LEADSPEND VALIDATION PLATFORM

Version 2.2

TECHNICAL SPECIFICATION

A-1. OVERVIEW

The LeadSpend Validation Platform provides real-time and batch validation of email addresses. The platform is accessible via a RESTful web service as well as an automated FTP service, as described in *Parts B* and *C* of this specification, respectively.

A-2. VALIDITY

LeadSpend defines the validity of an email address as follows:

Value	*Description*
`verified`	Mailbox exists, is reachable, and not known to be illegitimate or disposable.
`disposable`	Domain is administered by disposable email provider (*e.g.* Mailinator).
`unreachable`	Domain has no reachable mail exchangers (see discussion, below).
`illegitimate`	Seed, spamtrap, black hole, technical role account or inactive domain.
`undeliverable`	Mailbox or domain does not exist, or mailbox is full, suspended or disabled.
`unknown`	We were unable to conclusively verify or invalidate this address.

A **`role`** account is an email address that is not associated with a specific person, but rather with an office, position, group or task. Role accounts can be broadly classified as either *technical* or *nontechnical.*

- A *technical* role account has a local part such as *postmaster*, *abuse* or *spam*. Because such accounts should never be mailed, these addresses are classified as **`illegitimate`**.
- A *nontechnical* role account has a local part such as *sales*, *support*, *info*, *legal*, or *inquiries*. Whether you choose to accept such addresses depends on your use case, and may require case-by-case, manual evaluation. For these addresses, the value of the `role` field will be `true`.

A **`full`** mailbox is **`undeliverable`**, and typically indicative of an inactive account. However, if you would like to accept these, you can identify them by the the existence of the `full` field, containing a value of `true`.

An email address is **`unreachable`** if either no mail exchangers are advertised for its domain or they all persistently reject or do not reply to inbound connections. While almost always indicative of a domain which is not used to receive email, transient configuration or DNS errors on the receiving end may also be at fault.

B-1. WEB SERVICE OVERVIEW

The LeadSpend Validation Platform is accessible via a RESTful web service for real-time validation. API resources are identified by URLs and interacted with using the standard HTTP methods: GET, PUT, POST and DELETE.

An illustrative example of a real-time query and its result is given in *Section B-9*.

B-2. VERSIONING

The client specifies the desired API version in the URL of each request (*see Section B-5*) so that new versions do not break existing deployments. Whenever the introduction of compatibility-breaking changes is necessary, a new version number will be allocated, but this should be relatively infrequent. The most recent two API versions will be supported at any given time, but only the newest version will be actively maintained.

B-3. SECURITY

All API traffic is encrypted via TLS (SSL) and the servers are authenticated via 2048-bit X.509 digital certificates. Client-side (JavaScript) queries are authorized via the HTTP Referer (sic) header (*N.B.* in order to prevent theft of service by third parties via header spoofing, only a limited number of Referer (sic)-authorized queries are accepted per IP address). Server-side queries must use HTTP basic access authentication.

B-4. AVAILABILITY

In order to promote service availability even in the event of a data center-wide disruption, two API servers are provided, situated in geographically distinct data centers. Requests may be directed to either server. No appreciable difference in performance or behavior should be expected. Clients are expected to alternate API servers in the event of communication failure or persistent errors.

B-5. URL STRUCTURE

All resources are exposed according to the following URL structure:

```
https://{server}/{version}/path/to/resource
```

Required parameters:

Name	*Description*	*Value(s)*
`server`	The server name.	`primary.api.leadspend.com`, `secondary.api.leadspend.com`
`version`	The API version.	`v2`

B-6. STATUS CODES

Clients must examine the HTTP status code of server replies before attempting to interpret their content. The following status codes are specifically defined, but clients should be prepared to interpret other status codes, such as might be generated by intermediate proxies or firewalls, as per RFC 2616:

Status code	*Interpretation*
`200 OK`	The request was well-formed and the reply contains the result.
`202 Accepted`	The request was well-formed but the result is not yet available.
`400 Bad Request`	The request was malformed and should not be retried without modification. The reply content will contain a description of the error.
`401 Unauthorized`	No authorization credentials were supplied, they were invalid, or they were encoded incorrectly. Check your `WWW-Authenticate` header.
`500 Internal Server Error`	An unexpected error occurred. The request may be retried (perhaps with an alternate server), but persistent failures should be reported.
`503 Service Unavailable`	The server is temporarily overloaded and not accepting new requests. Requests should temporarily be directed to an alternate server.

B-7. REPRESENTATIONS

The default resource representation format is JSON; however, XML representations can be requested by specifying `application-xml` in the `Accept` header. Clients unable to modify headers may pass this value via an `accept-header` URI parameter, instead.

When validating server replies, clients should ensure that all expected fields are present. However, in order to accommodate intra-version additions to the API, any unexpected fields must be ignored. Clients must not rely on any undocumented fields.

B-8. REAL-TIME VALIDITY RESOURCE

GET is the only method allowed for this resource. Its relative path has the following structure:

```
/validity/{address}?timeout={timeout}
```

There are two required parameters:

Name	*Description*	*Value*
`address`	The email address to validate.	A percent-encoded email address.
`timeout`	The timeout, in seconds.	Integer (truncated to the range [*3*, *15*]).

The representation (named `validity`, if XML) will contain the following fields:

Name	*Description*	*Value(s)*
`address`	The email address.	A string (nominally an email address).
`result`	The validation result.	See *Section A-2*.

The representation may also contain one or more of the following fields:

Name	*Description*	*Value(s)*
`role`	Is this a nontechnical role account?	`true` or not present.
`full`	Is the mailbox full?	`true` or not present.
`timeout`	Was the query's timeout exhausted?	`true` or not present.
`retry`	You may retry after this many seconds.	An integer or not present.

Clients will receive a `result` of `unknown` for any queries not satisfied within the timeout period, starting from when the request is received, and ending when it is replied to, server-side.

Results of `unknown` or `unreachable` may occasionally be due to transient issues, such as DNS errors, mail exchanger misconfiguration, server rate-limiting (for real-time queries) or even internal errors. When there is any possibility of this being the case, the `result` field will be accompanied by a `retry` field, specifying how many seconds must be waited before retrying the query, if desired.

B-9. REAL-TIME VALIDITY RESOURCE—EXAMPLE

To validate `fake-address@leadspend.com`, the client might `GET` the resource at:

```
https://username:password@primary.api.leadspend.com
   /v2/validity/fake-address@leadspend.com?timeout=5
```

and would receive the following representation:

```
{ “address”: “fake-address@leadspend.com”,
  “result”: “undeliverable” }
```

C-1. FTP SERVICE OVERVIEW

The LeadSpend Validation Platform is accessible via an FTP service for batch processing. Special folders (see *Section C-6*) and error-checking (see *Sections C-7* and *C-8*) facilitate robust, fully automated integrations.

C-2. VERSIONING

The client specifies the desired service version in the server hostname (see *Section C-5*) so that new versions do not break existing deployments. Whenever the introduction of compatibility-breaking changes is necessary, a new version number will be allocated, but this should be relatively infrequent. The most recent two service versions will be supported at any given time, but only the newest will be actively maintained.

C-3. SECURITY

If the client uses FTPS then traffic is encrypted via TLS (SSL) and the servers authenticated via 2048-bit X.509 digital certificates. FTP is supported, but is not secure.

No client's files nor folders are visible to nor accessible by any other client.

C-4. AVAILABILITY

In order to promote service availability even in the event of data center-wide disruption, two FTP servers are provided, situated in geographically distinct data centers. Results will only be available on the server to which the file was submitted. Clients are expected to alternate FTP servers in the event of persistent network failure.

If an FTP server becomes unreachable while processing a file, the client may resubmit it to the other server. As long as the file's contents are *exactly* the same *(N.B.* dissimilar filename are fine), clients will not be double-charged for any overlapping queries that result.

C-5. HOSTNAME

The hostname of each FTP server conforms to the following structure:

```
{version}.{server}
```

Required parameters:

Name	*Description*	*Value(s)*
`server`	The server name.	`primary.ftp.leadspend.com`, `secondary.ftp.leadspend.com`
`version`	The service version.	`v2`

C-6. FOLDERS

The client has access to the following folders:

Path	*Description*
`uploads`	If possible, upload files here then move them to `incoming` (see *Section C-7*).
`incoming`	Place files here to submit them for validation.
`outgoing`	Pick up completed files here.
`errors`	Any malformed files placed in `incoming` are moved here.

The client has the following filesystem permissions and restrictions:

- Cannot create, rename or remove any folders.
- Can only upload files into the `uploads` and `incoming` folders.
- Can only move/rename from `uploads`, and only into `incoming` or `uploads`.
- May delete files from any folder.

C-7. UPLOADS

Surprisingly, the FTP(S) protocol does not provide any mechanism by which a server can reliably distinguish a failed upload (and a truncated file) from a successful one. As such, clients are strongly encouraged to upload files to the `uploads` folder and then move them to the `incoming` folder. Uploading directly to `incoming` is also allowed, but the client risks running a truncated file if there is a network interruption.

C-8. INPUT

Uploaded files must be RFC 4180-compliant CSVs (*cf.* http://tools.ietf.org/html/rfc4180):

- Commas are used to separate fields.
- DOS/Windows-style newline (CR+LF) are used to terminate records.
- The last record may or may not be terminated.
- Double-quotation marks are used to quote fields.
- Double-quotation marks are used to escape double-quotation marks.
- Fields containing commas, newlines or double-quotation marks must be quoted.
- Otherwise, quoting is optional.

In addition to the above, a header row is required. The email-address column must be named `email` (case-insensitive). Additional columns may be included, and will be preserved in the output, but none of them may be named `result`, `role` nor `full`. Additional column names may be reserved, in the future.

ASCII-mode FTP converts newline sequences during transfer, so use binary mode!

C-8. OUTPUT

Output files will adhere to the same format as used for input, except containing additional fields named `result`, `role` and `full` (with values as described in *Section B-8*).

To simplify parsing (*e.g.* for direct, MS-SQL bulk import), *every* field will be quoted.

C-9. AUTOMATION

It is helpful to think of the FTP service as a *state machine*: Whichever folder a file is in reflects its current processing *state*.

Importantly, because a file is only in a single state at any given time, it is not necessary to poll every folder. Rather, having uploaded a file, one need *only* poll the `incoming` folder. When the file is no longer there, one should look at the `errors` and `outgoing` folders to see if the file was malformed or successfully processed, respectively.

If malformed, manual intervention is required—something is wrong with how the file is formatted, and re-uploading it will not help. Review *Section C-8* and ensure you have included a proper header row, are quoting any special characters (and properly escaping any quotes), are using the correct line terminators, and are using binary transfer mode.

If successfully processed, the results will remain in the `outgoing` folder for at least 30 days, after which they may be deleted automatically by the FTP service. It is the client's responsibility to collect results before this time—and likewise to delete them, if so desired.

Before acting on the results, the client should always ensure they conform to the expected format, contain the expected records, and the results are within reasonable bounds.