

Verifying email addresses in consumer domains is generally more technically challenging than *B2B*

Block List See *DNSBL*.

CORS Cross Origin Resource Scripting

Allows modern browsers to work with script (e.g. JavaScript) and *JSON* data originating form other domains.

CORS is required to allow client script such a JavaScript, jQuery or AngularJS to work with results returned from an external *RESTful API*.

See Wikipedia - CORS for more information.

DDoS Distributed Denial of Service

See Wikipedia - Denial-of-service attack for more information.

DEA Disposable Email Address

There are many services available that permit users to use a one-time only email address. Typically, these email addresses are used by individuals wishing to gain access to content or services requiring registration of email addresses but same individuals not wishing to divulge their true identities (e.g. permanent email addresses).

DEA addresses should not be regarded as valid for email send purposes as it is unlikely that messages sent to DEA addresses will ever be read.

DNS Domain Name System

At its simplest level, DNS converts text based queries (e.g. a domain name) into IP addresses.

DNS is also responsible for providing the *MX* records needed to locate a domains mail servers.

See Wikipedia - Domain Name System for more information.

DNSBL DNS Block List

As an anti-spam measure, mail servers can use spam black lists to 'look up' the reputation of IP addresses and domains sending email. If an IP or domain is on a block list, the mail server may reject the senders email message.

See Wikipedia - DNSBL for more information.

ESP Email Service Provider

A service that sends emails on your behalf.

See Wikipedia - Email service provider (marketing) for more information.

Free Mail Addresses served by popular *B2C* service providers such as Hotmail, Yahoo, Live, AOL, Gmail and so on.

Grey Listing A technique used in mail servers as an anti-spam technique. Sometimes also known as "deferred", grey listing arbitrarily delays the delivery of emails with a "try again later" response to the client sending the email.

See Wikipedia - Grey Listing for more information.

HTTP Hypertext Transfer Protocol

See Wikipedia - Hypertext Transfer Protocol for more information.

IP Address **Internet Protocol** Address

See Wikipedia - IP Address for more information.

ISO 3166 International standard for country codes.

See Country Codes - ISO 3166 for more information.

JSON **JavaScript Object Notation**

JavaScript Object Notation, is an open standard format that uses human readable text to transmit data objects consisting of attribute value pairs. It is used primarily to transmit data between a server and web application, as an efficient, modern alternative to XML.

See Wikipedia - JSON for more information.

License Key License key authentication is best for situations where simplicity is required and you can keep the key private. An ideal use case for key authentication would be for server based applications calling the RESTful *API*.

Click here to request a license key.

ms Milliseconds.

MX **Mail Exchanger**

The MX is a server responsible for email interchange with a client.

NDR **Non Delivery Report**

A message that is returned to sender stating that delivery of an email address was not possible.

See Wikipedia - Bounce message for more information.

Office 365 Office 365 mail servers (e.g. x-com.mail.protection.outlook.com) are always configured with the catch all policy, accepting all emails sent to the domain and redirecting them to a central email box for manual inspection. Catch all configured servers cannot respond to requests for email address verification.

This does not affect our coverage of Hotmail, Live and Outlook mailboxes.

Punycode Punycode is a way to represent Unicode with the limited character subset of ASCII supported by the Domain Name System.

See Wikipedia - Punycode for more information.

RESTful **Representational state transfer**

See Wikipedia - RESTful for further information.

Role Address A role address is a generic mailbox such as info@<domain>, sales@<domain> used by organizations to manage email messages of similar organizational types. For example, email messages sent to sales@<domain> can be routed to an organizations sales team where a team of sales people can deal with enquiries.

Role addresses allow collaborative working based on groups rathert than indiviidual mailboxes.

SLA **Service Level Agreement**

See Wikipedia - SLA for more information and description of SLA.

See our Service Level Agreement.

SMTP **Simple Mail Transport Protocol**

SMTP is a protocol. It is the sequence of commands and responses between a client (the software sending an email) and server (the software receiving an email) that facilitates the sending and receiving of email between computer based email messaging systems.

Spam Trap Spam traps are email addresses used for the sole purpose of detecting spamming activities.

Spam traps are used by many block lists (*DNSBL*) to detect spammers.

For more information, see Wikipedia - Spam Traps.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index