CellarView2 v2 API Documentation

Document Revision 1.1
Date of Issue: 18 May 2017
Date of revision: 19 February 2019

Supriya Neewoor

Product Manager

Table of Contents

1. Purpose

To provide the API endpoint information and examples of the Cellar View 2 v2 web service. The web service handles GET & POST methods to retrieve merchants' stock holding information. The changes in v2 enable retrieval of large stock holdings so that stock systems can be integrated with Liv-ex.

2. Glossary of Terms

Term	Meaning
L-WIN	L-WIN - the Liv-ex Wine Identification Number – serves as a universal wine identifier for the wine trade. L-WIN is a unique seven to eighteen-digit numerical code that can be used to quickly and accurately identify a product. L-WIN allows wine companies to keep their preferred naming system, while introducing a new universal code.
Wine	The word wine below is referring to a specific wine (the producer and brand, grape or vineyard), vintage and unit size combination.
Bid	A buyer places a bid on the Exchange for buying a certain amount of wine.
Offer	A seller places an offer on the Exchange for selling a certain amount of wine.
Order	Order is a generic term for both bid/offer.
Market Price	The Market Price is based on the cheapest 6 and 12-pack prices advertised by leading merchants in the EU and Switzerland. (Where appropriate, alternative unit sizes are used for the calculation.) It provides a guide as to the price you are likely to pay for SIB-compliant stock in the market.
SIB	Standard in Bond trade terms: http://www.liv-ex.com/staticPageContent.do?pageKey=Rules_and_Regulations
SEP	Standard En Primeur: http://www.liv-ex.com/staticPageContent.do?pageKey=Rules_and_Regulations
Contract Type	Contract type is a generic term for SIB, SEP or Special (X).
Trade	A bid and offer match for a trade to take place on the Exchange for a certain amount of wine.
UID	UID is Liv-ex's unique identification number allocated to a case of wine in the Vine warehouse.
In Bond (IB)	Wines 'in bond' have not yet had the Duty and VAT paid on them. They must be stored in a bonded warehouse approved by HM Customs & Excise.
Duty Paid (DP)	Purchased wines which have passed through customs, with UK Duty and VAT paid on them.

3. Technical Standards

- Permitted users will be issued with a unique token (CLIENT_KEY) and password (CLIENT_SECRET) combination to control the access to the web service.
- The web services will consume and produce both XML and JSON. The user can provide the

content 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's will support the following methods:
    1. POST for create operation
    2. GET for read operation
    3. PUT for update operation
    4. DELETE for delete 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 PUT & 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 a 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

The data within the request header will be used to authenticate valid access to the REST API.

Note:

Each user will have to provide the following information in the request header of all API listings in this document.

Param

Name	Mandatory	Description
CLIENT_KEY	Y	A valid merchant GUID / token which is unique for each merchant.
CLIENT_SECRET	Y	Password/Secret for the merchant's access.
ACCEPT	N	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 for the request by default.

e.g.

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": "Unauthorized"
        "internalErrorCode": null
        "apiInfo": {
                "version": "2.0"
                "timestamp": "2016-07-08T17:23:54.859+01:00"
                "provider": "Liv-ex"
        }
}
```

Invalid header XML response

```
<Response>
        <Status>Unauthorized</Status>
        <HttpCode>401</HttpCode>
        <Message>Unauthorized</Message>
        <InternalErrorCode xsi:nil="true" />
         <ApiInfo>
                <Version>2.0</Version>
                <Timestamp>2016-07-08T17:23:54.859+01:00</Timestamp>
                <Provider>Liv-ex</Provider>
         </ApiInfo>
 </Response>
```

5. API Listing

5.1 Cellar View 2 v2 Service - GET Method

Description

This service retrieves all the stock (UIDs records) in a merchant's account. It will return the array of UIDs with a subset of properties per UID.

Base URI

logistics/v2/cellarView2

Note:

Pagination: If the number of UID records in the response of a GET request is above our maximum set per page, pagination becomes relevant**.**

The initial response will return the first page and specify the Page Info properties within the header of the response body so that subsequent calls can be made to retrieve remaining pages and UID records. E.g. totalRecords, totalPages, noOfRecords on the current page, and currentPage number.

Using the current and total number of pages, subsequent GET requests should then request the page number in the URI to retrieve the remaining UID records e.g. logistics/v2/cellarView2?page=2 *(replacing with the appropriate page number)*

Sample GET responses are given below.

JSON Response

Successful response with valid authentication (GET request)


```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully.",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "2.0",
    "timestamp": 1494511970365,
    "provider": "Liv-ex"
  },
  "pageInfo": {
    "totalRecords": 2,
    "totalPages": 1,
    "noOfRecords": 2,
    "currentPage": 1
  },
  "cellarViews": {
    "cellarViews": [
      {
        "uid": 407291,
        "lwin": 110633820080600750,
        "subAccount": "ABCD",
        "buyerRef": " PO345"
      },
      {
        "uid": 407292,
        "lwin": 110633820080600750,
        "subAccount": "ABCD",
        "buyerRef": " PO345"
      }
    ]
  },
  "error": {
    "error": null
  }
}
```


Response with no access to service

```
{
  "status": "Unauthorized",
  "httpCode": "403",
  "message": "Forbidden",
  "internalErrorCode": null,
  "apiInfo": {
    "version": "2.0",
    "timestamp": 1468941366104,
    "provider": "Liv-ex"
  }
}
```

XML Response

Successful response with valid authentication

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cellarViewResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://api.liv-ex.com/v1 https://api.liv-
ex.com/schema/v1/services.xsd">
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully.</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>2.0</Version>
        <Timestamp>2017-05-11T15:04:24.497+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <pageInfo>
        <totalRecords>2</totalRecords>
        <totalPages>1</totalPages>
        <noOfRecords>2</noOfRecords>
        <currentPage>1</currentPage>
    </pageInfo>
    <cellarViews>
        <cellarView>
            <uid>407291</uid>
            <lwin>110633820080600750</lwin>
            <subAccount>ABCD</subaccount>
            <buyerRef> PO345</buyerRef>
        </cellarView>
        <cellarView>
            <uid>407292</uid>
            <lwin>110633820080600750</lwin>
            <subAccount>ABCD</subaccount>
            <buyerRef>PO345</buyerRef>
        </cellarView>
</cellarViews>
    <error/>
</cellarViewResponse>
```

5.2 Cellar View 2 v2 Service - POST Method

A POST request will return more detail or properties for each UID record. The POST method should be used to request the status and details of a given lwin, sub account, buyer reference or UID.

For accounts that have a large amount of stock, it is recommended to use the GET method to retrieve all stock records initially, followed by POST method using the additional filter parameters (e.g. UID or sub account) to retrieve more detailed stock information. Other filter parameters such as lwin, and buyer ref can be used individually or in combination when using the POST method.

Sample POST requests and responses are given below.

Param

Name	Mandatory*	Description
lwin	N	A valid LWIN7, LWIN11, L-WIN16 or LWIN18.
subAccount	N	Merchant sub account
buyerRef	N	The merchant's reference for the purchased stock. The value is a string of up to 30-character length that was initially associated to the order/trade by the merchant. Can also be the value submitted as 'poNumber' in pre-advice API.
uid	N	The unique identification number allocated to a case of wine in the Liv-ex warehouse.

*At least one of the filter parameters must be provided in the body of a POST request.

Sample JSON Request Body

```
{"stockMovement":{"lwin":" 110633820080600750","subAccount":"ABCD", "buyerRef":"PO345",
"uid":"1234567"}}
```

Sample XML Request Body

```
<root>
    <stockMovement>
            <lwin>100604520041200750</lwin>
           <subAccount>ABCD</subAccount>
           <buyerRef>PO345</buyerRef>
      <uid>1234567</uid>
    </stockMovement>
</root>
```

JSON Response

The Cellar View 2 service will respond with HTTP Code 200 - OK in a successful response.

Succesful JSON response with valid authentication

```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully.",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "2.0",
    "timestamp": 1494513509348,
    "provider": "Liv-ex"
```

```
  },
  "cellarViews": {
    "cellarViews": [
      {
        "midCode": "Test",
        "uid": 1234567,
        "passportstatus": "Approved",
        "passportValidTill": "07/11/2019",
        "vTrans": 255401,
        "lxTrade": null,
        "rotation": "16/0134342",
        "warehouseStatus": "A",
        "subAccount": "ABCD",
        "warehouse": "TIL",
        "wineName": "Vietti, Barolo Lazzarito",
        "vintage": "2008",
        "unitSize": "6x75",
        "unitPrice": 476,
        "unitCurrency": "GBP",
        "lwin": "110633820080600750",
        "arrivalDate": "03/11/2016",
        "contract": "IB",
        "buyerRef": " PO345",
        "supplier": "73880",
        "payment": "Not Paid",
        "transactionNotesVisibleToBuyer": null,
        "uidNotesVisibleToBuyer": null,
        "photosLink": {
          "photos": [
            {
              "lowResUrl": "https://staging.api.liv-
ex.com/cellarViewPhotoService/v1/getImage?_1494513509348&token=120121921262&photoId=23137
9&isHighRes=false",
              "highResUrl": "https://staging.api.liv-
ex.com/cellarViewPhotoService/v1/getImage?_1494513509348&token=120121921262&photoId=23137
9&isHighRes=true"
            }
          ]
        },
        "bestBid": null,
        "bidCurrency": null,
        "bestOffer": null,
        "offerCurrency": null,
        "marketPrice": 385,
        "marketPriceCurrency": "GBP",
        "marketPriceDate": "09/12/2016",
        "lastTraded": null,
        "lastTradedCurrency": null,
        "lastTradedDate": null
      }
    ]
  },
  "error": {
    "error": null
  }
}
```


Note:

Warehouse Status: **A** - In Account (stock that is readily available in merchant's Vine account); **B** - Booking in (stock that has been received in the Vine network but is pending check or payment).

Response with invalid sub account


```
{
  "status": "Bad Request",
  "httpCode": "400",
  "message": "Request was unsuccessful.",
  "internalErrorCode": "R000",
  "apiInfo": {
    "version": "2.0",
    "timestamp": 1469005719380,
    "provider": "Liv-ex"
  },
  "cellarViews": null,
  "error": {
    "error": [
      {
        "code": "V034",
        "message": "Sub Account does not exist."
      }
    ]
  }
}
```


XML Response

Successful response with valid authentication


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cellarViewResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://api.liv-ex.com/v1 https://api.liv-
ex.com/schema/v1/services.xsd">
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully.</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>2.0</Version>
        <Timestamp>2017-05-11T15:57:40.305+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <cellarViews>
        <cellarView>
            <midCode>Test</midCode>
            <uid>1234567</uid>
            <passportStatus> Approved</passportStatus>
            <passportValidTill>07/11/2019</passportValidTill>
            <vTrans>255401</vTrans>
            <lxTrade xsi:nil="true"/>
            <rotation>16/0134342</rotation>
            <warehouseStatus>A</warehouseStatus>
            <subAccount>ABCD<subAccount/>
            <warehouse>TIL</warehouse>
            <wineName> Vietti, Barolo Lazzarito</wineName>
            <vintage>2008</vintage>
            <unitSize>6x75</unitSize>
            <unitPrice>476</unitPrice>
            <unitCurrency>GBP</unitCurrency>
            <lwin>110633820080600750</lwin>
            <arrivalDate>03/11/2016</arrivalDate>
            <contract>IB</contract>
            <buyerRef>PO345</buyerRef>
            <supplier>73880</supplier>
            <payment>Not Paid</payment>
            <transactionNotesVisibleToBuyer xsi:nil="true"/>
            <uidNotesVisibleToBuyer xsi:nil="true"/>
            <photosLink>
```

```
                <photo>
                    <lowResUrl> https://staging.api.liv-
ex.com/cellarViewPhotoService/v1/getImage?_1494513509348&token=120121921262&photoId=23137
9&isHighRes=false</lowResUrl>
                    <highResUrl> https://staging.api.liv-
ex.com/cellarViewPhotoService/v1/getImage?_1494513509348&token=120121921262&photoId=23137
9&isHighRes=true</highResUrl>
                </photo>
            </photosLink>
            <bestBid xsi:nil="true"/>
            <bidCurrency xsi:nil="true"/>
            <bestOffer xsi:nil="true"/>
            <offerCurrency xsi:nil="true"/>
            <marketPrice>385.0</marketPrice>
            <marketPriceCurrency>GBP</marketPriceCurrency>
            <marketPriceDate>09/12/2016</marketPriceDate>
            <lastTraded xsi:nil="true"/>
            <lastTradedCurrency xsi:nil="true"/>
            <lastTradedDate xsi:nil="true"/>
        </cellarView>
    </cellarViews>
    <error/>
</cellarViewResponse>
```


Invalid XML Response


```
<cellarViewResponse>
    <Status>Bad Request</Status>
    <HttpCode>400</HttpCode>
    <Message>Request was unsuccessful.</Message>
    <InternalErrorCode>R000</InternalErrorCode>
    <ApiInfo>
        <Version>2.0</Version>
        <Timestamp>2016-07-20T17:57:53.973+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <errors>
        <error>
            <code>V034</code>
            <message> subaccount does not exist </message>
        </error>
    </errors>
</cellarViewResponse>
```


6. Response Codes

This section describes the response codes that will be returned by the Cellar View.

Code	Message
R000	Request was unsuccessful
R001	Request completed successfully

6.1 Validation error codes

Code	Message

V002	Invalid parameter (subAccount / buyerRef)
V006	Invalid L-WIN number
V034	(subAccount / buyerRef) does not exist
V035	No records found
V036	Too many UIDs, please use filters

6.2 HTTP Status codes

HTTP defines a few of the meaningful status codes that can be returned from our API. These can be leveraged to help 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 or invalid authentication details are provided. Also useful to trigger an auth popup if the 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
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.