



LashBack DataSource Validator (beta)

Last Updated: Sept., 17, 2008

Overview

Lead manipulation is becoming an increasing problem in the online marketplace. More and more online marketers are being plagued with lead manipulation and forged sign-ups. Recent examples include rogue parties signing up email addresses harvested from suppression files and harvested from the web and dictionary attacks. This subjects legitimate marketers to decreased deliverability through blacklisting, higher complaint rates and a negative reputation in the marketplace. Unfortunately, even the best run lists are subject to this problem, even if double-opt in procedures and thorough lead validation is performed.

Solution: DataSource Validator

LashBack's DataSource Validator is a proactive, community-based, collaborative effort to collect, identify and distribute known IP addresses of forged lead sources.

How it Works

LashBack has millions of email addresses, called probes or seeds, which exist in only one place – a suppression file. These email addresses should never be signed up or sent email. LashBack clients can make a simple web service call as new leads come in or on a batch basis, check each lead and lead source IP against LashBack's list of probe email addresses and source IPs reported by the community that have been identified as submitting probes from LashBack with fraudulent identity information as valid lead data. In this way, the more LashBack clients who use the service the better the service becomes- true collaboration.

Technical Details

The Lead Source Validation SOAP web service is available from:

<http://secure.lashback.com/leadsourcevalidator/LeadSourceValidator.asmx>

Or via SSL at:

<https://secure.lashback.com/leadsourcevalidator/LeadSourceValidator.asmx>

Function: ValidateLead

Used for validating leads at the time of collection. This function processes one lead at a time.

Input Parameters:

strClientUserName (String): Your LashBack client username. e.g.

"Testusername". If you do not have your client username, please contact

your Client Services Manager.

strClientPassword (String): Your LashBack client password. e.g., “testpassword” If you do not have your client password, please contact your Client Services Manager.

strMD5ofLowerCaseSignupEmailAddress (String): A 32 bit MD5 Hash of the lower case of the lead’s email address. E.g., “47472247b9a55e34a590632e9f5d5609”. **Important** – please convert the lead’s email address to lower case before computing the MD5 Hash.

strSignupSourceIP (String): The IP address of the lead signing up. E.g., “127.0.0.1”.

Output Parameters:

Result – Integer. Possible Values are:

- 1 – data passed in appears invalid or username and password failed to validate
- 0 – Neither the email address nor signup IP appears to be from a bad source
- 1 – Either the IP address of the lead source is known to be bad or the email address of the lead.

Sample SOAP Calls

- **VB.NET** – <https://secure.lashback.com/leadsourcevalidator/examples/VBNET-LeadSourceValidatorExample.zip>
- **PERL with SOAP LITE**
-<https://secure.lashback.com/leadsourcevalidator/examples/PERLSOAPLITE-LeadSourceValidatorExample.zip>
- **PHP SOAP** –
<https://secure.lashback.com/leadsourcevalidator/examples/PHPSOAP-LeadSourceValidatorExample.zip>
- **PHP Pear** -
<https://secure.lashback.com/leadsourcevalidator/examples/PHPPEAR-LeadSourceValidatorExample.zip>

Function: ProcessLeadFile

Used for validating a file of leads in bulk.

Input Parameters:

strClientUserName: Your LashBack client username. e.g. “Testusername”. If you do not have your client username, please contact your Client Services

Manager.

strClientPassword: Your LashBack client password. e.g., “testpassword” If you do not have your client password, please contact your Client Services Manager.

bytFileData: A byte array containing your lead file data. The byte array should contain the following fields separated by commas: md5 hash of the lower case of the lead’s email address and the lead’s signup IP address. Each row should be separated by a carriage return and line feed (char 13, char 10). A sample lead file can be seen at

<https://secure.lashback.com/leadsourcevalidator/examples/samplefile.txt>.

The corresponding byte array that would be passed for this file can be viewed at

<https://secure.lashback.com/leadsourcevalidator/examples/samplefilebytes.txt>

Output Parameters:

This function returns an object called ProcessLeadFileResult. This object has the following properties:

ProcessLeadFileResult (object):

intBadHashes (Integer): represents the number of records found in the file which contained invalid MD5 Hashes (the length of the field was not 32-bits).

intBadIPs (Integer): represents the number of records found in the file which contained invalid source IP addresses.

intGoodLeads (Integer): represents the number of leads which were checked that were found to be good leads.

intBadLeads (Integer): represents the number of leads which were checked that were found to be bad leads.

intTotalRecordsProcessed (Integer): represents the total number of records which were imported and checked, which should match the number of records passed in.

intResultCode (Integer): Indicates whether file processing was a successful. If the result code is 0, then the process succeeded. If the intResultCode is -1, then the process failed. In this case, strErrorInfo will indicate additional information as to the cause of the failure.

strErrorInfo (String): Additional information on why processing failed. Only populated when intResultCode = -1.

bytReturnData (Byte Array): Byte representation of the lead file process results. This file contains three columns separated by commas: original MD5 hash of the leads email address, original source IP, and Lead Validation Result Code. The Lead Validation Code is an integer and can be one of four values:

-3: the source IP passed in is not valid

-2: the md5 hash of the email address is not valid

0: the lead was checked and is NOT considered bad

1: the lead was check and IS considered bad

Each row is separated by a carriage return and line feed (char 13, char 10).

This byte array may be saved to file and used for processing of results. A sample of bytReturnData can be viewed at:

<https://secure.lashback.com/leadsourcevalidator/examples/samplefileresultbytes.txt>. The corresponding text file, once the bytes are saved to file, can be viewed here:

<https://secure.lashback.com/leadsourcevalidator/examples/samplefileresults.txt>

Sample SOAP Calls

- **VB.NET –**
<https://secure.lashback.com/leadsourcevalidator/examples/VBNET-ProcessLeadFile.zip>

- **PHP SOAP-**
<https://secure.lashback.com/leadsourcevalidator/examples/PHPSOAP-ProcessLeadFile.zip>

Q & A

How much does the service cost?

During the beta period, the service will be provided at no charge. Depending on the cost to support this service, it may continue to be offered as a no-fee service or may be converted to a paid service. However, the no-fee beta period will be offered for at least 3 months.

Are my signup IPs and email addresses shared with other parties?

No. First, MD5 hash is a one-way encryption method. So LashBack or other parties would not be able to reverse-engineer the lead's email address hash to determine their actual email address. Further, only hashes which are found to be from a bad lead source are stored. As far as source IP addresses, only IPs of source found to have submitted a LashBack probe as a lead are stored so other parties in the system can be made aware that the source IP is bad.

How will LashBack prevent misuse of the service?

LashBack is not creating this service for list washing and is very conscientious of this perception. During the beta period, LashBack will be working to identify potential sources of misuse and may change the requirements of the service during this process.

Will LashBack be looking at other email addresses besides their probes?

If the service is leveraged, then LashBack will contact members of the anti-spam community to add other types of honey pot and other seed email addresses into the system. This is done to protect consumers whose email addresses are now being abused as a result of this fraudulent and criminal activity and legitimate email marketers striving for compliancy and the proper execution of best practices.