



Release API v2
(Deliver & Collection services)

Document Revision 7.0
Date of Issue: 15 July 2016
Date of revision: 29 May 2019

Supriya Neewoor

Product Manager

Table of Contents

1. Purpose	3
2. Glossary of Terms	3
3. Technical Standards	3
4. Request Header	4
5. API Listing	5
5.1 Release Service (POST Method)	5
6. Response Codes	9
6.1 Request validation error codes	9
6.2 HTTP Status codes	10

1. Purpose

To provide the API end point information and examples of the web services available for delivery or collection.

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.
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 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

content type in the request header. If the user does not provide any information, then the default content type will be JSON.

- The service supports ISO 8601.
- The service 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
- 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.

Parameter

Name	Mandatory	Description
CLIENT_KEY	Y	A valid GUID which will be unique for each user.
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.

Example header

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

Invalid header (JSON response)

```
{
  "status": "Unauthorized",
  "statusCode": "401",
  "message": "Request was unsuccessful",
  "livexCode": "R000"
  "apiInfo": {
    "version": "2.0",
    "timestamp": 1551628884,
    "provider": "Liv-ex"
  }
}
```

Invalid header (XML response)

```
<Response>
  <Status>Unauthorized</Status>
  <HttpCode>401</Code>
  <Message>Request was unsuccessful.</Message>
  <LivexCode>R001</LivexCode>
  <ApiInfo>
    <Version>2.0</Version>
    <Timestamp>2019-03-03T11:12:30</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
</Response>
```

5. API Listing

5.1 Release Service (POST Method)

Description

This service can be used to release wine cases from a merchant’s Vine account for delivery or collection.

The Type filter within the URL should be used to specify the type of release being requested (deliver or collection).

Base URI

logistics/v2/release?type=deliver

logistics/v2/release?type=collection

Request Parameters

Name	Mandatory	Description
UID (single or multiples)	Y	UID number(s) of the case of wine to be delivered or collected in the release order.
contract	Y	A valid contract type for the release. The possible value can be IB (In Bond) or DP (Duty Paid).
companyName	Y	Company name for Delivery or Transporter name for collection. Can be any string up to 255-character length.
contactName	Y	Contact name for Delivery or Collection. Can be any string up to 255-character length.
telephoneNo	Y	Telephone number of delivery contact or Transporter. Can be up to 20 digits in length. e.g. +447701234567
notes	N	Instruction notes for Delivery or Collection. Can be any string up to 255-character length.
email	Y	Email address of requesting merchant’s internal delivery contact. Must contain @ and can be up to 255-character length.
deliveryDate	Y for deliver	Format: dd/mm/yyyy Date delivery should take place. Must be a day in the future (for deliveries from Vine Tilbury please

		please reference the LCB delivery schedule)
address1	Y for deliver	First line of address for Delivery. Can be any string up to 255-character length.
address2	Y for deliver	Second line of address for Delivery. Can be any string up to 255-character length.
address3	N	Third line of address for Delivery. Can be any string up to 255-character length.
townOrCounty	Y for deliver	Town or County for Delivery. Can be any string up to 255-character length.
country	N	Country for Delivery. Can be any string up to 255-character length.
postcode	Y for deliver	Postcode for Delivery. Can be any string up to 10-character length.
exciseNumber	N	Excise number for Delivery
warehouseId	N	Warehouse ID for Delivery
collectionBehalfOf	N	Company name of the collection on behalf of. Can be any string up to 255-character length.

Sample Request Body

JSON Request (Deliver)

```
{
  "release": {
    "uids": {
      "uid": ["846938", "846939", "846940"]},
    "contract": "IB",
    "companyName": "Liv-ex Ltd",
    "contactName": "John Smith",
    "telephoneNo": "+442070628788",
    "notes": "Please call before arrival",
    "deliver": {
      "email": "john@liv-ex.com",
      "deliveryDate": "01/11/2017",
      "address1": "Studio 10, Battersea Studios",
      "address2": "82 Silverthorne Road",
      "address3": "Battersea",
      "townOrCounty": "London",
      "country": "United Kingdom",
      "postcode": "SW8 3HE"
    }
  }
}
```

JSON Request (Collection)

```
{
  "release": {
    "uids": {
      "uid": ["846938", "846939", "846940"]},
    "contract": "IB",
    "companyName": "To Your Door Ltd",
    "contactName": "Marilyn Roberts",
    "telephoneNo": "442070628788",
    "notes": "notes",
    "collection": {
      "collectionBehalfOf": "FW Ltd"
    }
  }
}
```

```
}

```

XML Request (Deliver)

```
<root>
<release>
  <uids>
    <uid>846938</uid>
    <uid>846939</uid>
    <uid>846940</uid>
  </uids>
  <contract>IB</contract>
  <companyName>Liv-ex Ltd</companyName>
  <contactName>John Smith</contactName>
  <telephoneNo>+442070628788</telephoneNo>
  <notes>notes</notes>
  <deliver>
    <email>john123@liv-ex.com</email>
    <deliveryDate>01/11/2017</deliveryDate>
    <address1> Studio 10, Battersea Studios</address1>
    <address2>82 Silverthorne Road</address2>
    <address3>Battersea</address3>
    <townOrCounty>London</townCounty>
    <country>United Kingdom</country>
    <postcode>SW8 3HE</postcode>
  </deliver>
</release>
</root>
```

XML Request (Collection)

```
<root>
<release>
  <uids>
    <uid>846938</uid>
    <uid>846939</uid>
    <uid>846940</uid>
  </uids>
  <contract>IB</contract>
  <companyName>Liv-ex Ltd</companyName>
  <contactName>John Smith</contactName>
  <telephoneNo>+442070628788</telephoneNo>
  <notes>notes</notes>
</release>
</root>
```

Sample Response Body

The Positions service will respond with HTTP Code 200 OK in a successful response.

JSON Response

The response is sent per request.

Response with a valid release order ID

```
{
  "status": "OK",
  "statusCode": "200",
  "message": "Request completed successfully.",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "2.0",
    "timestamp": 1469032145186,
    "provider": "Liv-ex"
  },
  "orderNo": "44602",
  "errors": null
}
```

Response with invalid UID

```
{
  "status": "Bad Request",
  "statusCode": "400",
  "message": "Request was unsuccessful.",
  "internalErrorCode": "R000",
  "apiInfo": {
    "version": "2.0",
    "timestamp": 1469032259636,
    "provider": "Liv-ex"
  },
  "orderNo": null,
  "errors": {
    "error": [
      {
        "code": "V019",
        "message": "UID does not exist or does not correspond to your account"
      }
    ]
  }
}
```

XML Response

The response is sent per request.

Response with a valid release order ID

```
<releaseResponse>
  <Status>OK</Status>
  <HttpCode>200</HttpCode>
  <Message>Request completed successfully.</Message>
  <InternalErrorCode>R001</InternalErrorCode>
  <ApiInfo>
    <Version>2.0</Version>
    <Timestamp>2016-07-20T17:38:09.695+01:00</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <orderNo>44603</orderNo>
</releaseResponse>
```


Response with invalid ID

```
<releaseResponse>
  <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>V019</code>
      <message>UID does not exist or does not correspond to your
account</message>
    </error>
  </errors>
</releaseResponse>
```

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

6.1 Request validation error codes

Code	Message
V000	Mandatory field missing.
V001	Merchant is not allowed to access the requested feed.
V002	Invalid parameter(s).
V019	UID does not exist or does not correspond to your account
V020	Unit(s) are currently being offered on Liv-ex. {uid}
V021	Transaction {lxtradeid} cannot be assigned from provided unit(s) there is mismatch pack size. Please contact your Vine Manager.
V022	Releases are currently suspended due to overdue invoices. Please contact the accounts team to arrange payment.
V023	You are on Resell only and have live offers on the following wines: {wineName}. Any wine on offer cannot be assigned.
V024	Unit(s) belong to unpaid Vtrans: {vtrans}. Please make a payment or remove unit from request in order to continue.
V025	You are exceeding your credit limit, Available credit = {amount}, either remove unpaid stock or make a payment before releasing stock.

V026	You have unpaid Vtrans: {vtrans}. Please contact your Vine manager.
V027	Web service only supports IB and DP as contract parameter.
V028	Unit(s) must have same tax status.
V029	Unit(s) must belong to same warehouse.
V030	Unit(s) have already been added to an existing release order.
V031	Some unit(s) contain DP tax status, please send DP as contract.
V032	Duty Paid release not allowed for non T1 tax code.
V037	Unit(s) must be from same Wine and pack size.

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.