Email Address Verification API

Release 3.0.12

Email Hippo Ltd.

Jan 28, 2019

Contents

Email Hippo 'More' *API* services combine traditional email address verification with next generation data enrichment services.

This document will show you how to get up and running with the the service. 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 *API* using a wide range of options for protocols, platforms and technologies.

Engage:

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

CHAPTER 1

Quick start

1.1 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:

1.1.1 1) Create account

Create your account.

1.1.2 2) Login

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

1.1.3 3) Try it

Plug your license key into the following

```
https://api.hippoapi.com/v3/more/json/INSERTYOURLICENSEKEY/john.doe@gmail.com
```

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

```
{
          "version": {
                "v": "More-(0.8.57)",
                "doc": null
          },
          "meta": {
                "lastModified": "Tue, 16 May 2017 11:13:34 GMT",
```

(continues on next page)

(continued from previous page)


```
                "expires": "Thu, 15 Jun 2017 11:13:34 GMT",
                "email": "john.doe@gmail.com",
                "tld": "com",
                "domain": "gmail.com",
                "subDomain": null,
                "user": "john.doe",
                "emailHashMd5": "e13743a7f1db7f4246badd6fd6ff54ff",
                "emailHashSha1": "d3b8f1645736029ea172b312cd995cb8aea9736a",
                "emailHashSha256":
↪"375320dd9ae7ed408002f3768e16cb5f28c861062fd50dff9a3bff62e9dce4ef"
          },
          "disposition": {
                "isRole": false,
                "isFreeMail": true
          },
          "emailVerification": {
                "syntaxVerification": {
                  "isSyntaxValid": true,
                  "reason": "Success"
                },
                "dnsVerification": {
                  "isDomainHasDnsRecord": true,
                  "isDomainHasMxRecords": true,
                  "recordRoot": {
                        "ipAddresses": [
                          "172.217.23.37"
                        ]
                  },
                  "recordWww": {
                        "ipAddresses": [
                          "172.217.23.37"
                        ]
                  },
                  "mxRecords": [
                        {
                          "preference": 40,
                          "exchange": "alt4.gmail-smtp-in.l.google.com",
                          "ipAddresses": [
                                "74.125.28.27"
                          ]
                        },
                        {
                          "preference": 20,
                          "exchange": "alt2.gmail-smtp-in.l.google.com",
                          "ipAddresses": [
                                "74.125.24.27"
                          ]
                        },
                        {
                          "preference": 5,
                          "exchange": "gmail-smtp-in.l.google.com",
                          "ipAddresses": [
                                "64.233.167.27"
                          ]
                        },
                        {
                          "preference": 30,
```


(continues on next page)

(continued from previous page)


```
                        "exchange": "alt3.gmail-smtp-in.l.google.com",
                        "ipAddresses": [
                              "108.177.97.27"
                        ]
                      },
                      {
                        "preference": 10,
                        "exchange": "alt1.gmail-smtp-in.l.google.com",
                        "ipAddresses": [
                              "74.125.131.27"
                        ]
                      }
                ],
                "txtRecords": [
                      "\"v=spf1 redirect=_spf.google.com\""
                ]
              },
              "mailboxVerification": {
                "result": "Bad",
                "reason": "MailboxDoesNotExist"
              }
        },
        "infrastructure": {
              "mail": {
                "serviceTypeId": "Gmail",
                "mailServerLocation": "US",
                "smtpBanner": "220 mx.google.com ESMTP 74si2054451wmf.101 - gsmtp"
              },
              "web": {
                "hasAliveWebServer": true
              }
        },
        "sendAssess": {
              "inboxQualityScore": 0.1,
              "sendRecommendation": "DoNotSend"
        },
        "spamAssess": {
              "isDisposableEmailAddress": false,
              "isDarkWebEmailAddress": false,
              "isGibberishDomain": false,
              "isGibberishUser": false,
              "domainRiskScore": 3.0,
              "formatRiskScore": 0.0,
              "profanityRiskScore": 0.0,
              "overallRiskScore": 0.8,
              "actionRecomendation": "Allow",
              "blockLists": [
                {
                      "blockListName": "spamhaus",
                      "isListed": false,
                      "listedReason": null,
                      "listedMoreInfo": null
                }
              ]
        },
        "spamTrapAssess": {
              "isSpamTrap": false,
```


(continues on next page)

(continued from previous page)


```
                "spamTrapDescriptor": null
          },
          "hippoTrust": {
                "score": 0.1,
                "level": "Low"
          },
          "social": {
                "gravatar": {
                  "imageUrl": "//www.gravatar.com/avatar/
↪e13743a7f1db7f4246badd6fd6ff54ff",
                  "profileUrl": "//www.gravatar.com/e13743a7f1db7f4246badd6fd6ff54ff"
                }
          },
          "domain": null,
          "performance": {
                "syntaxCheck": 0,
                "dnsLookup": 138,
                "spamAssessment": 0,
                "mailboxVerification": 292,
                "webInfrastructurePing": 0,
                "other": 0,
                "overallExecutionTime": 430
          },
          "diagnostic": {
                "key": "e6298826-d257-432f-a893-08af776206bf"
          }
        }
```


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!

CHAPTER 2

Compliance

2.1 Compliance

We're BIG on security.

If you are using the Email Hippo More API you need to be confident that your data is safe and that the processing is secure. Email Hippo security processes are accredited to the ISO 27001 standard; the highest independently assessed standard that is internationally recognised.

In addition to our stringent security processes, it is relevant to note: * All data in transit is encrypted using HTTPS. * All data at rest (e.g. stored for caching and reporting purposes) is secured using AES-265 bit encryption.

If you have any questions regarding data security, our processing terms or any compliance related question please contact our Data Protection Officer; DPO@emailhippo.com

Links to our documents:

- Privacy policy
- Cookie policy
- Terms of service
- Data processing terms

CHAPTER 3

Latest uptime statistics

CHAPTER 4

Live uptime report

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

CHAPTER 5

Live response time report

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

CHAPTER 6

Editions

6.1 About editions

There are three editions of endpoints.

- Basic
- Block lists
- More

Each varies in functionality and performance.

The schema across all editions remains consistent which delivers the following benefits:

- Consistent integration with a consistent entity model
- Easily change between editions based on data depth versus performance requirements.

6.1.1 Basic

Basic level performs simple, compute only email address syntax checks.

This is the fastest performing end point.

Performance: Fastest

6.1.2 Block lists

Performs basic level checks plus:

- DNS lookups
- Checking of email infrastructure against Email Hippo and third party lists for *DEA* and spam or other anti-social behavior.

Performance: Medium

6.1.3 More

The most thorough analysis and data enrichment.

Performs basic and block lists levels plus:

- Deep mail box verification
- Web site PING
- Social enrichment
- Spam scoring
- *Spam Trap* analysis
- Send Scoring
- Hippo Trust Scoring

Performance: Least fast

For more information on performance and features see Endpoint details.

CHAPTER 7

Integration guide

7.1 Schema

- Endpoint definitions
- WADL (swagger.io)

7.2 Return protocols

Email Hippo *API* services can return data in several formats:

- *JSON*
- *XML*
- *BSON*
- *protobuf*

7.3 Firewall rules

If your organization implements internal firewall rule policies, you may need to ask your IT staff to allow access to our API endpoints.

Our *API* services are delivered via Cloudflare. Please see the Cloudflare page "IP ranges" for a definition of the IP endpoints that are possible when accessing our *API*.

CHAPTER 8

Features

8.1 Features

8.1.1 Confidence in data security

With ISO27001:2013 certification, robust technology and clearly defined policies and procedures, you can trust Email Hippo with your data.

See *Compliance* for more information.

8.1.2 > 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*.

8.1.3 Multiple response formats

Since Version 3 of Email Hippo's services, it has set the bar higher for email address verification integration. Whilst most of Email Hippo's competitors only offer *JSON*, Email Hippo goes further with giving our customers more protocol options:

- *JSON* (industry standard modern text based interchange.)
- *XML* (industry standard legacy text based interchange. Great for interop with older systems.)
- *BSON* (industry standard binary based interchange. Ideal for direct storage in mongoDB.)
- *protobuf* (Google standard for binary based interchange. Ideal for applications requiring low bandwidth and high performance.)

8.1.4 Easy integration

See *Client libraries* to see how quick and easy it is to integrate with Email Hippo's services from over 19 different technologies and platforms.

8.1.5 Fanatical service quality management (SQM)

Email Hippo's 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.

8.1.6 Fast, transparent response times

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

8.1.7 Proprietary scoring and risk assessment

- Send risk assessment scoring based on Email Hippo proprietary scoring heuristics (new)
- Spam assessment and block-list risk scoring based on Email Hippo rules and 3rd party data sources including SpamHaus (new)
- Overall risk scoring based on Email Hippo assessment of Send Risk combined with spam assessment (new)

8.1.8 Multi factor verification and data enrichment

Progressive verification using multiple verification processes including:

- Syntax checking
- DNS checking
- Block-list checking (e.g. spamhaus)
- Web infrastructure checking
- Mailbox checking
- Proprietary risk scoring including assessment of risks for receiving email from (spam), sending email to (send score) and overall risk assesment.

8.1.9 Unrivalled coverage

Email Hippo leverages the advantages of its scalable infrastructure to provide coverage of domains that are technically challenging. Consumer facing domains tend to be more challenging to cover then business facing domains *B2C* domains including:

- Hotmail
- Yahoo
- Office 365
- AOL
- Yandex

8.1.10 Spam trap detection

Email Hippo has developed technology that can effectively identify any probable *Spam Trap*.

8.1.11 Disposable email address detection

Advanced disposable email address detection based on 'Email Hippo'_'s multi-vector real-time analysis.

Features include:

- Checking against static lists
- Real-time detection of common *DEA* providers obfuscation techniques (e.g. rotating domains, IP addresses and MX servers)

8.1.12 Gibberish detection

A common vector for persons wishing to remain anonymous is to register or use a pre-existing domain. Finding an available domain is not easy and as such, many opt for a 'Gibberish' domain such as 'sdfre45321qaxc.com'.

Email Hippo detects gibberish in both the user and domain elements of an email address.

8.1.13 Unrivalled performance

Strategic data centers in Europe, aggressive caching, global network delivery optimization and cloud based auto-scaling deliver outstanding performance. Typical queries are answered between 0.2 to 1.0 seconds.

Note: See *Technical specification*

8.1.14 On screen reporting

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

8.1.15 Thoughtful versioning

Endpoints are "versioned". This means that Email Hippo can continue to release new functionality without "breaking" existing client integrations which use legacy endpoints.

8.1.16 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* Disposable Email Address.
- *Spam Trap*.

8.1.17 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.

CHAPTER 9

Usage report

9.1 Usage report

Customers can access their current usage in real-time by accessing an easy to use, convenient *RESTful* endpoint.

9.1.1 Caveats and limitations

- Reports only on API usage. File based uploads are not included.

9.1.2 Getting started

Please see the detailed guide in the **QuotaReporting** section in the Endpoint definitions definitions.

CHAPTER 10

Reliability

10.1 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%.

10.1.1 About our infrastructure

We operate three data centers geographically dispersed as follows:

- Europe (Netherlands)
- United Kingdom (Ireland)
- United Kingdom (London)

Data centers provide automatic fail over to another working data center.

Network traffic is optimized for fast, reliable global delivery using Cloudflare Argo.

10.1.2 Service level agreement

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

Read our terms of service for more information - Terms of Service.

10.2 Real time monitoring

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

10.2.1 Live uptime report

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

10.2.2 Live response time report

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

10.2.3 Full monitoring statistics

See our Pingdom site for more information.

CHAPTER 11

Concurrency

11.1 Concurrency

To preserve the operational integrity of the service to all of our customers, there is a maximum concurrency enforced by our systems.

11.1.1 Limits

Allowed throughput is **140 email verifications per second**.

Throughput exceeding these limits will receive HTTP response code 429 (too many request) for subsequent requests for a duration of one minute.

11.1.2 Suggestions on how to manage throughput

There are several things that it may be helpful to think about to control throughput so as not to exceed the maximum limits described above such as:

- Test your integration with representative production loads over a period of time. Monitor response codes for any 429's. If you see any 429's please reduce the rate at which your application is querying our servers.
- For applications that can tolerate slight delays in your data processing mesh, consider using queuing infrastructure with a rate controllable processor. Your 'processor' can then schedule picking work of of the queue and submitting requests to our systems at a controllable rate.

11.1.3 Large throughput requirements

For sustained throughput more than **50 email verifications per second**, please contact us for options on private, dedicated service.

CHAPTER 12

Data dictionary

12.1 Data dictionary for API v3

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.

12.1.1 Open source common entities

Since v3 (code named 'More'), all common entities are avaialable on GitHub. Email Hippo uses these entities internally and exposes same over all of our v3 endpoints.

For a full definition of all of our entities and types, see the GitHub repository @ entity defintitions.

12.1.2 Related information

- For a full swagger.io based definition see the Endpoint Definitions
- View the full swagger.io compatible WADL

12.1.3 Response body content

Reponses are complex types. The table below shows the root fields.

Notes:

- Click the 'Field Name' entry for a link to the GitHub class library for the type
- Click the 'Description' link for further descriptive documentation on the field

Field Name	Type	Description	Example Data
version	version	*Version Information*	see demo
meta	meta	*Meta Information*	see demo
disposition	disposition	*Disposition Information*	see demo
emailVerification	emailVerification	*Email verification. Syntax, DNS, mailbox*	see demo
infrastructure	infrastructure	*Infrastructure details for domain web and mail*	see demo
sendAssess	sendAssess	*Send Recommendation*	see demo
spamAssess	spamAssess	*Spam Assessment*	see demo
spamTrapAssess	spamTrapAssess	*Spam Trap Assessment*	see demo
hippoTrust	hippoTrust	**'Hippo Trust'_**	2.0
social	social	*Social Information*.	see demo
domain	domain	For future use.	null
performance	performance	*Performance Information*.	see demo
diagnostic	diagnostic	Diagnostic key. Future use.	see demo

Version information

Type info: version

Contains details of the version and edition of API and a URL to the documentation.

Example:

```
"version": {
 "v": "More-(0.8.57)",
 "doc": null
}
```

Meta information

Type info: meta

Field Name	Type	Description	Example Data
lastModified	string	Last modified date/time of Email Hippo record	"Sat, 20 May 2017 12:13:36 GMT"
expires	string	Date/time that this record expires from Email Hippo cache	"Mon, 19 Jun 2017 12:13:36 GMT"
email	string	The email being queried	"abuse@hotmail.com.br"
tld	string	The Top Level Domain (TLD) of email being queried	"com.br"
domain	string	The domain of the email being queried	"hotmail.com.br"
subDomain	string	The sub domain (if any) of the email being queried	null
user	string	The user element of the email address	"abuse"
emailHashMd5	string	MD5 hash of the email address	"87da0257051ef17dd5580118ac2724f0"
emailHashSha1	string	SHA1 hash of the email address	"c1a6e8994311d2fbe3add4c7168be86f23dab452"
emailHashSha256	string	SHA265 hash of the email address	"29bf2669bc8ebc263eec23ed7859cb250352b9818471f2bc54b20f7e2f

Example:


```
"meta": {
 "lastModified": "Sat, 20 May 2017 12:13:36 GMT",
 "expires": "Mon, 19 Jun 2017 12:13:36 GMT",
 "email": "abuse@hotmail.com.br",
 "tld": "com.br",
 "domain": "hotmail.com.br",
 "subDomain": null,
 "user": "abuse",
 "emailHashMd5": "87da0257051ef17dd5580118ac2724f0",
 "emailHashSha1": "c1a6e8994311d2fbe3add4c7168be86f23dab452",
 "emailHashSha256": "29bf2669bc8ebc263eec23ed7859cb250352b9818471f2bc54b20f7e2f3b28c8"
}
```


Disposition information

Type info: disposition

Field Name	Type	Description	Example Data
isRole	boolean	Is a role address? (e.g. info@, sales@, postmaster@	true
isFreeMail	boolean	Is a free mail provider? (e.g. hotmail, aol etc)	true

Example:


```
"disposition": {
 "isRole": true,
 "isFreeMail": true
}
```


Email verification. Syntax, DNS, mailbox

Type info: emailVerification

Field Name	Type	Description	Example Data
syntaxVerification	syntaxVerification	*Syntax Verification* to RFC821	see example
dnsVerification	dnsVerification	*DNS Verification*	see example
mailboxVerification	mailboxVerification	*Mailbox Verification*	see example

Example:


```
"emailVerification": {
 "syntaxVerification": {
 "isSyntaxValid": true,
 "reason": "Success"
},
 "dnsVerification": {
  "isDomainHasDnsRecord": true,
  "isDomainHasMxRecords": true,
  "recordRoot": {
  "ipAddresses": [
  "65.55.118.92",
```


(continues on next page)

(continued from previous page)


```
 "157.56.198.220"
 ]
},
"recordWww": {
"ipAddresses": [
"157.56.198.220"
]
},
"mxRecords": [
 {
  "preference": 5,
  "exchange": "mx1.hotmail.com",
  "ipAddresses": [
   "65.55.33.135",
   "104.44.194.236",
   "104.44.194.237",
   "104.44.194.235",
   "65.54.188.72",
   "65.54.188.126",
   "104.44.194.234",
   "65.55.37.88",
   "65.55.37.104",
   "104.44.194.233",
  "65.55.37.72",
  "65.55.92.184",
  "65.55.92.168",
  "207.46.8.167",
  "65.55.92.136",
  "104.44.194.232",
  "65.55.33.119",
  "104.44.194.231"
 ]
},
{
 "preference": 5,
 "exchange": "mx2.hotmail.com",
 "ipAddresses": [
  "104.44.194.235",
  "65.55.92.136",
  "65.54.188.94",
  "65.55.37.88",
  "207.46.8.167",
  "65.55.37.120",
  "104.44.194.237",
  "104.44.194.234",
  "104.44.194.236",
  "65.55.92.184",
  "104.44.194.233",
  "65.54.188.126",
  "104.44.194.231",
  "207.46.8.199",
  "104.44.194.232",
  "65.55.92.152",
  "65.55.37.104",
  "65.55.33.135",
  "65.54.188.72"
 ]
```


(continues on next page)

(continued from previous page)


```
 },
 {
  "preference": 5,
  "exchange": "mx3.hotmail.com",
  "ipAddresses": [
   "65.55.37.120",
   "65.55.92.136",
   "65.55.92.152",
   "104.44.194.234",
   "65.55.33.119",
   "65.55.92.168",
   "104.44.194.232",
   "65.55.37.72",
   "104.44.194.235",
   "104.44.194.236",
   "65.54.188.94",
   "65.54.188.110",
   "207.46.8.167",
   "104.44.194.237",
   "104.44.194.231",
   "65.55.37.104",
   "104.44.194.233",
   "65.54.188.72",
   "207.46.8.199"
  ]
 },
 {
  "preference": 5,
  "exchange": "mx4.hotmail.com",
  "ipAddresses": [
   "65.55.37.120",
   "65.54.188.110",
   "104.44.194.235",
   "104.44.194.232",
   "65.55.92.168",
   "207.46.8.199",
   "65.54.188.94",
   "65.55.92.152",
   "104.44.194.237",
   "65.55.33.135",
   "65.55.37.88",
   "104.44.194.234",
   "65.55.92.184",
   "104.44.194.233",
   "104.44.194.231",
   "65.55.37.72",
   "104.44.194.236",
   "65.55.33.119"
  ]
 }
],
"txtRecords": [
 "\"v=spf1 include:spf-a.hotmail.com include:spf-b.hotmail.com include:spf-c.hotmail.
↪com include:spf-d.hotmail.com ~all\""
 ]
},
"mailboxVerification": {
```


(continues on next page)

(continued from previous page)

```
 "result": "Bad",
 "reason": "MailboxDoesNotExist"
 }
}
```

Syntax verification

Type info: syntaxVerification

Field Name	Type	Description	Example Data
isSyntax-Valid	boolean	Is the syntax of the email address correct according to RFC standards?	true
reason	syntaxReason	*Syntax Verification Reason Codes*	"Success"

Example:

```
"syntaxVerification": {
        "isSyntaxValid": true,
        "reason": "Success"
}
```

Syntax verification reason codes

None No status available.

AtSignNotFound The '@' sign not found.

DomainPartCompliancyFailure The syntax of a legal Internet host name was specified in RFC-952. One aspect of host name syntax is hereby changed: the restriction on the first character is relaxed to allow either a letter or a digit. (http://tools.ietf.org/html/rfc1123#section-2.1)

NB RFC 1123 updates RFC 1035, but this is not currently apparent from reading RFC 1035. Most common applications, including email and the Web, will generally not permit escaped strings (http://tools.ietf.org/html/rfc3696#section-2). The better strategy has now become to make the "at least one period" test, to verify LDH conformance (including verification that the apparent TLD name is not all-numeric)(http://tools.ietf.org/html/rfc3696#section-2) Characters outside the set of alphabetic characters, digits, and hyphen MUST NOT appear in domain name labels for SMTP clients or servers (http://tools.ietf.org/html/rfc5321#section-4.1.2) RFC5321 precludes the use of a trailing dot in a domain name for SMTP purposes (http://tools.ietf.org/html/rfc5321#section-4.1.2)

DoubleDotSequence Can't have empty element (consecutive dots or dots at the start or end)(http://tools.ietf.org/html/rfc5322#section-3.4.1)

InvalidAddressLength Email is too long.

The maximum total length of a reverse-path or forward-path is 256 characters (including the punctuation and element separators) (http://tools.ietf.org/html/rfc5321#section-4.5.3.1.3)

InvalidCharacterInSequence Invalid character in email address.

InvalidEmptyQuotedWord Invalid Empty Quoted Word.

InvalidFoldingWhiteSpaceSequence Folding White Space.

local-part = dot-atom / quoted-string / obs-local-part

obs-local-part = word (“.” word)(http://tools.ietf.org/html/rfc5322#section-3.4.1)

InvalidLocalPartLength Local part must be 64 characters or less.

InvalidWordBoundaryStart RFC5321 section 4.1.3.

Character preceding IPv4 address must be ‘:’. RFC5321 section 4.1.3

Success Syntax verification is successful.

TooManyAtSignsFound Too many @ signs found in email address. Only one is permitted.

UnbalancedCommentParenthesis Unbalanced comment parenthesis

UnexpectedQuotedPairSequence Any ASCII graphic (printing) character other than the at-sign (“@”), backslash, double quote, comma, or square brackets may appear without quoting. If any of that list of excluded characters are to appear, they must be quoted (http://tools.ietf.org/html/rfc3696#section-3)

Any excluded characters? i.e. 0x00-0x20, (,), <, >, [,], :, ;, @, , comma, period, “

Unknown Syntax verification failed for unknown reasons.

UnmatchedQuotedPair Unmatched quoted pair.

DNS verification

Type info: dnsVerification

Field Name	Type	Description	Example Data
isDomainHasDnsRecord	boolean	Does the dmain have any DNS records?	true
isDomainHasMxRecords	boolean	Does the domain have any *MX* records?	true
recordRoot	record	Details of root A record for domain	see example
recordWww	record	Details of records for WWW subdomain	see example
mxRecords	List of mxrecord	All *MX* records for domain	see example
txtRecords	List of string	All *TXT* records for domain	“”v=spf1 include:spf-a.hotmail.com include:spf-b.hotmail.com include:spf-c.hotmail.com include:spf-d.hotmail.com ~all”“

Example:

```
"dnsVerification": {
 "isDomainHasDnsRecord": true,
 "isDomainHasMxRecords": true,
 "recordRoot": {
 "ipAddresses": [
  "65.55.118.92",
  "157.56.198.220"
 ]
```

(continues on next page)

(continued from previous page)


```
},
"recordWww": {
 "ipAddresses": [
 "157.56.198.220"
 ]
},
"mxRecords": [
 {
  "preference": 5,
  "exchange": "mx1.hotmail.com",
  "ipAddresses": [
   "65.55.33.135",
   "104.44.194.236",
   "104.44.194.237",
   "104.44.194.235",
   "65.54.188.72",
   "65.54.188.126",
   "104.44.194.234",
   "65.55.37.88",
   "65.55.37.104",
   "104.44.194.233",
   "65.55.37.72",
   "65.55.92.184",
   "65.55.92.168",
   "207.46.8.167",
   "65.55.92.136",
   "104.44.194.232",
   "65.55.33.119",
   "104.44.194.231"
  ]
 },
 {
 "preference": 5,
 "exchange": "mx2.hotmail.com",
 "ipAddresses": [
  "104.44.194.235",
  "65.55.92.136",
  "65.54.188.94",
  "65.55.37.88",
  "207.46.8.167",
  "65.55.37.120",
  "104.44.194.237",
  "104.44.194.234",
  "104.44.194.236",
  "65.55.92.184",
  "104.44.194.233",
  "65.54.188.126",
  "104.44.194.231",
  "207.46.8.199",
  "104.44.194.232",
  "65.55.92.152",
  "65.55.37.104",
  "65.55.33.135",
  "65.54.188.72"
  ]
 },
 {
```


(continues on next page)

(continued from previous page)


```
 "preference": 5,
 "exchange": "mx3.hotmail.com",
 "ipAddresses": [
  "65.55.37.120",
  "65.55.92.136",
  "65.55.92.152",
  "104.44.194.234",
  "65.55.33.119",
  "65.55.92.168",
  "104.44.194.232",
  "65.55.37.72",
  "104.44.194.235",
  "104.44.194.236",
  "65.54.188.94",
  "65.54.188.110",
  "207.46.8.167",
  "104.44.194.237",
  "104.44.194.231",
  "65.55.37.104",
  "104.44.194.233",
  "65.54.188.72",
  "207.46.8.199"
  ]
 },
 {
 "preference": 5,
 "exchange": "mx4.hotmail.com",
 "ipAddresses": [
  "65.55.37.120",
  "65.54.188.110",
  "104.44.194.235",
  "104.44.194.232",
  "65.55.92.168",
  "207.46.8.199",
  "65.54.188.94",
  "65.55.92.152",
  "104.44.194.237",
  "65.55.33.135",
  "65.55.37.88",
  "104.44.194.234",
  "65.55.92.184",
  "104.44.194.233",
  "104.44.194.231",
  "65.55.37.72",
  "104.44.194.236",
  "65.55.33.119"
  ]
 }
],
"txtRecords": [
 "\"v=spf1 include:spf-a.hotmail.com include:spf-b.hotmail.com include:spf-c.hotmail.
↪com include:spf-d.hotmail.com ~all\""
 ]
}
```

Mailbox verification

Type info: mailboxVerification

Field Name	Type	Description	Example Data
result	result	*Primary Result Codes*	"Bad"
reason	reason	*Secondary Reason Codes*	"MailboxDoesNotExist"

Example:

```
"mailboxVerification": {
 "result": "Bad",
 "reason": "MailboxDoesNotExist"
}
```

Primary result codes

None No status available.

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 *Secondary Reason Codes*.

Secondary reason codes

Primary result codes with their possible secondary reason codes

Ok Success

Bad AtSignNotFound

DomainIsInexistent

MailboxFull

MailboxDoesNotExist

MailServerFaultDetected

NoMxServersFound

ServerDoesNotSupportInternationalMailboxes

TooManyAtSignsFound

PossibleSpamTrapDetected

RetryLater TransientNetworkFault

Unverifiable None

DomainIsWellKnownDea

GreyListing

ServerIsCatchAll

Unknown

UpredictableSystem

Secondary reason code definitions

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 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.

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.

MailServerFaultDetected An unspecified mail server fault was detected.

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.

UpredictableSystem Upredictable system infrastructure detected.

Various email services deliver unpredictable results to email address verification.

The reason for this unpredictability is that some email systems elect not to implement email standards (i.e. RFC 2821).

For systems that are known to be unpredictable, we return a secondary status of UpredictableSystem.

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*.

Infrastructure details for domain web and mail

Type info: infrastructure

Field Name	Type	Description	Example Data
mail	mailInfrastructure	*Mail Infrastructure*	see example
web	webInfrastructure	*Web Infrastructure*	see example

Example:

```
"infrastructure": {
 "mail": {
  "serviceTypeId": "Hotmail",
  "mailServerLocation": "US",
  "smtpBanner": "220 SNT004-MC9F19.hotmail.com Sending unsolicited commercial or bulk
↪e-mail to Microsoft's computer network is prohibited. Other restrictions are found
↪at http://privacy.microsoft.com/en-us/anti-spam.mspx. Sat, 20 May 2017 05:13:34 -
↪0700"
 },
"web": {
 "hasAliveWebServer": true
 }
}
```

Mail infrastructure

Type info: mailInfrastructure

Field Name	Type	Description	Example Data
serviceTypeId	serviceTypeId	*Service Type Identifier*.	"Hotmail"
mailServerLocation	string	Mail server location. 2 digit ISO code.	"US"
smtpBanner	string	*SMTP* banner received on connect to mail server.	see example

Example:

```
"mail": {
 "serviceTypeId": "Hotmail",
 "mailServerLocation": "US",
 "smtpBanner": "220 SNT004-MC9F19.hotmail.com Sending unsolicited commercial or bulk
↪e-mail to Microsoft's computer network is prohibited. Other restrictions are found
↪at http://privacy.microsoft.com/en-us/anti-spam.mspx. Sat, 20 May 2017 05:13:34 -
↪0700"
}
```

Service type identifier

Type info: serviceTypeId

Other Service not of pre-defined list of known types.

Aol AOL.

Hotmail Hotmail.

Gmail Gmail.

GoogleForBiz Google for business.

MessageLabs Symantec message labs.

Net4Sec Net4Sec.

Office365 Microsoft Office 365.

Yahoo Yahoo.

UceProtect UCE Protect.

Web infrastructure

Type info: webInfrastructure

Email Hippo performs a PING to establish whether a domain has a working web server / web site. A domain without a working website can be an indicator of low quality email domains.

Field Name	Type	Description	Example Data
hasAliveWeb-Server	boolean	Determines if domain has a web server that responds to PING.	true

Example:

```
"web": {
 "hasAliveWebServer": true
 }
```

Send assessment

Type info: sendAssess

Email Hippo performs an assessment of the risk associated with sending email to the email address queried. The overall score is based on a number of factors including:

- If the domain is determined to be a *DEA*
- If the mailbox is verified as 'Ok' or 'Good'
- Whether the email domain is listed in third party lists (e.g. SpamHaus)
- Whether the domain is determined to be FreeMail or is a role address
- Whether the domain has a working web site

Field Name	Type	Description	Example Data
inboxQualityScore	decimal	Inbox quality score.	0.1
sendRecommendation	sendAssesType	*Send recommendation*.	"DoNotSend"

Send recommendation

None No recommendation.

SafeToSend Safe to send email. Minimal risk of hard bounces or complaints.

DoNotSend Do not send. Hight risk of hard bounce and complaints.

RiskyToSend Sending to this email address is risky. Hard bounces and complaints are possible. Send at your own risk.

Example:

```
"sendAssess": {
 "inboxQualityScore": 0.1,
 "sendRecommendation": "DoNotSend"
}
```

Spam assessment

Type info: spamAssess

Email Hippo performs an assesment of the risk associated with receiving email from the address queried. The overall score is based on a number of factors from the table below.

Field Name	Type	Description	Example Data
isDisposableEmailAddress	boolean	Is the email domain a *DEA*?	false
isDarkWebEmailAddress	boolean	Is the email address domain hosted in the Dark Web?	false
isGibberishDomain	boolean	Is the email address domain deemed to be gibberish text?	false
isGibberishUser	boolean	Is the email address user deemed to be gibberish text?	false
domainRiskScore	decimal	General risk score of email address domain.	0
formatRiskScore	decimal	Format risk score of email address.	0
profanityRiskScore	decimal	Profanity risk score of email address.	0
overallRiskScore	decimal	Overall risk score for spam from this email address.	0
actionRecomendation	actionRecommendationType	What action should you take if receiving email from email address.	“Allow”
blockLists	List of blockList	*Blocklists*.	see example

Example:

```
"spamAssess": {
 "isDisposableEmailAddress": false,
 "isDarkWebEmailAddress": false,
 "isGibberishDomain": false,
 "isGibberishUser": false,
 "domainRiskScore": 0,
 "formatRiskScore": 0,
 "profanityRiskScore": 0,
 "overallRiskScore": 0,
 "actionRecomendation": "Allow",
 "blockLists": [
  {
   "blockListName": "spamhaus",
   "isListed": false,
   "listedReason": null,
   "listedMoreInfo": null
  }
 ]
}
```

Blocklists

Type info: List of blockList

Email Hippo includes references to third party spam block lists to enrich it's own email verification information.

Initially (on launch of v3), we include references to SpamHaus Domain Block List (DBL).

Note: Email Hippo may add additional data sources for blocklists in the future.

Field Name	Type	Description	Example Data
blockListName	string	Name of block list.	"spamhaus"
isListed	boolean	Is the email address domain listed in the block list?	true
listedReason	string	If the email address domain is listed in the block list, then why?	"127.0.1.2"
listedMoreInfo	string	Any additional information provided from the block list on reason(s)	"https://www.spamhaus.org/query/domain/dbltest.com"

Example:

```
"blockLists": [
  {
  "blockListName": "spamhaus",
  "isListed": true,
  "listedReason": "127.0.1.2",
  "listedMoreInfo": "https://www.spamhaus.org/query/domain/dbltest.com"
  }
 ]
```

Spam trap assessment

Type info: List of spamTrapAssess

Email Hippo maintains a list of known *Spam Trap*.

Field Name	Type	Description	Example Data
isSpamTrap	boolean	Is this email address a known spam trap?	true
spamTrapDescriptor	string	Description of spam trap.	"uceprotect"

Example:

```
"spamTrapAssess": {
        "isSpamTrap": true,
        "spamTrapDescriptor": "uceprotect"
        }
```

Email Hippo Trust Score

Type info: List of hippoTrust

For email verification and data enrichment performed to the 'More' level, Email Hippo supplies a Trust Score.

About the Email Hippo Trust Score Email Hippo created the Trust Score to provide an 'at a glance' determination of quality from the point of view of drilling deeper than just the email address itself.

Email Hippo Trust Score is designed to answer a fundamental question posed from the perspective of a business owner, merchant, data broker or lean generation service:

How much can I trust the person associated with this email address?

The Trust Score takes dozens of metrics and signals into consideration when making this assesment and providing the final score.

Field Name	Type	Description	Example Data
score	decimal	*How much can I trust the person associated with this email address?*	0.1
level	trustLevel-Type	**'Hippo Trust Level'_**.	"Low"

Trust level

Type info: trustLevelType

Trust Level	Description	Score range
None	No information on trust	
Low	Low trust level	Less than 2.66
Medium	Medium trust level	2.66 to 6.99
High	High trust level	7 to 10

Example:

```
"hippoTrust": {
 "score": 0.1,
 "level": "Low"
}
```

Social information

Type info: social

Email Hippo can provide social data. On initial launch of v3, Gravatar information is supplied.

Field Name	Type	Description	Example Data
social	social	Social information associated with email address	see example

Example:

```
"social": {
 "gravatar": {
  "imageUrl": "//www.gravatar.com/avatar/87da0257051ef17dd5580118ac2724f0",
  "profileUrl": "//www.gravatar.com/87da0257051ef17dd5580118ac2724f0"
 }
}
```

Performance information

Type info: performance

Detailed performance metrics are provided for all queries. All timings are expressed in milliseconds.

Field Name	Type	Description	Example Data
syntaxCheck	integer	Processing time to check syntax of email address.	see example
dnsLookup	integer	Processing time to gather and check *DNS* of email address.	see example
spamAssessment	integer	Processing time to assess email address for spam behavior.	see example
mailboxVerification	integer	Processing time to check mail box of email address.	see example
webInfrastructurePing	integer	Processing time to PING web site of email address.	see example
other	integer	Processing time for miscellaneous processing of email address.	see example
overallExecutionTime	integer	Total processing time.	see example

Example:

```
"performance": {
 "syntaxCheck": 0,
 "dnsLookup": 250,
 "spamAssessment": 0,
 "mailboxVerification": 5348,
 "webInfrastructurePing": 0,
 "other": 0,
 "overallExecutionTime": 5598
}
```

12.1.4 Response header

HTTP status codes

In additional to the application level codes (see *Primary Result Codes* and *Secondary Reason 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.

429 Too many requests. See *Concurrency* for further information.

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

CHAPTER 13

Client libraries

13.1 Client libraries

13.1.1 .NET

Use our high performance .NET client library.

13.1.2 Swagger code generator

Use the Swagger Code Generator with our Swagger Schema to generate API client code for:

- ActionScript
- Bash
- C# (.net 2.0, 4.0 or later)
- C++ (cpprest, Qt5, Tizen)
- Clojure
- Dart
- Elixir
- Go
- Groovy
- Haskell
- Java (Jersey1.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign)
- Node.js (ES5, ES6, AngularJS with Google Closure Compiler annotations) Objective-C
- Perl
- PHP

- Python
- Ruby
- Scala
- Swift (2.x, 3.x)
- Typescript (Angular1.x, Angular2.x, Fetch, jQuery, Node)

CHAPTER 14

Technical specification

14.1 Technical specification

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).
Security and encryption	Transport security using HTTPS. Data at rest encrypted using 256-bit AES encryption.
Integration	RESTful GET over HTTPS, XML GET over HTTPS, BSON over HTTPS, protobuf over HTTPS.
Authentication	License key.
Infrastructure	Geographically dispersed cloud data centers, auto load balance / failover.

CHAPTER 15

Change log

15.1 Change log

15.1.1 V3.4

- Added quota reporting *RESTful* endpoint

15.1.2 V3.0 (codename 'More')

Release date: May 16th, 2017

- Domain, user and sub domain splitting (new)
- Disposition tagging for Free-mail and Role based addresses (new)
- Enhanced syntax validation with reason code
- Enhanced DNS verification returning A, MX and TXT records (new)
- Mailbox verification to SMTP level
- Mail infrastructure identification including SMTP banner (new)
- Mail infrastructure geographic location ISO code
- Web infrastructure PING - detects if domain has alive web server (new)
- Social data / images from Gravatar (new)
- Send risk assessment scoring based on EmailHippo proprietary scoring heuristics (new)
- Spam assesment and blocklist risk scoring based on EmailHippo rules and 3rd party data sources including SpamHaus (new)
- Overall risk scoring based on Email Hippo assesment of Send risk combined with spam assesment (new)
- Gibberish domain and user detection (new)

- Advanced *DEA* detection based on Email Hippo multi-vector realtime analysis (new)
- Detailed diagnostic performance timings (new)
- Moved endpoints to domain to api.hippoapi.com (new)

15.1.3 V2.5

Release date: June, 2016

- Added infrastructure identifier node (new)

15.1.4 V2.0

Release date: November 2015

- Deployed to global, cloud based distributed architecture
- Added mail server location data ISO code
- Domain and user information
- *DEA* detection based on static lists

15.1.5 V1.0

Release date: Deprecated October 2015

Version no longer available or supported.

CHAPTER 16

FAQs

16.1 Frequently asked questions

16.1.1 Can I trust you with my data?

Great question. See *Compliance* for more information.

16.1.2 How can I get a key?

Click here to signup.

16.1.3 How do I call the API?

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

```
https://api.hippoapi.com/v3/more/json/ABCD1234/john.doe@gmail.com
```

Note: Several response formats (other than *JSON*) are available. For a detailed explanation of the responses available, see *Schema*.

16.1.4 What comes back from the API?

Various text or binary response formats.

Note: For a detailed explanation of the responses available, see *Schema*.

16.1.5 How reliable is the API?

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

16.1.6 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.

16.1.7 Do you cache results?

To deliver the speed and reliability demanded by our customers, verification results are cached as follows:

- **Level 1 cache**: CloudFlare based. Cache expiration 2 hours.
- **Level 2 cache**: Microsoft Azure based. Cache expiration 30 days.

No personally identifiable information is stored in our cache infrastructure.

16.1.8 Can I get my usage in real-time?

Yes. Please see *Usage report* for more information.

16.1.9 Can it do Hotmail?

Yes.

16.1.10 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.

16.1.11 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.

16.1.12 Can I get blacklisted using this API?

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

16.1.13 Will anyone know that I am verifying their email address?

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

16.1.14 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=
```

(continues on next page)

(continued from previous page)

```
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.

CHAPTER 17

Glossary

17.1 Glossary

ACL Access Control List.

An ACL determines what networking traffic is allowed to pass and what traffic is blocked.

An ACL change is sometimes required to your company firewall in order to access our API.

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*.

BSON Binary Object Notation

See Wikipedia - BSON for further information.

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.

protobuf Protocol Buffers is a method of serializing structured data.

See Wikipedia - Protocol Buffers for more information.

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.

RFC Request for Comments

The principal technical development and standards-setting bodies for The Internet.

See Wikipedia - Request for Comments 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.

TXT TXT records associate arbitary and unformatted text with a domain. TXT records uses include Sender Policy Framework (SPF) and other domain validation applications.

For more information, see Wikipedia - TXT record.

XML e(X)tensible Markup Language

See Wikipedia - XML for further information.

CHAPTER 18

Indices and tables

- genindex
- modindex
- search

Index