Mail Fulfillment by LetterStream

Integration API
December 21, 2020

LetterStream, Inc.
8551 East Anderson
Suite 108
Scottsdale, AZ 85255

Phone 480-473-3282
Toll Free 1-888-501-5288
Fax 480-473-3208
Web. . . . . .www.letterstream.com
email. . . . . chris@letterstream.com

Changes:
December 21, 2020
    Updated all urls to https://www.letterstream.com
August 31, 2020
    Pointed out that batch id must be unique (consists of zip naming in batch method and job name in post method)
March 26, 2020
    Added duplex info for post method
Oct 8, 2019
    Added base64 decoding info for proof requests
Jun 21, 2019
    Added latest curl file upload code to php samples
Jan 14, 2019
    Added document proof call information
Dec 5,2018
    Added firstclass_hse option
Dec 4, 2018
    Added paper options table including additional options
Sep 20, 2018
    Updated verbiage about to address unique doc id requirements
Feb 28, 2018:
    USPS tracking number length change to 22 digits
Jan 12, 2018:
    Added json responseformat
    trackx will return xml formatted tracking data

I. Overview

With a LetterStream API Partnership you can quickly and easily provide your customers with access to our highly automated mail processing services. Your customers can send postcards, first-class letters (in both #10 and large flat envelopes) and USPS certified mail. All of this without ever leaving your website or application. To your users it can be as simple as clicking a button.

In order to participate in this service the following items need to describe your situation:

1. I have an existing website/application in which myself or my clients need the ability to send postal mail easily.
2. I can easily bill my clients for mailings at the time they occur.
3. I produce and or have electronic access to the documents that I or my clients want to mail.
4. Further, I have access to the sender/recipient data that accompanies those documents.
5. I am, or have a qualified software developer who can modify my code to successfully implement a web-based API.

If the above criteria describes your situation, the LetterStream API may be just what you need. If you have questions or concerns about the criteria above, you may want to contact LetterStream before proceeding.

II. Sign up (attaining your API_ID and API_KEY)

If you are reading this document, then you should have already signed up for an API account. If not, please visit http://www.letterstream.com/api.php and do so now. After signing up, you'll be asked to verify your email address and will then be contacted by a LetterStream API specialist. After a brief discussion with our API specialist, you will be authorized to login to your LetterStream account.

What you'll see

As an API partner, you'll see our standard dashboard (My Jobs), where all of your mail jobs will be listed. The dashboard is fully search-able and provides all of the information that you will be able to retrieve via calls to the API. Additionally, under the (My Account) tab you will see a link to (API Information). There you will find additional information/help regarding the API.

API Docs
The API Docs are available for viewing and downloading. Also on this page you may find your API Identifier (API_ID) and an integration key (API_Key). You can think of these two variables as the username and password that will be used to authenticate your API access into the LetterStream system. In order to maintain a level of security, the API_KEY will be used as a PSK (Pre Shared Key) to generate a hash that will accompany the API_ID for account verification. As such, it's imperative that the API_KEY never be shared. (more about this below)

Reports
We provide some basic reporting to our API partners that will provide at-a-glance usage information.

PreFlight
A tool to allow visual inspection/verification that your certified and first-class documents will fit our windowed envelopes.

Knowledge Base
Due to the nature of our API partnership it is your responsibility to provide front-line support of mail issues to your users. As such, we provide a continually updated knowledge-base for your reference. Through the Knowledge Base link, you will be able to access this information and submit new trouble tickets. Additionally, we are available via email and phone to provide you (not

your customer/user) with support on any issue that may arise.

Manage PrePay
Found under the (My Account -> Billing Information) tabs. As an API Partner you will be required to maintain funds on account to cover the cost of your submitted mailings. In other words, you are required to pay in advance. Your account is able to accept mail jobs prior to payment, but those jobs will not be released into our production environment until payment is received. The Manage PrePay link provides tools necessary for you to manage and add to your prepay funds.

III. Getting Started

Important: Due to the nature of the fulfillment model, we have written these apis with the idea that our api partners desire to retrieve the available information in the background (server to server) and then display that information to your users in-line within your website/application. As such, while some of the examples in this document are written as simple GET query strings, this is done for clarity of illustration. All of our apis will require a POST. Some return data will be enclosed in <div> tags which should provide an easy way for you to format as desired. Further, when requesting a file (certified signatures or document pre-flighting via the API), the file will be streamed back to you. It will be necessary for you to save that stream to a file and return that to your users with the appropriate headers.

Your responsibility: While we shoot for 100% up-time, it is your responsibility, in this partnership, to write your code in a way that will gracefully handle any sort of outage. If you don't get a success response from our system, your system needs to either queue the request and try again later, or notify someone so that support can be sought.

Debugging: To aid in setup, it's possible to include a "debug" argument with any connection. This will cause any relevant errors or system messages to be returned in a more verbose manner. There are three error levels (1,2,3) with debug='3' being the most verbose.

First step – Account Authentication: The first item of interest is the Account Authentication request. To maintain a level of security, we require that an md5 hashed string be passed with all requests along with a unique id (timestamp). The string to hash will be constructed as follows. You may also find additional tools under (My Account -> API Information) to aid in the process of constructing the hashed string.

Table 3.1 Example of an Authentication Request

```
$api_id = 'qwerty';          //found by logging into your LetterStream account
$unique_id = time();         //this will only be accepted by our system one time – 10-18 digits long
$string_to_hash = substr($unique_id,-6).$api_key.substr($unique_id,6);
$hash = md5(base64_encode($string_to_hash));
```

Then the assembled query would look like...

```
https://www.letterstream.com/apis/index.php?a=$api_id&h=$hash&t=$unique_id&debug=3
```

To recap, create a unique id (a Unix timestamp may be suitable, but be aware that the unique id can only be submitted to our system one time and must be numeric 10-18 digits in length). Concatenate the last 6 digits/character of the unique id with your complete **api_key** and the first 6 digits/characters of the unique id. Then base64 encode and finally md5 that string and pass the hashed string (as argument "h"), your api_id (as argument "a") and the unique id (as argument "t"). You can expect to receive one of the following responses from our system with debugging turned on.

Table 3.2 Examples of Verification Response

```
AUTHOK 051010 08:14:38    //account in good standing and connection successful
IDOK 051010 08:14:48      //API_ID found but hash lookup failed
DUP 051010 08:14:52        //unique id is not valid (duplicate)
BAD 051010 08:14:56       //API_ID not valid
```

IV. Job Submission

1. Batch Submission (method 1) *preferred

NOTE: It's best to start by using the "Complete ZIP example" found at the top of the API documentation page on the LetterStream website.

Batch submission allows you to submit mixed mail data. You can submit first-class letters along with postcards or postcards with a certifed letter and some large envelope (flats) letters. In other words you can send any mixup of mail data in a single process.

Overview and Requirments

This method requires that you assemble your document collection and a simple csv data file into a zip archive.

First, each unique recipient must include a unique/stand-alone pdf file (**there is one exception to this noted below). Second, there must be an accompanying csv data file that contains a unique id for each mail piece (recipient), the pdf filename, the mail type, and sender/recipient information and optional information. This data file and all accompanying pdfs must be archived in a zip format. The name of the csv file will be used for the root of the job name(s) and as the batch id. As such the csv's name must be unique and should be something that will help you to identify this group of mail for future reference.

This method has a **filesize cap of 50MB**.

**The exception to the one pdf/one recipient requirement as noted above is in the case of some legal documents that are sent to multiple recipients. Those documents generally contain the names and addresses of all intended recipients so that all may see who is copied on that correspondence. In this case, each recipient must still carry a unique id, but the filename field can point to a single pdf. Documents of this type will require a coversheet to accommodate our windowed envelopes.

CSV file formatting

Again, the name of the csv file will be used as both the batch id and the root of the names of the jobs created within the batch **AND MUST BE UNIQUE**. So, if you submit a csv named “summary1.csv” the batch id will be “summary1” and jobs will be named “summary1_1, summary1_2, etc”.

The csv contents will have the following columns, default options and required items. A complete example can be downloaded from the api documentation page of your account dashboard.

Table 4.1.1 Batch CSV file layout

#	Column Name	Description	Valid options	Default	Required
1	UniqueDocId	A unique document id. Must be unique to any active or mailed jobs.	Alphanumeric max 20char	na	Yes
2	PDFFileName	The corresponding filename of the pdf as saved in the zip archive Special template options available for propostcards. Please contact us for more info.	Must reference a valid pdf file	na	Yes
3	RecipientName1	Recipient address Name 1		na	Yes
4	RecipientName2	Recipient address Name 2		na	optional
5	RecipientAddr1	Recipient address street address			Yes
6	RecipientAddr2	Recipient address suite #			optional
7	RecipientCity	Recipient address city			Yes
8	RecipientState	Recipient address state – 2 characters	2 char alpha		Yes
9	RecipientZip	Recipient address zip	5-10 numeric plus “-”		Yes
10	SenderName1	Sender address Name 1			Yes
11	SenderName2	Sender address Name 2			optional
12	SenderAddr1	Sender address street address			Yes
13	SenderAddr2	Sender address suite #			optional
14	SenderCity	Sender address city			Yes
15	SenderState	Sender address state	2 char alpha		Yes
16	SenderZip	Sender address zip	5-10 numeric plus “-”		Yes
17	PageCount	Number of pages in pdf file included in column 2	numeric		Yes
18	MailType	The type of document/mail piece	firstclass|firstclass_hse|certified|certnoerr|postcard|flat|propostcard	firstclass	No will default
19	CoverSheet	Shall we generate and include a coversheet? If your document is not formatted to fit our windowed envelopes, you must select Y for this option. We automatically create the coversheet using the supplied address information.	Y|N	Y	No will default
20	Duplex	Print on both sides (Y) or just one side (N)	Y|N	N	No will default
21	Ink	(B)lack and white or (C)olor ink	B|C	B	No will default

22	Paper	We have numerous paper options including colors and perforated statement paper, etc. Please ask about any special paper requests.	See table below for options	W	No will default
23	Return Envelope	#9 Right window return envelope with security tint. Designed to be used in conjunction with PERF(orated) paper option. Window dimensions are from bottom left side of document 4.625 inches to 8.375 inches and from the bottom edge of the document .625 inches and 1.75 inches. Please use the pre-flight tool to verify positioning. 634 for statement paper.	Y|9RWS|9LWS|634|N 634 – 6 ¾ return env Y = 9RWS either option can be specified 9LWS should be specified for a #9 Left Window Security return envelope	N	No will default
24	Affidavit	In some special circumstances, customers require an affidavit of mailing. We can also provide certificate of bulk mailing as well as additional services. These services will incur additional fees. Contact us for more information.	A|N	N	No will default

Table 4.1.2 Example of a simple batch submission method POST

```
<form method="post" enctype="multipart/form-data"
action="https://www.letterstream.com/apis/index.php">
<input type="hidden" name="a" value="<? echo $api_id ?>">
<input type="hidden" name="h" value="<? echo $hash ?>">
<input type="hidden" name="t" value="<? echo $unique_id ?>">
<input type="file" name="multi_file">                    //your assemble zip file
<input type="submit" value="Submit test to LetterStream">
</form>
```

Table 4.1.3 Paper Options

W = White 8.5x11 20#
Y = Canary 8.5x11 24#
B = Light Blue 8.5x11 24#
G = Light Green 8.5x11 24#
O = Orange 8.5x11 24#
R = Light Red 8.5x11 24#
I = Ivory 8.5x11 24#
PERF = 8.5x11 2/3rd Perf tearoff 24# - tearoff fits in 9RWS and 9LWS return envelopes
PERF2 or STMT = 8.5x11 Statement Paper with coupon perf
    and stub 24# - coupon fits in 634 (6 ¾ return envelope)
CKBLUE or CKB = Blue check stock (check on top)
CKRED or CKR = Red (burgundy) check stock (check on top)
CKGRN or CKG = Green check stock (check on top)
CPN4UP = 4 up coupon paper with coupon and stub. Coupon fits 634 (6 ¾ return envelope)

Postcard options W4X6 = White cardstock cut to 4”tall x 6”wide W85X55 = White cardstock cut to 5.5”tall x 8.5”wide W5X425 = White cardstock cut to 4.25”tall x 5”wide W8X4 = White cardstock cut to 4”tall x 8”wide

2. HTTP POST (method 2)

This method has a **transaction limit of 50 per day** and is intended for low volume customers.

This method provides a simple inline interface when sending a single file to one or more recipients. The POST should contain the required fields listed below unless marked as optional. Please refer to table 4.2.2 for formatting address arguments and table 4.2.3 for a look at how you can create a simple, effective POST, although, please note that this api interface is designed for server to server communication and as such, provides no user interface.

Table 4.2.1 Required Fields

Field Name	Description
a	Your api_id
h	Md5 hash as explained in table 1.1
t	Unique id (timestamp) as explained in table 1.1
job	(required) ** **must be unique** ** A job name. Even though it may only be one recipient, it's still a job in our system. Adding an easily recognizable name for future reference will make it easier to find information later on. This must be unique across all active or mailed jobs.
to[]	(required) an array of recipients addresses shall be formatted as described in tables 2.2 and 2.3, a
from	(required) from/return address (max of 1 from address)
single_file	(required) the pdf to be mailed – this single file will be sent to every recipient in the “to” argument The file can be transferred as a multipart/form-data file element or as blob. Base64 encoding may be necessary for the latter.
pages	(required) number of pages in file
mailtype	(optional defaults to firstclass) Type of mail ('firstclass',’firstclass_hse’,'certified',’certnoerr’, 'postcard','flat',’propostcard’)
coversheet	(optional defaults to true) True/false shall we create a coversheet or is your pdf formatted to fit our envelopes
duplex	(optional defaults to simplex) (Y)duplex|(N)simplex
ink	(optional defaults to B) (B)lack|(C)olor
paper	(optional defaults to W) paper options listed above

returnenv	(optional defaults to N) options are Y|9RWS|9LWS|634|N

Table 4.2.2 To/From postal address formatting

doc_id:name_1:name_2:address_1:address_2:city:state:zip recipient example 123456:John:Doe:123 S Sunny St:Suite 101:Scottsdale:AZ:85281 sender example (doc_id is not included) John:Doe:123 S Sunny St:Suite 101:Scottsdale:AZ:85281 ****only domestic addresses are eligible for certified mail delivery**** **doc_id must be unique to every recipient and should follow the same specification as in table 4.1.1->UniqueDocId** **job must be unique to every valid submission A note about string delimiters: It is possible to use either : (colon) or | (pipe) character as an address string delimiter under the POST method. Please use one or the other, do not mix.

Table 4.2.3 Example of method #2 POST

```
<form method="post" enctype="multipart/form-data"
action="https://www.letterstream.com/apis/index.php">
<input type="hidden" name="a" value="<? echo $api_id ?>">
<input type="hidden" name="h" value="<? echo $hash ?>">
<input type="hidden" name="t" value="<? echo $unique_id ?>">
<input type="hidden" name="job" value="my unique job name">
<input type="hidden" name="from" value="Chris:Coleman:123 South Some St:Suite
123:Scottsdale:AZ:85281">
<input type="hidden" name="to[]" value="12345678:John:Recipient1:456 South Some St:Suite
456:FirstCity:AZ:85281">
<input type="hidden" name="to[]" value="98765432:Jane:Recipient2:457 South Some St:Suite
457:SecondCity:AZ:85282">

<input type="file" name="single_file" value="">
<input type="submit" value="Submit test to LetterStream">
</form>
```

*please note, each to[] address must have a unique document id as described in section 4.1.1 also job must be unique

3. Preauth

It is possible to submit a preauth=1 argument with your job submission. By doing so, your request will be processed but will not be released into production until an authorization request has been received. This option was primarily created to allow for the return of pricing information prior to releasing the job. In other words, it would be possible to submit a job in preauth mode, present your user with pricing information and upon their acceptance of the pricing, submit an auth request which would release the item into our production environment. Examples of the preauth and auth responses can be found below.

Preauth Authorization

Authorization is achieved by resubmitting the authcode as the argument "doauth" (doauth=authcode).

Table 4.3.1 Examples of Submission Response

Successful job submission response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messages id="your_api_id">
    <message type="info">
        <code>-100</code>
        <details>Success</details>
        <batch>your_batch_id</batch>
        <quantity>2</quantity>
        <cost>12.28</cost>
        <doc><id>1305794309</id><job>117001</job><cost>6.14</cost></doc>
        <doc><id>1305794310</id><job>117001</job><cost>6.14</cost></doc>
    </message>
</messages>
```

Successful preauth response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messages id="your_api_id">
    <message type="info">
        <code>-200</code>
```

```
            <details>Successful preauth: Please resubmit authcode to complete order</details>
            <authcode>a6e9cc7f2643a4d5c169cfb9212431sd</authcode>
            <batch>your_batch_id</batch>
            <quantity>1</quantity>
            <cost>0.83</cost>
            <doc><id>398</id><job>117002</job></doc>
      </message>
</messages>
```

Successful preauth authorization

```
<messages id="your_api_id">
      <message type="info">
            <code>-200</code>
            <details>Success</details>
            <batch>your_batch_id</batch>
            <quantity>1</quantity>
            <cost>0.83</cost>
      </message>
</messages>
```

An error returned for lack of monetary funding

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messages id="your_api_id">
      <message type="error">
            <code>-911</code>
            <details>Warning: Insufficient funding. Items will not be processed until additional funding
is added to your account (please add funds to your account) </details>
      </message>
</messages>
```

V. Mail Types

As seen above in method 2 and in the summary file for method 1, it is possible to send a number of different types of mail. These types include:

1. 'firstclass' - First Class Letter (#10 2-window envelope)
   - options include color print, color paper, simplex/duplex, return envelope (#9 windowed), coversheet (optional if address placement is in conformance with our envelopes)
2. 'firstclass_hse' - First Class Letter (#10 2-window envelope with "Homeowner Statement Enclosed" endorsement printed in black on front of envelope)
   - options include color print, color paper, simplex/duplex, return envelope (#9 windowed), coversheet (optional if address placement is in conformance with our envelopes)
3. 'certified' - Certified with Electronic Return Receipt (#10 3-window envelope, to/from location matches first-class envelope but includes a 3rd window for certified mail barcode and tracking number)
   - options include color print, color paper, simplex/duplex, return envelope (#9 windowed), coversheet (optional if address placement is in conformance with our envelopes)
4. 'certnoerr' – Certified without Electronic Return Receipt (#10 3-window envelope, to/from location matches first-class envelope but includes a 3rd window for certified mail barcode and tracking number) – FYI, a signature will NOT be collected by the USPS
   - options include color print, color paper, simplex/duplex, return envelope (#9 windowed), coversheet (optional if address placement is in conformance with our envelopes)

5. ‘postcard’ - Postcard (5.5”x4.25” 100# cardstock addressed on one side, pdf imposition on reverse)
    - options include color print, color cardstock
6. ‘flat’ - Flats (10x13 windowed envelope capable of holding 75sheets of 8.5x11 paper, includes coversheet by default)
    - options include color print, color paper, simplex/duplex, return envelope (#9 windowed), coversheet (optional if address placement is in conformance with our envelopes)

VI. Tracking Data

There are two ways to access letter tracking data for certified mail. These samples are given as a simple query string for ease of explanation. Actual requests should be made as POST.

Using the argument value “track” will return html formatted response string. If you prefer to receive this data in xml format please use “trackx”.

Table 6.1 Linking to tracking information

https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id&cert=71147344282010607726 &getinfo=track
OR
https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id&doc_id=0123456789&&getinfo=track
The second method will return job status if document mailtype is not certified

To retrieve the signature file, simply change the getinfo argument to “getinfo=sig”, like this:

Table 6.2 Request signature file

https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id&cert=71147344282010607726 &getinfo=sig
OR
https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id&doc_id=0123456789&getinfo=sig
The second method will return -1 if document mailtype is not certified

Signature files are returned as streamed data. Therefore, it is necessary for you to handle that stream and apply necessary headers for presentation to your users. You can find an example of this in the appendix of this document.

To retrieve the document proof, using doc_id as the reference field, simply change the getinfo argument to “getinfo=proof”, like this:

Table 6.3 Request document proof

https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id&doc_id=0123456789&getinfo=proof

Proofs are returned as base64 encoded streamed PDF data. Therefore, it is necessary for you to handle that stream (base64 decode) and apply necessary headers for presentation to your users. You can find an example of this in the appendix of this document.

VII. Batch, Job, Doc, Account Status

You may query our system to determine your mail's current stage of production. This will allow you to provide your customers with some feedback regarding whether or not an item is in production, or already mailed, along with the latest timestamp of this activity within our system. Again, samples presented as simple query string for ease of explanation. Actual requests should be submitted as POST.

Table 7.1 Batch Status information

https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id &batchstatus=mybatch1,anotherbatchid

Table 7.2 Job Status information

https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id &jobstatus=117011,117012

Table 7.3 Doc Status information

https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id &docstatus=0123456789,45678,23456

Table 7.4 Account Status information

Provides account balance information https://www.letterstream.com/apis/index.php?a=your_api_id&h=your_md5_hash&t=your_unique_id &accountstatus=1

VIII. Document Template Pre-Flight

At LetterStream, we use 2 or 3 window #10 envelopes as our primary mail vessel for firstclass and certified mail. Therefor, your documents must be designed in such a way as to allow the TO and FROM addresses to display through the windows. Further, if you are sending a certified letter, there must be blank space available for us to overprint our certified bar-code and tracking number. To help with this, you may submit a document for a visual or automated(coming soon) pre-flight. If the document passes the pre-flight you may submit it without a coversheet, otherwise, the document must be edited to accommodate our envelopes or a coversheet will be needed.

This tool is not meant to be used for every submission. Rather it's intended use is for the verification of document template designs within your system.

You must submit a pdf file and we will return a marked-up pdf, providing visual cues of window placement. By including the display=true argument, the results will be displayed. If you'd prefer a data stream, please omit the display argument.

You may also access this tool by logging in to the LetterStream.com website and visiting “My Account”->“PDF Tools” and using the PreFlight form at the top of the page.

Table 8.1 Submitting document for pre-flight

```
<form method='post' enctype='multipart/form-data'
action='https://www.letterstream.com/apis/index.php'>
<input type='hidden' name='a' value='your_api_id'>
<input type='hidden' name='h' value='your_md5_hash'>
<input type='hidden' name='t' value='your_unique_id'>
<input type='hidden' name='display' value='true'>
<select name='preflight'>
<option value='visual'>Visual Pre-flight</option>
<option value='auto'>Automated Pre-flight</option> //coming soon
</select>
<input type='file' name='preflight_file'>
<input type='submit' value='Do Pre-flight'>
</form>
```

IX. API PUSH (Tracking data)

We provide a callback function that enables our system to automatically push certified mail tracking information to your system as they become available. More information is available on the “API Information” page of our website.

Appendix

Example 1. PHP curl request for signature and display results to user

```
$api_id = 'your_api_id';
$api_key = 'your_api_key';

$unique_id = time();

$string_to_hash = substr($unique_id,-6).$api_key.substr($unique_id,0,6);
$hash = md5(base64_encode($string_to_hash));

$data = array('a' => $api_id, 'h' => $hash, 't'=> $unique_id );

// normally you would collect this from your users form submission
$data['cert'] = '71147344282010607726';
$data['getinfo'] = 'sig';

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'https://www.letterstream.com/apis/');
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);

$response = curl_exec($ch);

if ( preg_match("/%PDF/", $response) ) {

    $temp = fopen('test123.pdf','w');
    fwrite($temp, $response);
    fclose($temp);

    header("Pragma: dummy=bogus");
    header("Cache-Control: private");
    header('Content-disposition: inline; filename=test123.pdf');
    header("Content-Type: application/pdf");
    header("Content-Transfer-Encoding: binary");
    header('Content-Length: '. filesize('test123.pdf'));

    readfile('test123.pdf');
}
elseif( strlen($response) > 10000 ) {
    $temp = fopen('/tmp/test123.pdf','w');
    fwrite($temp, base64_decode($response));
    fclose($temp);
    header("Pragma: dummy=bogus");
    header("Cache-Control: private");
    header('Content-disposition: inline; filename=test123.pdf');
    header("Content-Type: application/pdf");
    header("Content-Transfer-Encoding: binary");
    header('Content-Length: '. filesize('/tmp/test123.pdf'));
    readfile('/tmp/test123.pdf');
}
else {
    echo $response;
}
```

Example 2. PERL submission elements

```
//partial example provided courtesy of Arpit Khemka AMS 2012

use Math::Round;
use LWP::Simple;
use LWP::UserAgent;
use XML::DOM;
use MIME::Base64;
use EMS::Digest::Perl::MD5 qw(md5 md5_hex md5_base64);
```

```
sub sendLetter
{
	my ($jobId,$to,$from,$single_file,$pages,$mailtype,$coversheet) = @_;

	my $api_id = 'your id';
	my $api_key = 'your key';
	my $unique_id = time();
	my $string_to_hash = substr($unique_id,-6).$api_key.substr($unique_id,0,6);
	my $encoded = encode_base64($string_to_hash);
	chop($encoded);
	my $hash = md5_hex($encoded);

	$single_file = "/home/httpd/html/attachments/letter/".$single_file;

	my $ua = new LWP::UserAgent;
	my $response = $ua->post('https://www.letterstream.com/apis/', Content_Type => 'multipart/form-data',
	Content => [	a => "$api_id",
		h => "$hash",
		t => "$unique_id",
		debug => "3",
		job => "$jobId",
		"to[]" => "$to",
		from => "$from",
		pages => "$pages",
		mailtype => "$mailtype",
		coversheet => "$coversheet",
		single_file => ["$single_file"],
	]);
	my $content = $response->content;
	print "\n Response: $content \n";
	return $content;
}

sub processResponse
{
	my ($letterId,$response) =@_;
	my $parser = new XML::DOM::Parser;
	my $doc = $parser->parse ("$response");
	my $responseCode = $doc->getElementsByTagName("code")->item(1)->getFirstChild->getNodeValue;
	my ($jobId,$cost,$error);
	if($responseCode eq '-100' || $responseCode == -100)
	{
		$cost = $doc->getElementsByTagName("cost")->item(0)->getFirstChild->getNodeValue;
		$jobId = $doc->getElementsByTagName("job")->item(0)->getFirstChild->getNodeValue;
	}
	else
	{
		$error = $doc->getElementsByTagName("details")->item(1)->getFirstChild->getNodeValue;

	}
	print "\n In processResponse : $jobId,$cost,$error \n";
	return ($jobId,$cost,$error);
}
```

Example 3. Java Example Using Groovy

```
//Single file submission sample provided by Kent Livingston of Avoka 2013 – Thanks Kent!

import wslite.rest.*
import org.apache.commons.codec.binary.Base64;
import com.avoka.core.util.CoreUtils;
import java.io.File;
import com.avoka.core.util.FileUtils;


// Create a new date and print it out
def now = new Date()
def api_id = ‘your_api_id’; //found by logging into your Letter
def api_key = ‘your_api_key’’;
def unique_id = now.getTime().toString(); //this will only be accepted by our system one time 10 -18 digits long
def job_id = "ADX_" + unique_id
/*
Concatenate the last 6 digits/character of the unique id with your complete api_key and the first 6 digits/characters
of the unique id.
Then base64 encode and finally md5 that string and pass the hashed string (as argument)your api_id (as
argument) and the unique id (as argument)
*/
def start = unique_id.substring(unique_id.length() - 6)
def end = unique_id.substring(0,6)
def string_to_hash = start.concat(api_key).concat(end)
def encoded_str = Base64.encodeBase64String(string_to_hash.getBytes("UTF-8"))
def md5_str = CoreUtils.toMD5Hash(encoded_str)

println "====================================="
println "job_id " + job_id
println "====================================="

def resultFileContent
def resultFileContentBase64

File receiptForSignFile = new File(/path/to/your/pdf/file');

try {
  resultFileContent = FileUtils.readFileAsByteArray(receiptForSignFile);
  resultFileContentBase64 = Base64.encodeBase64String(resultFileContent);
} catch (Exception e) {
  e.printStackTrace();
}

println "====================================="
println 'Content64 ' + resultFileContentBase64
println "====================================="


def fromString = "John:Doe:123 S Sunny St:Suite 101:Scottsdale:AZ:85281"
def toString = unique_id + ":John:Doe:123 S Sunny St:Suite 101:Scottsdale:AZ:85281"
def toStringArray = new String[1]

toStringArray[0] = toString;

def client = new RESTClient("https://www.letterstream.com/apis/")
// for testing only!
client.httpClient.sslTrustAllCerts = true
client.defaultAcceptHeader = "text/xml"
client.defaultContentTypeHeader = "application/x-www-form-urlencoded"
client.defaultCharset = "UTF-8"

def response = client.post(path:'apis/index.php') {
                                                    urlenc single_file: resultFileContentBase64, a: api_id, h: md5_str, t:
unique_id, debug: 3, job: job_id, from: fromString, pages: 1, 'to[]': toString
}

println "statusMessage " + response.statusMessage

import com.avoka.component.xml.XmlDoc

// Determine the type of application
def xmlResponse = new XmlDoc(response.data)
println "xmlResponse " + xmlResponse
```

Example 4. Simple PHP form-based test script

```
<?php
if( $_POST ) {
   $api_id = ‘'your_api_id”;
   $api_key = 'your_api_key';

   $unique_id = time(); //this will only be accepted by our system one time
   $string_to_hash = substr($unique_id,-6).$api_key.substr($unique_id,0,6);
   $hash = md5(base64_encode($string_to_hash));

   $data = array('a' => $api_id, 'h' => $hash, 't'=> $unique_id );

   if( $_FILES['single_file']['tmp_name'] ) {

      if (function_exists('curl_file_create')) { // php 5.5+
           $data['single_file'] = curl_file_create($_FILES['single_file']['tmp_name']);
      } else {
           $data['single_file'] = '@'.$_FILES['single_file']['tmp_name'];
      }
      $data['job'] = $_POST['job'];
      $data['pages'] = $_POST['pages'];
      $data['from'] = $_POST['from'];
      $data['to[]'] = $_POST['uniqueid'].':'.$_POST['to'];
      $data['mailtype'] = $_POST['mailtype'];
      $data['paper'] = $_POST['paper'];
      $data['ink'] = $_POST['ink'];
      $data['returnenv'] = $_POST['returnenv'];
      $data['coversheet'] = $_POST['coversheet'];
      $data['preauth'] = $_POST['preauth'];
```

```
        $data['debug'] = $_POST['debug'];
    }
    elseif( $_FILES['multi_file']['tmp_name'] ) {
        if (function_exists('curl_file_create')) { // php 5.5+
            $data['multi_file'] = curl_file_create($_FILES['multi_file']['tmp_name']);
        } else {
            $data['multi_file'] = '@'.$_FILES['multi_file']['tmp_name'];
        }

        $data['debug'] = 3;
    }
    elseif( $_POST['trackingid'] ) {
        $data['cert'] = $_POST['trackingid'];
        $data['getinfo'] = $_POST['cert'];
        $data['debug'] = 3;
    }
    elseif( $_POST['accountstatus'] ) {
        $data['accountstatus'] = 1;
        $data['debug'] = 3;
    }
    elseif( $_POST['doc_id'] ) {
        $data['doc_id'] = $_POST['doc_id'];
        $data['getinfo'] = $_POST['cert'];
        $data['debug'] = 3;
    }
    elseif( $_POST['doauth'] ) {
        $data['doauth'] = $_POST['doauth'];
        $data['debug'] = 3;
    }

    //print_r($data);
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, 'https://www.letterstream.com/apis/');
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
    $response = curl_exec($ch);

    if ( preg_match("/%PDF/", $response) ) {
        $temp = fopen('/tmp/test123.pdf','w');
        fwrite($temp, $response);
        fclose($temp);
        header("Pragma: dummy=bogus");
        header("Cache-Control: private");
        header('Content-disposition: inline; filename=test123.pdf');
        header("Content-Type: application/pdf");
        header("Content-Transfer-Encoding: binary");
        header('Content-Length: '. filesize('/tmp/test123.pdf'));
        readfile('/tmp/test123.pdf');
    }
    elseif( strlen($response) > 10000 ) {
        $temp = fopen('/tmp/test123.pdf','w');
        fwrite($temp, base64_decode($response));
        fclose($temp);
        header("Pragma: dummy=bogus");
        header("Cache-Control: private");
        header('Content-disposition: inline; filename=test123.pdf');
        header("Content-Type: application/pdf");
        header("Content-Transfer-Encoding: binary");
        header('Content-Length: '. filesize('/tmp/test123.pdf'));
        readfile('/tmp/test123.pdf');
    }
    else {
        echo $response;
        exit;
    }
}
else {

    ?>
    <div class="column">
        <fieldset id="singlefile">
            <legend>Single File Upload w/ Options</legend>
            <br>
            <form method="post" enctype="multipart/form-data">
                Job Name: <input type="text" name="job" value="test job <?=date('m/d/y h:i')?>"><br>
                Your PDF <input name="single_file" type="file"><br>
                Number of pages in PDF: <input type="text" name="pages"><br>

                Return Address: <textarea name="from" cols="40" rows="1">John:Doe:123 S Sunny St:Suite
101:Scottsdale:AZ:85281</textarea><br>
                Unique recipient ID: <input type="text" name="uniqueid" value="<?=time()?>"><br>
                To Address: <textarea name="to" cols="40" rows="1">Husband:and Wife:456 South Some
```

```
St:Apt1:Some City:AZ:85260</textarea><br>
                Mail Type: <select name="mailtype">
                    <option value="firstclass">firstclass</option>
                    <option value="certified">certified</option>
                    <option value="certnoerr">certified no err</option>
                    <option value="postcard">postcard</option>
                    <option value="propostcard">PROpostcard</option>
                    <option value="flat">flat</option>
                </select><br>
                Paper: <select name="paper">
                    <option value="">plain white</option>
                    <option value="PERF">PERF Statement</option>
                    <option value="Y">Yellow</option>
                    <option value="B">Blue</option>
                    <option value="G">Green</option>
                    <option value="O">Orange</option>
                    <option value="I">Ivory</option>
                    <option value="R">Red</option>
                </select><br>
                Coversheet? <select name="coversheet">
                    <option value="N">N</option>
                    <option value="Y">Y</option>
                </select><br>
                Color Ink? <input type="checkbox" name="ink" value="C"><br>
                Return Envelope? <input type="checkbox" name="returnenv" value="Y"><br>
                PreAuth? <input type="checkbox" name="preauth" value="1"><br>
                Debug: <select name="debug">
                    <option value="3">Most Verbose</option>
                    <option value="2">Somewhere in the middle</option>
                    <option value="1">Least Verbose</option>
                    <option value="">None</option>
                </select><br><br>
                <input type="submit" name="submit" value="Submit Job">
            </form>
        </fieldset>

        <fieldset id="multifile">
            <legend>Multi File Upload w/ Options</legend>
            <form method="post" enctype="multipart/form-data">
                multi<input name="multi_file" type="file"><br>
                <input type="submit" name="submit">
            </form>
        </fieldset>
    </div>

    <div class="column">
        <fieldset id="multifile">
            <legend>Data Retreival Options</legend>
            <br>
            <h4>By certified tracking number</h4>
            <form method="post" enctype="multipart/form-data">
                <select name="cert">
                    <option value="sig">signature file</option>
                    <option value="track">tracking info</option>
                    <option value="trackx">tracking info xml</option>
                    <option value="proof">proof</option>
                </select>
                tracking #<input type="text" name="trackingid" value="71147344282010607726" size="25"><br>
                <input type="submit" name="submit" value="submit">
            </form>
            <br>
            <h4>By unique recipient id</h4>
            <form method="post" enctype="multipart/form-data">
                <select name="cert">
                    <option value="proof">proof</option>
                    <option value="sig">signature file</option>
                    <option value="track">tracking info</option>
                    <option value="trackx">tracking info xml</option>
                </select>
                doc id<input type="text" name="doc_id" value="1351115816"><br>
                <input type="submit" name="submit" value="submit">
            </form>
            <br>
            <h4>Do Job Auth</h4>
            <form method="post" enctype="multipart/form-data">
                do auth<input type="text" name="doauth" value="55550a5048bf6110d2bd9c2b7c410107"
size="40"><br>
                <input type="submit" name="submit" value="submit">
            </form>
            <br>
            <h4>Check Account Status</h4>
            <form method="post" enctype="multipart/form-data">
                <input type="hidden" name="accountstatus" value="1"><br>
            <input type="submit" name="submit" value="submit">
         </form>
      </fieldset>
   </div>
   <?
}
?>
```

JSON

If you prefer return data to be presented in json format please include an argument key/value pair of “responseformat=json”

USPS TRACKING NUMBERS

Starting March 2018 the USPS tracking numbers that we return will take on a new format and will contain 22 digits. Prior to this, our tracking numbers were 20 digits in length. The 20 digit numbers remain valid for requesting signatures and tracking information.

Legacy version looks like – 71147344282010607726

New version looks like - 9207190142980468301468