LIV | EX

THE FINE WINE MARKET

List Tally 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 Tally API v1. The web service handles POST method to facilitate the retrieval of information about the user's lists.

2. Glossary of Terms

Term	Meaning
LWIN	LWIN - the Liv-ex Wine Identification Number – serves as a universal wine identifier for the wine trade. LWIN is a unique seven to eighteen-digit numerical code that can be used to quickly and accurately identify a product. LWIN allows wine companies to keep their preferred naming system, while introducing a new universal code.
listID	Identifier for each individual product list uploaded to the Liv-ex system

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.
- 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.

Param

Name	Mandatory	Description
CLIENT_KEY	Y	A valid merchant GUID which will be unique for each merchant
CLIENT_SECRET	Y	Password/Secret for the merchants CLIENT_KEY
ACCEPT	Y	Accept header is a way for a client to specify the media type of the response content it is expecting. Th 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 Fetch List Information Service – POST method

Description

This API will be used to fetch information about all lists that belong to the calling customer or the individual calling user, based on the request parameters used. A successful POST request will be responded with the metadata of the listID's being called.

Base URI

listAnalysis/v1/listTally

Request parameters

Name	Mandatory	Description
createdBy	N Default = "all"	List filter. Enables querying for all custom lists created within your organisation or only for lists created with your user credentials. Values: "all", "my lists" Type: alphanumeric

Response parameters:

Name	Description
listID	Unique identifying key attached to each list, created using the List Manager API POST method Type: 128-bit hexadecimal
listName	List name (assigned list name) Type: alphanumeric
listType	Type of the list specified by the caller (e.g. product list, custom list) Type: alphanumeric
listStatus	All listStatus will show as live Type: alphanumeric
linesTotal	Total number of lines associated with a given list Type: integer
linesUnmatched	Total number of lines not matched to LWIN codes (any level) Type: integer
linesMatched	Total number of lines matched to LWIN codes (any level) Type: integer
createdDate	Type: datetime (ISO 8601) / epoch (if json)
lastModifiedDate	Last time the list was modified (lines added/deleted/edited, list created/edited) Type: datetime / epoch
lastAccessedDate	Last time that the list was accessed Type: datetime / epoch
lwinRefreshDate	Type: datetime (ISO 8601) / epoch (if json)
newMatches	Number of new LWIN matches since the last session
createdBy	Name of the user who created the list. First + Last name of the user Type: alphanumeric
note	Note provided by the user Type: alphanumeric
totalLists	Number of live lists belonging to the merchant
matchingLists	Number of lists matching matching current filters

Sample Request Body

JSON Request

```
{
   "listTallyRequest": {
      "createdBy": "my lists"
   }
}
```

XML request

```
<root>
   <listTally>
      <createdBy>my lists</createdBy>
   </listTally>
</root>
```

Sample Response Body

JSON Response

```
{
   "status": "OK",
   "httpCode": "200",
   "message": "Request completed successfully",
   "internalErrorCode": "R001",
   "apiInfo": {
      "version": "1.0",
      "timestamp": 1690544004006,
      "provider": "Liv-ex"
   },
   "listTallyResponse": {
      "totalLists": 83,
      "matchingLists": 47,
      "lists": [
         {
            "listID": "cd347e26-33dc-4fb2-849c-767d27d85895",
            "listName": "Testing 12345 (6)",
            "listStatus": "live",
            "listType": "Custom List",
            "linesTotal": 7,
            "linesUnmatched": 4,
            "linesMatched": null,
            "createdDate": 1690285180750,
            "lastModifiedDate": 1690476767712,
            "lastAccessedDate": 1690285180750,
            "lwinRefreshDate": null,
            "newMatches": null,
            "createdBy": "Fred Haselton",
            "note": "line manager post and patch testing"
        },
        {
            "listID": "89a3fef2-80ec-4a39-82e3-e17fa770b4cf",
            "listName": "My List (21)",
            "listStatus": "live",
            "listType": "Custom List",
            "linesTotal": 0,
            "linesUnmatched": 0,
            "linesMatched": null,
            "createdDate": 1690465643046,
            "lastModifiedDate": 1690465643046,
            "lastAccessedDate": 1690465643046,
            "lwinRefreshDate": null,
            "newMatches": null,
            "createdBy": "Fred Haselton",
            "note": "These note are to describe the origin of the list"
        }
      ]
    },
    "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>2023-07-28T11:31:18.057Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <listTallyResponse>
    <totalLists>83</totalLists>
    <matchingLists>83</matchingLists>
    <lists>
      <listID>cd347e26-33dc-4fb2-849c-767d27d85895</listID>
      <listName>Testing 12345 (6)</listName>
      <listStatus>live</listStatus>
      <listType>Custom List</listType>
      <linesTotal>7</linesTotal>
      <linesUnmatched>4</linesUnmatched>
      <createdDate>2023-07-25T11:39:40.750Z</createdDate>
      <lastModifiedDate>2023-07-27T16:52:47.712Z</lastModifiedDate>
      <lastAccessedDate>2023-07-25T11:39:40.750Z</lastAccessedDate>
      <createdBy>Fred Haselton</createdBy>
      <note>line manager post and patch testing</note>
    </lists>
    <lists>
      <listID>89a3fef2-80ec-4a39-82e3-e17fa770b4cf</listID>
      <listName>My List (21)</listName>
      <listStatus>live</listStatus>
      <listType>Custom List</listType>
      <linesTotal>0</linesTotal>
      <linesUnmatched>0</linesUnmatched>
      <createdDate>2023-07-27T13:47:23.046Z</createdDate>
      <lastModifiedDate>2023-07-27T13:47:23.046Z</lastModifiedDate>
      <lastAccessedDate>2023-07-27T13:47:23.046Z</lastAccessedDate>
      <createdBy>Fred Haselton</createdBy>
      <note>These note are to describe the origin of the list</note>
    </lists>
  </listTallyResponse>
</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

There are no validations for this API as we will apply default values when a request in incorrect.

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.