LIV|EX

THE FINE WINE MARKET

Bulk Order Action API v3

Document Revision 1.0
Date of Issue: 08 August 2018
Date of revision: 08 August 2018

Nick Palmer

Product Manager

Table of Contents

1. Purpose

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

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.
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/
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 Special (X).

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. 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": 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>2017-11-04T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
<Response>
```

5. API Listing

5.1 Bulk Order Action service (POST method)

Description

This service can be used to make bulk changes to order positions on the Li-ex exchange platform. The service offers 3 actions:

- Order suspension
- Order reactivation
- Order renewal

The order(s) to be manipulated should be identified by their associated orderGUID. There is no limit to the number of orderGUIDs that can be submitted per request, but only one action can be specified per payload.

Base URI

exchange/v3/bulkOrderAction

Request Parameters

Name	Mandatory	Description
orderGUID	Y	The Liv-ex order identifier. Enables querying of a specific order. Attribute is ignored if lwin value is provided. Multiple values are permitted. Type: 128-bit alphanumeric
bulkAction	Y	The action to apply to the orderGUID(s) specified. Type: alphanumeric Values: 'bulkSuspend', 'bulkReactivate', 'bulkRenew'

Sample Request Body

JSON Request

```
{
    "orderGUID": ["0afbc261-56c5-4641-b6c1-bdebf99fcd87","1464cf08-4328-4425-84fe-
d3530a2cbc89"],
    "bulkAction": "bulkSuspend"
}
```

XML Request

```
<BulkOrderAction>
    <orderGUID>0afbc261-56c5-4641-b6c1-bdebf99fcd87</orderGUID>
    <orderGUID>1464cf08-4328-4425-84fe-d3530a2cbc89</orderGUID>
    <bulkAction>bulkSuspend</bulkAction>
</BulkOrderAction>
```

Sample Response Body

The Bulk Order Action service will respond with HTTP Code 200 OK in a successful response. Where multiple orderGUIDs have been submitted but some have failed the service will return an HTTP 207 (Multiple statuses) specifying the orderGUIDs that have failed. See section 6 for a full list of API response and error codes.

Response parameters

Name	Description
orderGuid	The GUID of the bid/offer position. For priceType = 'list' this attribute will show as null Type: 128-bit hexadecimal

JSON Response

The response is sent per request.

Success JSON response (200)

```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully.",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "3.0",
        "timestamp": 1532951533994,
        "provider": "Liv-ex"
    },
    "orders": [],
    "errors": null
}
```

Multi-status JSON response (207)

```
{
    "status": "Multiple statuses",
    "httpCode": "207",
    "message": "Request partially completed",
    "internalErrorCode": "R002",
    "apiInfo": {
        "version": "3.0",
        "timestamp": 1532950423476,
        "provider": "Liv-ex"
    },
    "orders": [
        {
            "orderGUID": "ab76f8bf-7d45-4716-8ff6-f787a9e26cef",
            "error": {
                "code": "TR003",
                "message": "bid.fat.finger.weak.warn"
            }
        },
        {
            "orderGUID": "10d4cb1a-baf5-4c94-8e21-9301ba4bf760",
            "error": {
                "code": "TR005",
                "message": "offer.fat.finger.weak.warn"
            }
        }
    ],
    "errors": null
}
```

Invalid JSON response

```
{
    "status": "Unauthorized",
    "httpCode": "401",
    "message": "Unauthorized",
    "internalErrorCode": null,
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1532951953090,
        "provider": "Liv-ex"
    },
    "orders": null
}
```

XML Response

The response is sent per request.

Success XML response (200)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<orders 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>3.0</Version>
        <Timestamp>2018-07-30T13:53:19.309+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
</orders>
```

Multi-status XML response (207)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<orders 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>Multiple statuses</Status>
    <HttpCode>207</HttpCode>
    <Message>Request partially completed</Message>
    <InternalErrorCode>R002</InternalErrorCode>
    <ApiInfo>
        <Version>3.0</Version>
        <Timestamp>2018-07-30T13:58:37.397+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <orders>
        <orderGUID>3a003355-18c3-4b39-9b17-3d92c5d08185</orderGUID>
        <error>
            <code>TR005</code>
            <message>offer.fat.finger.weak.warn</message>
        </error>
    </orders>
</orders>
```

Invalid XML Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<exchangeResponse 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>Unauthorized</Status>
    <HttpCode>401</HttpCode>
    <Message>Unauthorized</Message>
    <InternalErrorCode xsi:nil="true"/>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2018-07-30T13:49:46.179+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <Orders xsi:nil="true"/>
</exchangeResponse>
```

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
V000	Mandatory field missing
V001	Merchant is not allowed to access the requested feed
V002	Invalid parameter(s)
V004	Invalid number parameter: positive number expected for <paramName>
V005	Merchant is not active
V012	Invalid request headers. Please provide value for header <header name>
V018	Mandatory field missing (<fieldName>).
V056	GUID is not available or does not exist
V058	Invalid / incorrect LWIN: [<value>]. Please provide a valid LWIN11, LWIN16 or LWIN18 code.
V059	Invalid / incorrect contractType: [<value>]. Possible values are 'sib', 'sep', 'x' or 'all'.
V061	invalid / incorrect currrency: [<value>]. Possible values are 'gbp', 'eur', 'chf', 'usd', 'hkd', 'jpy', 'sgd', 'gbp/btt', 'eur/btt', 'chf/btt'.

V066	Invalid / incorrect orderStatus: [<value>]. Possible values are ‘live’, ‘suspended’, ‘all’.
V067	Invalid / incorrect onlyExpiring: [<value>]. Possible values are ‘true’, ‘false’.
V068	Invalid / incorrect orderType: [<value>]. Possible values are ‘bid’, ‘offer’, ‘all’.

6.2 Trade validation error codes

Code	Error Key	Meaning
TR001	ep.offer.restrict.due.to. exceeding.qty.than.stored	Merchant En Primeur selling privileges have been restricted.
TR002	invalid.min.unit.and.qty	Your bid does not meet the minimum quantity terms of the contract
TR003	bid.fat.finger.weak.warn	A bid appears to be above Market Price. Please check carefully before proceeding.
TR004	bid.fat.finger.strong.warn	A bid appears to be above Market Price. Please check carefully before proceeding.
TR005	offer.fat.finger.weak.warn	An offer appears to be below Market Price. Please check carefully before proceeding.
TR006	offer.fat.finger.strong.warn	An offer appears to be below Market Price. Please check carefully before proceeding.
TR007	order.did.not.confirm. successfully.try.again	The order did not confirm successfully. Please check request syntax and try again.
TR010	phy.offer.restrict.due.to. exceeding.qty.than.stored	Merchant trading status is set to 'buy and resell only'.
TR011	merchant.about.to.match.his. offer	Merchant is about to match their own offer
TR012	merchant.about.to.match.his. bid	Merchant is about to match their own bid
TR014	merchant.status.no.trading	Merchant does not have trading privileges.
TR015	merchant.status.sell.only.no. phy.bid	Merchant account is ‘sell only'.
TR016	merchant.status.sell.only.ep. resell.for.a.vintage	Merchant is not permitted to buy EP stock. EP sell status is ‘resell only’.
TR017	merchant.status.sell.only.no. ep.for.a.vintage	Merchant is not permitted to sell EP stock.
TR018	merchant.status.sell.only.ep. allowed	Merchant is not permitted to sell EP stock.

TR019	merchant.status.sso.no.ep.for .a.vintage	Merchant is not allowed to buy and sell EP stock as the trading status is 'sell special only.
TR020	sso.offer.restrict.due.to.exceed ing.qty.than.stored	You are currently only allowed to create special offers. Please change your offer

6.3 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	Conveys information about multiple resources in situations where multiple status codes have been returned.
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.