LIV|EX

THE FINE WINE MARKET

ABV Bulk Data API v1

Document Revision 1.0
Date of Issue: 04 April 2024
Date of revision: 04 April 2024

Barnabas Mullan
Senior Business Analyst

Table of Contents

1. Purpose

To provide the API end point information and examples of the web services available for the ABV Bulk Data API.

2. 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 read 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 a service capable of accepting and processing an XML payload as a POST request.
- The APIs will be accessible at https://api.liv-ex.com/ followed by their specific base URIs

3. 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. 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.

e.g.

Example header

```
CLIENT_KEY: 94B5CC70-BC3D-49C3-B636-C3C7552E543D
CLIENT_SECRET: merchantpasswd
ACCEPT: application/json
CONTENT-TYPE: application/json
```

Invalid header JSON response

```
{
    "status": "Unauthorized",
    "httpCode": "401",
    "message": "Request was unsuccessful",
    "livexCode": "R000"
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1518524979121,
        "provider": "Liv-ex"
    }
}
```

Invalid header XML response

```
<Response>
    <Status>Unauthorized</Status>
    <HttpCode>401</Code>
    <Message>Request was unsuccessful.</Message>
    <LivexCode>R001</LivexCode>
     <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2021-07-01T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
<Response>
```

4. API Listing

4.1 ABV Bulk Data service (POST method)

Description

This service will be used to retrieve a list of ABV values for LWIN11s, based either on provided LWINs or an input listID. A successful POST request will respond with de-duplicated LWIN11s, meaning that the number of results may vary from the number of lines within your listID.

Base URI

/abv/data/v1/abvBulkData

Request parameters

Name	Mandatory	Description
?limit	N default = 50	Values: Any value between '1' and '10000' **Type:** integer
?offset	N default = 1	**Type:** alphanumeric

Example base URI including pagination: /abv/data/v1/abvBulkData?offset=1&limit=25

The response contains a "pageInfo" element that states:

1. totalResults: The number of unique LWIN11s returned.
2. limit: as per the request, a maximum value of 10,000.
3. offset: the current 'page' of results. Count starts from 1.

Request parameters

Name	Mandatory	Description
lwin	N (unless listID is not provided)	List of LWINs. Possible LWIN lengths: 11/16/18. Multiple different LWIN lengths can be supplied in a singular request. Max limit: 1500 Type: Alphanumeric array
listID	N (unless LWIN is not provided)	Custom ListID or 'stock list' Type: Alphanumeric, case insensitive
includeUnverified	N (default value: False)	When true, the API will include ABVs which have not been verified by Liv-ex. When false, the API will only include responses for verified ABVs. Boolean: True/False

Response parameters

Name	**Description**
listID	List ID value if specified in the request Type: 128-bit hexadecimal
lwin	Returns the LWIN11 Type: alphanumeric
alcoholValue	Alcohol by volume value for the given wine and vintage Type: double Example "13.5"
isVerified	Indicates whether the alcohol value provided has been verified by the Liv-ex warehouse team. Physically verified values will have isVerified = true

	Type: Boolean true / false
lastUpdateDate	Type: Epoch time if JSON Example (JSON): "1549537950898"

Sorting order:

When LWINs have been provided in the request, the response will be returned respecting the input order, after de-duplication.

When a valid listID is provided, the response will be returned respecting the order of LWIN11s in the list, after de-duplication.

Sample Requests

JSON

```
{
    "abvBulkData": {
        "listID": "Stock list",
        "includeUnverified": true
    }
}
```

XML

```
<root>
    <abvBulkData>
        <lwin>10071012000</lwin>
        <lwin>1007101202000750</lwin>
        <includeUnverified>false</includeUnverified>
    </abvBulkData>
</root>
```

Sample Responses

The response is sent per request. Where there is no matching data that can be found, but the LWIN is valid, the API will return NULL results. Invalid LWINs will not be included in the API response.

JSON Response

```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
```

```
        "apiInfo": {
            "version": "1.0",
            "timestamp": 1712311778486,
            "provider": "Liv-ex"
        },
        "pageInfo": {
            "totalResults": 207,
            "limit": 50,
            "offset": 1
        },
        "abvBulkDataResponse": {
            "listID": "Stock list",
            "abvBulkData": [
                {
                    "lwin": "28528322021",
                    "alcoholValue": null,
                    "isVerified": null,
                    "lastUpdateDate": null
                },
                {
                    "lwin": "10340622018",
                    "alcoholValue": "13.5",
                    "isVerified": "false",
                    "lastUpdateDate": 1678118733000
                }
            ]
        },
        "errors": null
    }
```


XML Response

The response is sent per request. The example below shows the response for the XML Request example.


```
<?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>2024-04-04T15:55:20.033Z</Timestamp>
        <Provider>Liv-ex</Provider>
```

```
    </ApiInfo>
    <pageInfo>
        <totalResults>2</totalResults>
        <limit>50</limit>
        <offset>1</offset>
    </pageInfo>
    <abvBulkDataResponse>
        <abvBulkData>
            <lwin>10071012000</lwin>
            <alcoholValue>13.0</alcoholValue>
            <isVerified>true</isVerified>
            <lastUpdateDate>2020-10-07T19:48:03Z</lastUpdateDate>
        </abvBulkData>
        <abvBulkData>
            <lwin>10071012020</lwin>
            <alcoholValue>13.5</alcoholValue>
            <isVerified>true</isVerified>
            <lastUpdateDate>2022-11-23T15:20:12Z</lastUpdateDate>
        </abvBulkData>
    </abvBulkDataResponse>
</root>
```


5. 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

5.1 Request Validation error codes

Code	Message
V018	Mandatory field missing [%s].
V058	Invalid / incorrect LWIN: [${v}]. Please provide a valid LWIN11, LWIN16 or LWIN18 code.
V174	Invalid/incorrect listID: [%]. Please provide a valid listID value.

5.2 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.