LIV|EX

THE FINE WINE MARKET

Your Trade Details API v1

Document Revision 1.1
Date of Issue: 10 December 2020
Date of revision: 18 January 2021

Mathew McGill

Business Analyst

Table of Contents

1. Purpose

To provide the API end point information and examples of the web services available for Your Trade Details.

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 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
- 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

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.

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",
    "httpCode": "401",
    "message": "Request was unsuccessful",
    "livexCode": "R000"
    "apiInfo": {
        "version": "1.0",
        "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>1.0</Version>
        <Timestamp>2019-03-03T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
<Response>
```

4. API Listing

4.1 Your Trade Details service (POST method)

Description

This service can be used to request details of your Liv-ex trades.

Base URI

accounts/v1/yourTradeDetails

Request Parameters

Name	Mandatory	Description
lxTradeNo	Y	The Liv-ex reference number for the trade. Both versions ‘LX12345’ and '12345' for LX trade will be accepted Type: integer

Sample Request Body

JSON Request

```
{
 "yourTradeDetails":{
 "lxTradeNo": "LX207604"
 }
}
```

XML Request

```
<yourTradeDetails>
  <yourTradeDetail>
  <lxTradeNo>247930</lxTradeNo>
  </yourTradeDetail>
</yourTradeDetails>
```

Sample Response Body

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

Response parameters

Name	Description
lxTradeNo	The Liv-ex reference number for the trade. Type: integer
tradeDate	The timestamp of when the trade took place. Type: date (epoch time if JSON)
tradeStatus	States whether the trade was completed and Traded, or cancelled and deleted. Type: alphanumeric
orderGUID	The Liv-ex order identifier. Type: 128-bit alphanumeric
lwin	LWIN18 Type: alphanumeric
lwinName	Vintage-specific LWIN wine name description e.g. ‘Chateau Grand-Puy-Lacoste 5eme Cru Classe, Pauillac’. Type: alphanumeric
lwinRegion	Region of origin (where applicable, specific to local laws). Type: alphanumeric
vintage	The year in which the product was created. Type: alphanumeric
packSize	The LWIN pack size description e.g. '06'. Type: alphanumeric
bottleSize	The LWIN bottle size description e.g. '00750'. Type: alphanumeric
role	States the relation of the user to the trade, either buy or sell. Type: alphanumeric

userName	First name and last name of buyer or seller. Type: alphanumeric
merchantRef	Type: alphanumeric
contractType	SIB, SEP, X Type: alphanumeric
dutyPaid	*Null unless contractType = X* States whether stock offered on X contract has the tax status of duty paid. Type: Boolean 'true' or 'false'
minimumQty	*Null unless contractType = X* States whether the seller has placed a minimum volume of units on the trade. Type: integer
deliveryPeriod	*Null unless contractType = X* States the lead time on the offer in weeks. Standard Liv-ex terms (SIB) are 2 weeks. If deliveryPeriod = 0, the offer is Special Now. Type: integer
condition	*Null unless contractType = X* Text field that states any issues with the stock or its packaging. Type: string
photoGUID	Specials only The photo resource ID. Type: integer
lowResolutionPhotoUrl	*Both sell and buy for specials only* HTTP link to a low-resolution version of the photo resource. Type: alphanumeric
quantity	Number of units traded. Type: integer
currency	Either EUR, GBP or USD. The prices that follow will all be expressed in this currency. Type: alphanumeric
bottlePrice	Type: double
unitPrice	The price amount of one unit in the trade. Type: double
totalUnitPrice	The price amount for the total traded goods in the trade, this could be made up of either a single or multiple units. totalUnitPrice = quantity x unitPrice Type: double
commissionRate	The percentage rate which commission is charged per trade. Type: double
commissionAmount	The amount of commission due on the trade. Type: double
settlementFee	The amount due for the settlement process of the trade. Type: double
totalNetAmount	The amount a seller can expect, and a buyer owes for the trade. Not including VAT. When role = buy: totalNetAmount = totalUnitPrice + settlementFee + comissionAmount

	When role = sell: totalNetAmount = totalUnitPrice - settlementFee - comissionAmount Type: double
paid	States whether the trade has been paid or not. Type: Boolean true/false
prepaid	States whether the trade has been paid before an invoice has been issued for the trade. Type: Boolean true/false
invoiceNo	The number of the invoice for the buyer of the trade only. *null if not invoiced or where role = sell* Type: integer

JSON Response

The response is sent per request.


```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1610978675455,
        "provider": "Liv-ex"
    },
    "yourTradeDetails": {
        "lxTradeNo": 232224,
        "tradeDate": 1591040328000,
        "tradeStatus": "traded",
        "orderGUID": "e59e46c3-c95607ce1f9b",
        "lwin": "10131792013060075O",
        "lwinName": "Chateau La Mission Haut-Brion Cru Classe, Pessac-Leognan",
        "lwinRegion": "Bordeaux",
        "vintage": "2013",
        "packSize": "06",
        "bottleSize": "00750",
        "role": "Buy",
        "userName": "API User",
        "merchantRef": "BID-11796",
        "contractType": "SIB",
        "special": {
            "dutyPaid": null,
            "minimumQty": null,
            "deliveryPeriod": null,
            "condition": null,
            "photos": null
        },
        "quantity": 1,
        "currency": "GBP",
        "bottlePrice": 103.0,
        "unitPrice": 618.0,
        "totalUnitPrice": 618.0,
        "commissionRate": "2%",
        "commissionAmount": 12.36,
        "settlementFee": 4.0,
        "totalNetAmount": 634.36,
        "paid": true,
        "prepaid": false,
        "invoiceNo": 312726
```

```
    },
    "errors": null
}
```


Invalid JSON Response


```
{
    "status": "Bad Request",
    "httpCode": "400",
    "message": "Request was unsuccessful",
    "internalErrorCode": "R000",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1607612946295,
        "provider": "Liv-ex"
    },
    "yourTradeDetails": null,
    "errors": {
        "error": [
            {
                "code": "V100",
                "message": "Liv-Ex trade number: [24792] is not available or does not
exist."
            }
        ]
    }
}
```


XML Response

The response is sent as per request.


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<yourTradeDetails>
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2021-01-18T14:10:10.732Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <yourTradeDetails>
        <lxTradeNo>232318</lxTradeNo>
        <tradeDate>2020-03-17T10:10:17Z</tradeDate>
        <tradeStatus>traded</tradeStatus>
        <orderGUID>fcec5fc-46e7-a7fe-780ed958b27a</orderGUID>
        <lwin>1004012200612007 50</lwin>
        <lwinName>Mount Mary, Quintet, Yarra Valley</lwinName>
        <lwinRegion>Victoria</lwinRegion>
        <vintage>2006</vintage>
        <packSize>12</packSize>
        <bottleSize>00750</bottleSize>
        <role>Sell</role>
        <userName>User</userName>
        <merchantRef>LXXX1</merchantRef>
        <contractType>X</contractType>
        <special>
            <dutyPaid>true</dutyPaid>
            <minimumQty>1</minimumQty>
            <deliveryPeriod>0</deliveryPeriod>
```

```
            <condition>As per photos</condition>
            <photos>
                <photoGUID xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
<lowResolutionPhotoUrl>https://dcoygdx2udjif.cloudfront.net/mod/1689444.JPG?Expires=lhREuSY7
JkN3t9rE2UHtE6mZ5RZDp2mTpzQ__&amp;Key-Pair-Id=APKAIDBHABKBT2H3R7XA</lowResolutionPhotoUrl>
            </photos>
        </special>
        <quantity>1</quantity>
        <currency>GBP</currency>
        <bottlePrice>79.5</bottlePrice>
        <unitPrice>954.0</unitPrice>
        <totalUnitPrice>954.0</totalUnitPrice>
        <commissionRate>2.00%</commissionRate>
        <commissionAmount>19.08</commissionAmount>
        <settlementFee>4.0</settlementFee>
        <totalNetAmount>930.92</totalNetAmount>
        <paid>true</paid>
        <prepaid xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
        <invoiceNo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
    </yourTradeDetails>
</yourTradeDetails>
```


Invalid XML Response


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<yourTradeDetails>
    <Status>Bad Request</Status>
    <HttpCode>400</HttpCode>
    <Message>Request was unsuccessful</Message>
    <InternalErrorCode>R000</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2020-12-10T15:45:23.459Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <errors>
        <error>
            <code>V100</code>
            <message>Liv-Ex trade number: [24793] is not available or does not
exist.</message>
        </error>
    </errors>
</yourTradeDetails>
```

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

5.1 Request validation error codes

Code	Message
V000	Mandatory field missing
V100	Liv-ex trade number: [%s] is not available or does not exist

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)
207 Multiple statuses	The message body contains several separate responses of differing statuses
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.