LIV|EX

THE FINE WINE MARKET

# Market Depth API v1

Document Revision 1.1
Date of Issue: 5 April 2019
Date of revision: 22 July 2019

Nick Palmer

Product Manager

## Table of Contents

# 1. Purpose

To provide the API end point information and examples of the web services available for Market Depth.

# 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). |
| **SIB** | Standard in Bond trade terms: https://www.liv-ex.com/knowledge/liv-ex-trading-contracts/ |
| **SEP** | Standard En Primeur: https://www.liv-ex.com/knowledge/liv-ex-trading-contracts/ |
| **X (Special)** | Special contract trade terms: https://www.liv-ex.com/knowledge/liv-ex-trading-contracts/ |
| **Contract Type** | Contract type is a generic term for SIB, SEP or X (Special). |
| **CI** | Condition Issue |

# 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

# 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.<br>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.<br>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>
```

# 5. API Listing

## 5.1 Market Depth service (POST method)

### Description

This service will retrieve bid, offer, trade and list price information for a given LWIN code.

### Base URI

data/v1/marketDepth

### Request Parameters

| Name | Mandatory | Description |
|---|---|---|
| lwin | Y | LWIN11/16/18. Only one LWIN permitted per request<br>Type: integer |
| priceType | Y | Multiple values are permitted<br>Values: 'all', 'bid', 'offer', 'list', 'trade'<br>Type: alphanumeric |
| timeframe | N<br>Default = 14 | Determines the number of days to return data for. Applies to priceType 'list' or 'trade' only. Value 'last5' returns last 5 event occurrences irrespective of time. Only one value permitted per request<br>Values: 'last5', '14', '35', '90'<br>Type: alphanumeric |
| format | N<br>Default = all | Only one value permitted per request<br>Values: 'all', 'cases', 'bottles', 'halves', 'mags', 'large'<br>Type: alphanumeric |
| currency | Y | Only one value permitted per request<br>Values: 'gbp', 'eur', 'chf', 'usd', 'hkd', 'jpy', 'sgd', 'gbp/btt', 'eur/btt', 'chf/btt'<br>Type: alphanumeric |

### Sample Request Body

### JSON Request

```
{
        "lwin": 10118722008,
        "priceType": ["list", "offer"],
        "timeframe": "35",
        "format": "all",
        "currency": "GBP"
}
```

### XML Request

```
<marketDepthRequest>
```

```
        <lwin>10118722008</lwin>
        <priceType>list</priceType>
        <priceType>offer</priceType>
        <timeframe>35</timeframe>
        <format>all</format>
        <currency>GBP</currency>
</marketDepthRequest>
```

### Sample Response Body

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

### Response parameters

| Name | Description |
|---|---|
| lwin | Type: integer<br>Example: 123456720080600750 |
| wineName | Type: alphanumeric<br>Example: Figeac |
| vintage | Type: 4-digit integer<br>Example: 2014, 1000 (for NV) |
| currency | Type: alphanumeric<br>Example: GBP |
| priceType | Type: alphanumeric<br>Example: trade |
| priceDate | Type: datetime. Epoch (JSON), yyyy-mm-dd (XML) |
| packSize | Type: 2-digit integer<br>Example: 6 |
| bottleSize | Type: 5-digit integer<br>Example: 00750 |
| price | Type: double (2dp)<br>Example: 251.35 |
| quantity | Type: integer<br>Example: 1 |
| myPosition | Type: Boolean true/false |
| orderGUID | Type: 128-bit hexadecimal<br>Example: |
| contractType | Type: alphanumeric<br>Example: SIB, SEP, X (for special) |
| ci | Type: alphanumeric<br>Example |
| isSold | Type: boolean true/false |

## JSON Response

The response is sent per request.

```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully.",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1554806904368,
        "provider": "Liv-ex"
    },
    "marketDepth": [
        {
            "lwin": "10118722008",
            "wineName": "Lafite Rothschild",
            "vintage": 2008,
            "currency": "GBP",
            "priceType": "list",
            "priceDate": 1551830400000,
            "packSize": "01",
            "bottleSize": "00375",
            "price": 330,
            "quantity": 2,
            "myPosition": false,
            "orderGUID": null,
            "contractType": null,
            "ci": "us",
            "isSold": false
        },
        {
            "lwin": "10118722008",
            "wineName": "Lafite Rothschild",
            "vintage": 2008,
            "currency": "GBP",
            "priceType": "offer",
            "priceDate": 1549966870000,
            "packSize": "01",
            "bottleSize": "00750",
            "price": 550,
            "quantity": 1,
            "myPosition": false,
            "orderGUID": "24937d65-be8a-4a53-914a-2d6b170a7222",
            "contractType": "X",
            "ci": null,
            "isSold": null
        },
        {
            "lwin": "10118722008",
            "wineName": "Lafite Rothschild",
            "vintage": 2008,
            "currency": "GBP",
            "priceType": "list",
            "priceDate": 1554246000000,
            "packSize": "01",
            "bottleSize": "00750",
            "price": 592,
            "quantity": 5,
            "myPosition": false,
            "orderGUID": null,
            "contractType": null,
            "ci": "-",
            "isSold": false
        }
    ],
    "errors": null
}
```

**Invalid JSON response**


```
{
    "status": "Bad Request",
    "httpCode": "400",
    "message": "Request was unsuccessful.",
    "internalErrorCode": "R000",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1554803821692,
        "provider": "Liv-ex"
    },
    "marketDepth": null,
    "errors": {
        "error": [
            {
                "code": "V116",
                "message": "Invalid / incorrect format: [ll]. Possible values are 'all',
'cases', 'bottles', 'halves', 'mags', 'large'."
            },
            {
                "code": "V120",
                "message": "Invalid / incorrect timeframe: [45]. Possible values are
'last5', '14', '35', '90'."
            }
        ]
    }
}
```


## XML Response

The response is sent per request.


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<marketDepthResponses xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://tpg-qa-api.liv-ex.com/v1 https://tpg-qa-api.liv-
ex.com/schema/v1/services.xsd">
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully.</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2019-04-09T11:49:36.664+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <marketDepth>
        <lwin>10118722008</lwin>
        <wineName>Lafite Rothschild</wineName>
        <vintage>2008</vintage>
        <currency>GBP</currency>
        <priceType>list</priceType>
        <priceDate>2019-03-06T00:00:00Z</priceDate>
        <packSize>01</packSize>
        <bottleSize>00375</bottleSize>
        <price>330.0</price>
        <quantity>2</quantity>
        <myPosition>false</myPosition>
        <orderGUID xsi:nil="true"/>
        <ci>us</ci>
        <isSold>false</isSold>
    </marketDepth>
    <marketDepth>
        <lwin>10118722008</lwin>
        <wineName>Lafite Rothschild</wineName>
        <vintage>2008</vintage>
```

```
        <currency>GBP</currency>
        <priceType>offer</priceType>
        <priceDate>2019-02-12T10:21:10Z</priceDate>
        <packSize>01</packSize>
        <bottleSize>00750</bottleSize>
        <price>550.0</price>
        <quantity>1</quantity>
        <myPosition>false</myPosition>
        <orderGUID>24937d65-be8a-4a53-914a-2d6b170a7222</orderGUID>
        <contractType>X</contractType>
        <ci xsi:nil="true"/>
        <isSold xsi:nil="true"/>
    </marketDepth>
    <marketDepth>
        <lwin>10118722008</lwin>
        <wineName>Lafite Rothschild</wineName>
        <vintage>2008</vintage>
        <currency>GBP</currency>
        <priceType>list</priceType>
        <priceDate>2019-04-03T00:00:00+01:00</priceDate>
        <packSize>01</packSize>
        <bottleSize>00750</bottleSize>
        <price>592.0</price>
        <quantity>5</quantity>
        <myPosition>false</myPosition>
        <orderGUID xsi:nil="true"/>
        <ci>-</ci>
        <isSold>false</isSold>
    </marketDepth>
</marketDepthResponses>
```


**Invalid XML Response**


```
?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<marketDepthResponses xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://tpg-qa-api.liv-ex.com/v1 https://tpg-qa-api.liv-
ex.com/schema/v1/services.xsd">
    <Status>Bad Request</Status>
    <HttpCode>400</HttpCode>
    <Message>Request was unsuccessful.</Message>
    <InternalErrorCode>R000</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2019-04-09T10:57:52.192+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <errors>
        <error>
            <code>V116</code>
            <message>Invalid / incorrect format: [ll]. Possible values are 'all',
'cases', 'bottles', 'halves', 'mags', 'large'.</message>
        </error>
        <error>
            <code>V120</code>
            <message>Invalid / incorrect timeframe: [45]. Possible values are 'last5',
'14', '35', '90'.</message>
        </error>
    </errors>
</marketDepthResponses>
```

# 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

| Code | Message |
|---|---|
| **V058** | Invalid / incorrect LWIN: [lwin_code]. Please provide a valid LWIN11, LWIN16 or LWIN18 code. |
| **V061** | Invalid / incorrect currrency: [GaBP]. Possible values are 'gbp', 'eur', 'chf', 'usd', 'hkd', 'jpy', 'sgd', 'gbp/btt', 'eur/btt', 'chf/btt'. |
| **V116** | Invalid / incorrect format: [a]. Possible values are 'all', 'cases', 'bottles', 'halves', 'mags', 'large'. |
| **V117** | Timeframe last5 is only available for priceType 'list' or 'trade'. |
| **V118** | Only one priceType value is permitted per request for when timeframe = last5. |
| **V119** | Invalid / incorrect priceType: [[j]]. Possible values are 'bid', 'offer', 'list', 'trade' or 'all'. |
| **V120** | Invalid / incorrect timeframe: [10]. Possible values are 'last5', '14', '35', '90'. |

## 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) |

| | |
|---|---|
| 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. |