List View API v1

Document Revision 1.0
Date of Issue: 05/10/2023

Fred Haselton
Business Analyst

Table of Contents

1. Purpose

To provide the API end point information and examples of the List View API v1. The web service handles the POST method to facilitate the recalling of line metadata from a specified list.

2. Glossary of Terms

Term	Meaning
listID	Identifier for each individual product list uploaded to the Liv-ex system, created via the List Manager POST API
lineID	Identifier for each individual line within the list, created via the Line Manager POST API

3. Technical Standards

- Permitted users will be issued with a unique token (CLIENT_KEY) and password (CLIENT_SECRET) combination to control the access for all the web services covered under Exchange Integration.
- The web services will consume and produce both XML and JSON. The user can provide the contents type in the request header. If the user does not provide any information, then the default content type will be JSON.
- The project will support ISO 8601.
- The project will only support HTTPS protocol for client and server communications.
- The API will support the following methods:
    - POST for create operation
- Pretty printing for output readability only is supported if required.
- Compression for bandwidth savings are used.
- For HTTP users who can only work on GET & POST methods, we provide a Header 'X-HTTP-Method-Override' for DELETE
- Authentication mechanism will be custom based on CLIENT_KEY and CLIENT_SECRET.
- For any PUSH services we require a direct POST URL which should be backed by as service capable of accepting and process XML payload as POST request.
- The APIs will be accessible at https://api.liv-ex.com/ followed by their specific base URIs.

4. Request Header

This information will be used to authenticate valid access to the REST API. Each user will have to provide the following information in the request header. Please note that the API expects the 4 headers as listed within this documentation and submitting a request with additional headers may lead to errors and/or failed responses.

Name	Mandatory	Description
CLIENT_KEY	Y	A valid user GUID which will be unique for each merchant
CLIENT_SECRET	Y	Password/Secret for the user CLIENT_KEY
ACCEPT	Y	Accept header is a way for the user to specify the media type of the response content it is expecting. The values for the content type will be application/json or application/xml. If no/invalid content type is found in the request, then JSON format will be used by default
CONTENT-TYPE	Y For POST requests	Content-type is a way to specify the media type of request being sent from the client to the server. The values for the content type will be application/json or application/xml. If no/invalid content type is found in the request, then JSON format will be used by default.

Example Header (JSON)

```
CLIENT_KEY: 12A34BC56-DE7F-89G0-H1J2345K678L
CLIENT_SECRET: dummy_password
ACCEPT: application/json
CONTENT-TYPE: application/json
```

Invalid header (JSON response)

```
{
    "status": "Unauthorized",
    "httpCode": "401",
    "message": "Unauthorized",
    "internalErrorCode": null,
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1550676412005,
        "provider": "Liv-ex"
    }
}
```

Invalid header (XML response)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://aby-uat-api.liv-ex.com/v1 https://aby-uat-api.liv-
ex.com/schema/v1/services.xsd">
    <Status>Unauthorized</Status>
    <HttpCode>401</HttpCode>
    <Message>Unauthorized</Message>
    <InternalErrorCode xsi:nil="true"/>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2019-02-20T15:28:48.623Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>

</Response>
```

5. API Listing

5.1 View the line data of a list – POST method

Description

This service will be used to recall the line information from a specified list, based on the input listID. A successful POST request will be responded with the metadata of the lineID's within the list.

Base URI

listAnalysis/v1/listView

URI Pagination Parameters

Name	Mandatory	Description
?limit	N default = 50	Values: Any value between '1' and '10,000'. Any provided value above 10,000 will be defaulted to 10,000. A "value" is counted as one line from the list. **Type:** integer
?offset	N default = 1	**Type:** alphanumeric

Example base URI including pagination: /listAnalysis/v1/listView?offset=1&limit=25

Request parameters

Name	Mandatory	Description
listID	Y	The ListID (created from List Manager POST method) that you would like to call line metadata from Type: 128-bit hexadecimal
lineID	N Default = all lines from listID specified	The LineID(%s) (created from Line Manager POST method) that you would like to view the metadata of Type: array
matchStatus	N Default = all	LWIN match status to show whether the LWIN of this line is valid Type: alphanumeric Values: all, matched, unmatched

Response parameters

Name	Description
listID	List ID Type: 128-bit hexadecimal
linesTotal	Total number of lines with a “live” status associated with the provided listID Type: integer
linesMatched	Total number of lines with a “live” status associated with the provided listID matched to LWIN Type: integer
linesUnmatched	Total number of lines with a “live” status associated with the provided listID not matched to LWIN Type: integer
lineID	Unique key of an individual line Type: 128-bit hexadecimal
value	Cell value Type: alphanumeric
lineStatus	The status of the line Type:alphanumeric

matchStatus	The status of the line – whether it has been matched to LWIN or not Type: alphanumeric
matchedLwin	The LWIN code that the line has been matched to Type: alphanumeric
matchUpdateDate	Date when LWIN match was last updated Type: datetime / EPOCH for JSON
lwinName	The LWIN display name Type: alphanumeric
vintage	The vintage of the wine Type: alphanumeric
bottleSize	Bottle size in ml e.g. 00750 or 01500 Type: alphanumeric
packSize	Size of pack e.g. 06 or 12 Type: alphanumeric
lxHeader	Liv-ex header assignment Type: alphanumeric
userHeader	User provided column names Type: alphanumeric
yourProductID	Provided product code by the customer Type: alpanumeric
inputLwin	The LWIN value that was provided when creating or editing a line Type: alphanumeric

Sample Request Body

JSON Request

```
{
  "listView": {
    "listID": "0bbcf74f-4eeb-4f9d-bea0-366d16c9dde8",
    "lineID":[
    "dca50c63-fc09-4b97-a7c8-72ec552643b2",
    "6412badb-e7ec-40c8-a63c-1f241e2d33f3"
    ],
    "matchStatus":"all"
  }
```

```
}
```

XML Request

```
<root>
  <listView>
    <listID>0bbcf74f-4eeb-4f9d-bea0-366d16c9dde8</listID>
    <lineID>dca50c63-fc09-4b97-a7c8-72ec552643b2</lineID>
    <lineID>6412badb-e7ec-40c8-a63c-1f241e2d33f3</lineID>
    <matchStatus>all</matchStatus>
  </listView>
</root>
```

Sample Response body

JSON Response


```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1689787204861,
    "provider": "Liv-ex"
  },
  "pageInfo": {
    "totalResults": 3,
    "limit": 10000,
    "offset": 1
  },
  "listViewResponse": {
    "listID": "0bbcf74f-4eeb-4f9d-bea0-366d16c9dde8",
    "listLink": "https://lon-qa-app.liv-ex.com/#/list-studio?listName=Testing%2012345",
    "linesTotal": 3,
    "linesMatched": 2,
    "linesUnmatched": 1,
    "lines": [
      {
        "lineID": "dca50c63-fc09-4b97-a7c8-72ec552643b2",
        "lineStatus": "live",
        "matchStatus": "true",
        "yourProductID": "SKU123456",
```

```
      "inputLwin": "10118722010",
      "matchedLwin": "10118722010",
      "lwinName": "Chateau Lafite Rothschild Premier Cru Classe, Pauillac",
      "vintage": "2010",
      "customLineData": [
        {
          "lxHeader": [
            "vintage"
          ],
          "userHeader": "vintage",
          "value": "2017"
        },
        {
          "lxHeader": [
            "wine"
          ],
          "userHeader": "wine",
          "value": "chateaux lafite roth"
        }
      ],
      "errors": {
        "error": []
      }
    },
    {
      "lineID": "6412badb-e7ec-40c8-a63c-1f241e2d33f3",
      "lineStatus": "live",
      "matchStatus": "false",
      "yourProductID": "SKU123456",
      "inputLwin": "1234567",
      "customLineData": [
        {
          "lxHeader": [
            "vintage"
          ],
          "userHeader": "vintage",
          "value": "2017"
        },
        {
          "lxHeader": [
            "wine"
          ],
          "userHeader": "wine",
          "value": "chateaux lafite roth"
```

```
            }
          ],
          "errors": {
            "error": []
          }
        }
      ]
    },
    "errors": null
}
```


XML Response


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<root>
  <Status>OK</Status>
  <HttpCode>200</HttpCode>
  <Message>Request completed successfully</Message>
  <InternalErrorCode>R001</InternalErrorCode>
  <ApiInfo>
    <Version>1.0</Version>
    <Timestamp>2023-07-19T17:23:24.005Z</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <pageInfo>
    <totalResults>3</totalResults>
    <limit>10000</limit>
    <offset>1</offset>
  </pageInfo>
  <listViewResponse>
    <listID>0bbcf74f-4eeb-4f9d-bea0-366d16c9dde8</listID>
    <listLink>https://lon-qa-app.liv-ex.com/#/list-studio?listName=Testing%2012345</listLink>
    <linesTotal>3</linesTotal>
    <linesMatched>2</linesMatched>
    <linesUnmatched>1</linesUnmatched>
    <lines>
      <lineID>dca50c63-fc09-4b97-a7c8-72ec552643b2</lineID>
      <lineStatus>live</lineStatus>
      <matchStatus>true</matchStatus>
      <matchUpdateDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
      <yourProductID>SKU123456</yourProductID>
      <inputLwin>10118722010</inputLwin>
      <matchedLwin>10118722010</matchedLwin>
```

```
        <lwinName>Chateau Lafite Rothschild Premier Cru Classe, Pauillac</lwinName>
        <vintage>2010</vintage>
        <customLineData>
          <lxHeader>vintage</lxHeader>
          <userHeader>vintage</userHeader>
          <value>2017</value>
        </customLineData>
        <customLineData>
          <lxHeader>wine</lxHeader>
          <userHeader>wine</userHeader>
          <value>chateaux lafite roth</value>
        </customLineData>
        <errors/>
      </lines>
      <lines>
        <lineID>6412badb-e7ec-40c8-a63c-1f241e2d33f3</lineID>
        <lineStatus>live</lineStatus>
        <matchStatus>false</matchStatus>
        <matchUpdateDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
        <yourProductID>SKU123456</yourProductID>
        <inputLwin>1234567</inputLwin>
        <customLineData>
          <lxHeader>vintage</lxHeader>
          <userHeader>vintage</userHeader>
          <value>2017</value>
        </customLineData>
        <customLineData>
          <lxHeader>wine</lxHeader>
          <userHeader>wine</userHeader>
          <value>chateaux lafite roth</value>
        </customLineData>
        <errors/>
      </lines>
    </listViewResponse>
  </root>
```


6. Response Codes

This section describes the response codes that will be returned by the Exchange Integration services.

Code	Message
R000	Request was unsuccessful

R001	Request completed successfully
R002	Request partially completed

6.1 Request validation error codes

Code	Error message	Trigger event
V018	Mandatory field missing(%).	When API request is submitted with no listID request parameter
V174	Invalid/incorrect listID:[%]. Please provide a valid listID value	Where listID request parameter specifies a listID that is not linked to the user account
V185	Invalid/incorrect lineID: [%]. Please provide a valid lineID value	Where lineID request parameter specifies a lineID that is not linked to the listID provided

6.2 HTTP Status codes

HTTP defines a bunch of meaningful status codes that can be returned from our API. These can be leveraged to help our API Merchants/consumers route their responses accordingly:

Code	Message
200 OK	Response to a successful GET, POST, PUT, DELETE. Can also be used for a POST that doesn't result in a creation.
201 Created	Response to a POST that results in a creation.
202 Accepted	The request has been accepted and will be processed later. It is a classic answer to asynchronous calls (for better UX or performances).
204 No Content	Response to a successful request that won't be returning a body (like a DELETE request)
400 Bad Request	The request is malformed, such as if the body does not parse
401 Unauthorized	When no and/or invalid authentication details are provided. Can also be used to trigger an auth popup if API is used from a browser
403 Forbidden	When authentication succeeded but authenticated user doesn't have access to the resource
404 Not Found	When a non-existent resource is requested
405 Method Not Allowed	When an HTTP method is being requested that isn't allowed for the authenticated user
406 Not Acceptable	Nothing matches the Accept-* Header of the request. As an example, you ask for an XML formatted resource, but it is only available as JSON.
409 Conflict	Indicates one or more supplied parameters are triggering a validation error. A relevant TR code should be returned in the response.
410 Gone	Indicates that the resource at this end point is no longer available. Useful as a blanket response for old API versions
415 Unsupported Media Type	If incorrect content type was provided as part of the request

422 Unprocessable Entity	Used for validation errors. Should be used if the server cannot process the entity, e.g. if an image cannot be formatted or mandatory fields are missing in the payload.
429 Too Many Requests	When a request is rejected due to rate limiting
500 Internal Server Error	The general catch-all error when the server-side throws an exception. The request may be correct, but an execution problem has been encountered at our end.