LIV | EX

THE FINE WINE MARKET

Direct Market Access documentation

Document revision 2.2
Date of Issue: 22 November 2017
Date of revision: 19 March 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 Exchange Integration.

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: 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/
Special Now	An offer of stock that is ready for immediate dispatch from Liv-ex warehouses.
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 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
- Add and delete services are one order at a time by default, but multiple orders and deletions are possible.
- 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 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 Exchange Integration API will be accessible at https://api.liv-ex.com/exchange

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.

Param

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

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": "Request was unsuccessful",
      "livexCode": "R000"
      "apiInfo": {
            "version": "2.0",
            "timestamp": "2017-11-04T11:12:30",
            "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>2017-11-04T11:12:30</Timestamp>
           <Provider>Liv-ex</Provider>
      </ApiInfo>
<Response>
```

5. API Listing

5.1 Heartbeat (HEAD or GET method)

Description

This may be used by merchant systems to ping/poll the Exchange Integration service for availability, and if its available then further requests for Orders may be placed.

Base URI

exchange/heartbeat

Response

Exchange Integration service will respond with HTTP Code 200 - OK in the response, depending on whether a HEAD or GET request was made.
HTTP HEAD - https://api.liv-ex.com/exchange/heartbeat

```
OK 200
```

HTTP GET - https://api.liv-ex.com/exchange/heartbeat

```
{
   "status": "OK",
   "httpCode": "200",
   "message": "available",
   "internalErrorCode": null,
   "apiInfo": {
      "version": "1.0",
      "timestamp": 1511190157556,
      "provider": "Liv-ex"
   },
   "orders": null
}
```

If the Exchange Integration service is down (because of maintenance or any network issues) then the response will return HTTP Error Code **500 - Internal Server Error.**

Note: Liv-ex will automatically suspend integrated merchants' SIB orders with 'suspended' status on the restart of the Exchange Integration service (if the Exchange Integration service has been down because of maintenance or any network issues). This should be followed by delete + add requests by the merchant system to be able sync and reinstate their respective positions.

Throttling

Every heartbeat invocation registers as a valid API request, and will be used against the allowable throttling limit i.e. API calls made on hourly basis. Once threshold is reached no more calls will be allowed till next allowable time period.

5.2 Add Order Service (POST method)

Description

This service will be used to add a bid or offer.

Base URI

exchange/v2/orders

Param

Name	Mandatory	Description
contractType	Y	A valid contract type of the order. The possible value can be **sib** (Standard In Bond), **sep** (Standard En Primeur) and **x** (Special). **Type:** alphanumeric
orderType	Y	A valid type of the order. The possible values can be **B** (bid) and **O** (offer). **Type:** alphanumeric **Note:** For **contractType = X** (Special) only **orderType = B** is permitted.
orderGUID	N Mandatory for contractType X	An order idenfication code that the Special bid should be placed against. **Type:** 128-bit hexadecimal

orderStatus	Y	A valid order status. The possible values can be L (live) and S (suspended). See recommendation note. **Type:** alphanumeric
expiryDate	N	A valid future date in yyyy-mm-dd format e.g. 2015-07-31. **Type:** alphanumeric, ISO8601 format
lwin	Y	A valid LWIN7 or LWIN18 code **Type:** integer, 7-digits (LWIN7), 18-digits (LWIN18)
vintage	N	Mandatory if LWIN7 is provided. The value can be one year less than current year. For non-vintage wines use 1000. **Type:** 4-digit integer
bottleInCase	N	Mandatory if LWIN7 is provided. Values are typically 1, 3, 6, 12, 24 **Type:** 2-digit integer
bottleSize	N	Mandatory if LWIN7 is provided. The values must be in ml (millilitres) expressed as a 5-digit term e.g. 75cl = 00750. **Type:** 5-digit integer
currency	Y	Either EUR or GBP. The currency used must match the trading currency pre-agreed with Liv-ex. **Type:** 3-character alphanumeric
price	Y	A valid positive value. GBP prices will be rounded to the nearest whole integer. EUR prices will be round to 1 decimal place. **Type:** integer or double
quantity	Y	A valid positive integer value of quantity of packs such as 1,2, 50 etc. **Type:** integer
merchantRef	N	An optional text field used to attach a reference to the order. Limited to 30 characters. Strings exceeding this limit will be truncated. **Type:** alphanumeric (30-character limit)

Recommendation Notes:

To unsuspend a suspended position through the Exchange Integration service, please send a delete for the suspended order first then add a new order.

Sample Request Body

By default, we will have request as multiple array/list of Orders even though the Client may pass only one as given in the example below.

JSON Request (single order)

```
{"orders":[
     {
          "contractType": "sib",
          "orderType": "B",
          "orderGUID": "",
          "orderStatus": "L",
          "expiryDate": "2015-07-31",
          "lwin": "1122763",
          "vintage": "2007",
          "bottleInCase": "12",
          "bottleSize": "00750",
          "currency": "EUR",
          "price": "416",
          "quantity": "1",
          "merchantRef": "MyRef"
     }]
}
```

JSON Request (multiple orders)

```
{"orders":[
     {
          "contractType": "sib",
          "orderType": "o",
          "orderGUID": "",
          "orderStatus": "L",
          "expiryDate": "2017-03-06",
          "lwin": "1102200",
          "vintage": "2013",
          "bottleInCase": "6",
          "bottleSize": "00750",
          "currency": "GBP",
          "price": "150",
          "quantity": "1",
          "merchantRef": "MyRef"
     },
     {
         "contractType": "x",
         "orderType": "b",
          "orderGUID": "a7f51a59-63a3-4a87-8c1b-3aed0269dd72",
         "orderStatus": "L",
         "expiryDate": "2017-03-06",
         "lwin": "1012316",
         "vintage": "1990",
          "bottleInCase": "6",
         "bottleSize": "00750",
         "currency": "GBP",
         "price": "875",
         "quantity": "1",
         "merchantRef": "MyRef"
     }
]}
```

XML Request (single order)

```
<Orders>
    <Order>
        <contractType>sib</contractType>
        <orderType>o</orderType>
        <orderGUID></orderGUID>
        <orderStatus>L</orderStatus>
        <expiryDate>2017-02-05</expiryDate>
        <lwin>1012316</lwin>
        <vintage>1990</vintage>
        <bottleInCase>6</bottleInCase>
        <bottleSize>00750</bottleSize>
        <currency>GBP</currency>
        <price>5250</price>
        <quantity>1</quantity>
        <merchantRef>MyRef</merchantRef>
    </Order>
</Orders>
```

XML Request (multiple orders)

```
<Orders>
    <Order>
        <contractType>sib</contractType>
        <orderType>o</orderType>
        <orderGUID></orderGUID>
        <orderStatus>L</orderStatus>
        <expiryDate>2017-03-05</expiryDate>
        <lwin>1012316</lwin>
        <vintage>1990</vintage>
        <bottleInCase>6</bottleInCase>
        <bottleSize>00750</bottleSize>
        <currency>GBP</currency>
        <price>5250</price>
        <quantity>1</quantity>
        <merchantRef>MyRef</merchantRef>
    </Order>
    <Order>
        <contractType>x</contractType>
        <orderType>b</orderType>
        <orderGUID>a7f51a59-63a3-4a87-8c1b-3aed0269dd72</orderGUID>
        <orderStatus>L</orderStatus>
        <expiryDate>2017-03-05</expiryDate>
        <lwin>1012387</lwin>
        <vintage>2011</vintage>
        <bottleInCase>12</bottleInCase>
        <bottleSize>00750</bottleSize>
        <currency>GBP</currency>
        <price>525</price>
        <quantity>1</quantity>
        <merchantRef>MyRef</merchantRef>
    </Order>
</Orders>
```

Sample Response Body

Name	Mandatory	Description

merchantRef	Y	The optional text field that was included as part of the initial request.
orderGUID	Y	The GUID of the new order that has been created. The GUID is required when sending a DELETE request.
orderPlaceDate	Y	The timestamp of when the new order was placed.

JSON Response

The response is sent per order

```
{
	"status": "OK",
	"httpCode": "200",
	"message": "Request completed successfully",
	"internalErrorCode": "R001",
	"apiInfo": {
		"version": "2.0",
		"timestamp": "1521454694828",
		"provider": "Liv-ex"
	},
	"orders": [{
		"merchantRef": "MyRef",
		"orderGUID": "94B5CC70-BC3D-49C3-B636-C3C7552E543D",
		"orderPlaceDate": "1521454694537"
		"errors": "",
	}]
}
```

Invalid JSON response

```
{
	"status": "failure",
	"httpCode": "400",
	"internalErrorCode": "R002",
	"message": "Request partially completed",
	"apiInfo": {
	    "version": "2.0",
	    "timestamp": "1521454694828",
	    "provider": "Liv-ex"
	},
	"orders": [{
		"merchantRef": "MyRef",
		"orderGUID": "",
		"orderPlaceDate": ""
		"errors": {
			"error": [{
				"code": "TR005",
				"message": "Please provide positive numeric value of qty."
				},
				{
				"code": "TR006",
				"message": "Please provide positive decimal value of price."
			}]
		},
	}]
}
```

XML Response

The response is sent per order

```
<exchangeResponse>
    <Status>OK</Status>
    <HttpCode>200</Code>
    <Message>Request completed successfully.</Message>
    <internalErrorCode>R001</internalErrorCode>
    <Orders>
        <Order>
            <merchantRef>MyRef</MerchantRef>
            <orderGUID>94B5CC70-BC3D-49C3-B636-C3C7552E543D</OrderId>
            <orderPlaceDate>2015-06-04T07:22:25</OrderPlaceDate>
            <Errors/>
        </Order>
    </Orders>
</exchangeResponse>
```

Invalid XML Response

```
<exchangeResponse>
    <Status>failure</Status>
    <HttpCode>400</Code>
    <Message>Sorry, problem encountered while processing the Order.</Message>
    <internalErrorCode>R001</internalErrorCode>
    <ApiInfo>
        <Version>2.0</Version>
        <Timestamp>2015-06-04T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <Orders>
        <Order>
            <merchantRef>MyRef</MerchantRef>
            <orderGUID/>
            <orderPlaceDate/>
            <Errors>
                <Error>
                    <Code>TR005</Code>
                    <Message>Please provide positive numeric value of qty.</Message>
                </Error>
                <Error>
                    <Code>TR006</Code>
                    <Message>Please provide positive decimal value of price</Message>
                </Error>
            </Errors>
        </Order>
    </Orders>
</exchangeResponse>
```

5.3 Delete Order Service (DELETE method)

Description

This web service will be used to delete the bid or offer of a merchant.

Base URI

exchange/v2/orders

Parameters

Name	Mandatory	Description

orderGUID	Y	A valid order GUID for the account.

Sample JSON DELETE request body

Order ID is a GUID number returned when using add order service

JSON
```
{
    "orderGUID": ["94B5CC70-BC3D-49C3-B636-C3C7552E543D"]
}
```

XML
```
<orderDeleteRequest>
    <orderGUID>94B5CC70-BC3D-49C3-B636-C3C7552E543D </orderGUID>
</orderDeleteRequest>
```

JSON Response

Response with valid order ID

```
{
    "status":"OK",
    "httpCode":"200",
    "message":"Request completed successfully.",
    "internalErrorCode":"R001",
    "apiInfo": {
        "version":"2.0",
        "timestamp":1462450114401,
        "provider":"Liv-ex"
    },
    "orders":{
        "order": [{
        "merchantRef":"MyRef",
        "orderGUID":"94B5CC70-BC3D-49C3-B636-C3C7552E543D",
        "orderPlaceDate":1462450114337,
        "errors”: null
        }]
    }
}
```

Response with invalid order ID

```
{
    "status": "failure",
    "httpCode": "400",
    "message": "Sorry, problem encountered while processing the Order.",
    "internalErrorCode": "R000",
    "apiInfo": {
        "version": "2.0",
        "timestamp": "1462450114401",
        "provider": "Liv-ex"
        },
    "orders": [{
    "merchantRef": "MyRef",
    "orderGUID": "94B5CC70-BC3D-49C3-B636-C3C7552E543D",
    "response": ""
    "errors": [{
        "error": {
            "code": "TR001",
            "message": "Merchant and order combination does not match."
            }
        }],
    }]
}
```

XML Response

Response with valid order ID

```
<?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>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully.</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>2.0</Version>
        <Timestamp>2017-11-21T12:26:34.917Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <Orders>
        <order>
            <MerchantRef>MyRef</MerchantRef>
            <OrderGUID>c1c9a11f-5c8f-4746-99d9-06bb4d2216b6</OrderGUID>
            <OrderPlaceDate>2017-11-21T12:26:34.838Z</OrderPlaceDate>
            <Errors xsi:nil="true"/>
        </order>
    </Orders>
</exchangeResponse>
```

Response with invalid order ID

```
<Response>
    <Status>failure</Status>
    <HttpCode>400</Code>
    <internalErrorCode>R000</internalErrorCode>
    <Message>Request unsuccessful</Message>
    <ApiInfo>
        <Version>2.0</Version>
        <Timestamp>2017-11-04T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <Orders>
        <Order>
            <merchantRef>MyRef</MerchantRef>
            <orderGUID>94B5CC70-BC3D-49C3-B636-C3C7552E543D</OrderId>
            <Errors>
                <Error>
                <Code>TR001</Code>
                <Message>Merchant and order combination does not match.</Message>
                </Error>
            </Errors>
        </Order>
    </Orders>
</Response>
```

5.4 PUSH notifications

Description

A PUSH message will be invoked when a trade matches. The information will be sent to a user's system as a PUSH request (XML/JSON payload) to the merchant's URL.

Merchant URL

<merchant_url>

A URL that each merchant should provide to Liv-ex to allow information to be pushed back to their system. The URL should be backed by a HTTP-based POST service capable of interpreting an incoming XML/JSON payoad.

Push service requirements - merchant side

The PUSH service is comprised of 2 parts

1. A HEAD request (to check the merchant system is alive).
2. The PUSH payload

The service will always send a HEAD request to ping the Merchant URL before sending the PUSH notification. If the Merchant URL fails to respond to the HEAD with HTTP Code 200 OK, the PUSH notification will not be sent.

Merchant systems must respond to the PUSH request with a 200 OK to confirm receipt of the payload.

Important – Order suspend behaviour

PUSHs are invoked whenever an order belonging to the integrated user is added, amended, suspended or deleted. If Liv-ex cannot validate that the user's system is alive ALL ORDERS WILL BE SWITCHED TO A SUSPENDED STATE.

If the Merchant URL is inaccessible (no 200 OK is received for the HEAD), Liv-ex will retry up to 4 times per a retry schedule. If the retries fail, all live orders will be suspended.

To sync and reactivate, merchant systems must DELETE and ADD their respective positions.

5.4.1 Confirm Trade (PUSH method)

Confirm Trade PUSH example

XML

```
<PushResponse>
      <trade>
            <order_guid>94B5CC70-BC3D-49C3-B636-C3C7552E543D</order_guid>
            <merchant_ref>test ref</merchant_ref>
            <trade_id>123450</trade_id>
            <qty>20</qty>
             <trade_date>2015-06-12T17:00:00</trade_date>
      </trade>
</PushResponse>
```

JSON

```
 {
      "trade": {
            "order_guid": "94B5CC70-BC3D-49C3-B636-C3C7552E543D",
            "merchant_ref": "test ref",
            "trade_id": "123450",
            "qty": "20"
            "trade_date": "2015-06-12T17:00:00"
      }
}
```

Param

Name	Description
order_guid	Bid or Offer Id.
merchant_ref	The merchant reference for the order. This node will only be pushed if there was a reference provided on the order.
trade_id	Liv-ex Trade Id.
qty	The total quantity of the trade.
trade_date	The date and time of trade in ISO 8601 format.

5.4.2 Order Update (PUSH method)

Order Update PUSH example

XML
```
<PushResponse>
    <order>
        <order_guid>94B5CC70-BC3D-49C3-B636-C3C7552E543D</order_guid>
        <merchant_ref>Abc</merchant_ref>
        <push_type>Order Suspended</push_type>
        <contract_type>SIB</contract_type>
        <order_type>Bid</order_type>
        <order_status>Suspended</order_status>
        <expiry_date>2015-06-12T17:00:00</expiry_date>
        <lwin>101430720081200750</lwin>
        <price>240</price>
        <qty>1</qty>
        <order_update_date>2015-06-04T07:22:25</order_update_date>
    </order>
</ PushResponse>
```

JSON
```
{
    "order": {
        "order_guid": "94B5CC70-BC3D-49C3-B636-C3C7552E543D",
        "merchant_ref": "Abc",
        "push_type": "Order Suspended",
        "contract_type": "SIB",
        "order_type": "Bid",
        "order_status": "Suspended",
        "expiry_date": "2015-06-12T17:00:00",
        "lwin": "101430720081200750",
        "price": "240",
        "qty": "1",
        "order_update_date": "2015-06-04T07:22:25"
        }
}
```

Param

Name	**Description**
order_guid	Order ID which was created/edited/deleted.
merchant_ref	Merchant reference for the order if provided. This node will not be pushed if there is no reference provided by merchant.
push_type	Type of push request. The possible values will be Order Created, Order Edited, Order Suspended, Unsuspended, Order Deleted, Order Blocked, Order Unblocked.
contract_type	The contract type of the order. The possible values will be SIB.
order_type	The order type. The possible values will be Bid/Offer.
order_status	The current order status in the system. The possible values will be Live, Suspended, Deleted
expiry_date	The expiry date in ISO 8601 format.

lwin	The LWIN18 of the wine will be provided.
price	The order price provided by the merchant in their trading currency.
qty	The total quantity of the order provided by the merchant.
order_update_date	The order update date time in ISO 8601 format.

5.5 Order Status Service (POST method)

Description

This service may be used by merchant systems to check the status of one or more live postions on the exchange prior to sending a bid or offer.

Base URI

/exchange/v1/orderStatus

Param

Name	Mandatory	Description
orderGUID	Y	An order idenfication code (GUID) **Type:** 128-bit hexadecimal

Sample request body

The service accepts one or more order GUIDs per request. There is a maximum limit of 50 GUIDs per request.

JSON Request

Single GUID

```
{
    “orderGUID”:
    [
        "9a68b502-72cd-4a10-84f8-d1d5979538e3"
    ]
}
```

Multiple GUIDs

```
{
    “orderGUID”:
    [
        "9a68b502-72cd-4a10-84f8-d1d5979538e3",
        "fe971427-e6e8-43ba-9223-e0d49b9c8505",
        "d2bbcd37-2db2-48d6-bb32-ad8c0030edb1",
        "1bf62e70-656e-4f15-87b8-a1944043e1f9"
    ]
}
```

XML Request

Single GUID

```
<orderStatusRequest>
```

```
    <orderGUID>9a68b502-72cd-4a10-84f8-d1d5979538e3</orderGUID>
</orderStatusRequest>
```

Multiple GUIDs

```
<orderStatusRequest>
    <orderGUID>9a68b502-72cd-4a10-84f8-d1d5979538e3</orderGUID>
    <orderGUID> fe971427-e6e8-43ba-9223-e0d49b9c8505</orderGUID>
    <orderGUID> d2bbcd37-2db2-48d6-bb32-ad8c0030edb1</orderGUID>
    <orderGUID>1bf62e70-656e-4f15-87b8-a1944043e1f9</orderGUID>
</orderStatusRequest>
```

JSON response

The response is sent per request

```
{
    "orderStatus": {
        "status": [
        {
            "orderGUID": "9a68b502-72cd-4a10-84f8-d1d5979538e3",
            "contractType": "X",
            "special": {
                "dutyPaid": false,
                "minimumQty": 1,
                "deliveryPeriod": 0,
                "condition": "banded cases"
                },
            "orderType": "O",
            "orderStatus": "L",
            "expiryDate": "2018-01-15",
            "lwin": "1160743",
            "vintage": 2006,
            "bottlesInCase": "03",
            "bottleSize": "00750",
            "quantity": 2,
            "currency": "GBP",
            "price": 1725,
            "myOrder": false,
            "errors": null
            }
        ]
    },
    "error": null,
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully.",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1510649691167,
        "provider": "Liv-ex"
    }
}
```

Invalid JSON response

```
{
    "orderStatus": null,
    "error": {
        "code": "V056",
        "message": "GUID is not available or does not exist"
    },
    "status": "Bad Request",
    "httpCode": "400",
    "message": "Request was unsuccessful.",
    "internalErrorCode": "R000",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1511190885207,
        "provider": "Liv-ex"
    }
}
```

XML response

The response is sent per request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<orderStatusResponse 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>1.0</Version>
        <Timestamp>2017-11-14T10:18:29.451Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <Orders>
        <order>
          <orderGUID>9a68b502-72cd-4a10-84f8-d1d5979538e3</GUID>
          <contractType>X</contractType>
          <special>
              <dutyPaid>false</dutyPaid>
              <minimumQty>1</minimumQty>
              <deliveryPeriod>0</deliveryPeriod>
              <condition>banded cases</condition>
          </special>
          <orderType>O</orderType>
          <orderStatus>L</orderStatus>
          <expiryDate>2018-01-15T00:00:00Z</expiryDate>
          <lwin>1160743</lwin>
          <vintage>2006</vintage>
          <bottleInCase>03</bottleInCase>
          <bottleSize>00750</bottleSize>
          <quantity>2</quantity>
          <currency>GBP</currency>
          <price>1725.0</price>
          <myOrder>false</myOrder>
          <errors xsi:nil="true"/>
        </order>
    </Orders>
</orderStatusResponse>
```

Invalid XML response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<orderStatusResponse 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>Bad Request</Status>
    <HttpCode>400</HttpCode>
    <Message>Request was unsuccessful.</Message>
    <InternalErrorCode>R000</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2017-11-20T15:16:09.418Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <orderStatus xsi:nil="true"/>
    <error>
        <code>V056</code>
        <message>GUID is not available or does not exist</message>
    </error>
</orderStatusResponse>
```

Param

Name	**Description**
orderGUID	The order idenfication code (GUID) that was requestd
contractType	The contract type of the order. The possible value will be **SIB** (Standard In Bond), **SEP** (Standard En Primeur) and **X** (Special).
dutyPaid	**null** unless contractType = X States whether the stock offered on the Special contract has a tax status of duty paid or not. If set to 'false' stock should be considered In Bond (IB). **Type:** Boolean 'true' or 'false'
minimumQty	**null** unless contractType = X States whether the seller has placed a minimum volume of units on the trade. E.g the seller has 50 units on offer with a minimumQty value of 10. **Type:** integer
deliveryPeriod	**null** unless contractType = X States whether the lead time on the offer is different to the standard Liv-ex terms of 2 weeks. If deliveryPeriod = 0, the offer is **Special Now** i.e. the stock is n the Liv-ex warehouse, has been checked and is ready for immediate dispatch. **Type:** integer
condition	**null** unless contractType = X A free text fied that states any issues with the stock or its packaging. **Type:** string
orderType	The order type. The possible values can be **B** (bid) or **O** (offer).

orderStatus	The status of the order. The possible values will be **L** (live) and **S** (suspended). Orders that have been deleted or have traded will return an error.
expiryDate	The expiry date in ISO8601 format.
lwin	The LWIN7 of the wine associated with the order GUID
vintage	The vintage of the wine associated with the order GUID (4-digit integer).
bottleInCase	The case size of the wine associated with the order GUID (2-digit integer).
bottleSize	The bottle size of the wine associated with the order GUID expressed in ml (millilitres) (5-digit integer).
quantity	The listed quantity of units available under the associated order GUID. A valid positive integer value of quantity of packs such as 1, 8, 50 etc.
currency	Either EUR, EUR/btt or GBP. The currency will match the trading currency pre-agreed with Liv-ex. The prices that follow will all be expressed in this currency.
price	The listed price of the associated order on the exchange. GBP prices will be rounded to the nearest whole integer. EUR prices will be round to 2 decimal places.
quantity	A valid positive integer value of quantity of packs such as 1,2, 50 etc.
myOrder	A Boolean field that states if the order GUID queried is owned by the user making the request. Possible value can be **true** or **false**.

6. Exchange validations override

Description

The Liv-ex exchange plafrom validates every order it receives to ensure the correct products are traded. A subset of validations (known internally as 'fat finger') look for price and quantity values differ significantly from normal and block these offers.

- Fat finger weak / strong
  Bid or offers that are significantly higher or lower than the market price are rejected
- Quantity higher than price
  Orders where the quantity value is higher than the sale price per unit are rejected.

It is possible to opt out of these validations. Pleae contact your exchange manger if you would like to know more.

7. 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

7.1 Request validation error codes

Code	Message
V000	Mandatory field missing.
V001	Merchant is not allowed to access the requested feed.
V002	Invalid parameter(s).
V003	Wrong date format. Date should be 'yyyy-MM-dd'.
V004	Invalid number parameter: positive number expected for {paramName}.
V005	Merchant is not active.
V006	Invalid LWIN number.
V007	Invalid LWIN 7.
V008	Invalid LWIN 18.
V009	Web service only supports B (Bid) and O (Offer) as order type parameter.
V010	Web service only supports SIB and SEP as contract type parameter.
V011	Web service only supports L (Live) and S (Suspend) as order state parameter.
V012	Invalid request headers. Please provide value for header {header name}.
V013	Please provide valid vintage.
V015	Invalid currency.
V053	GUID is mandatory for contract type X.
V054	Parent order is not live
V055	Order details do not match order GUID
V056	GUID is not available or does not exist

7.2 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.exceeding.qty.than.stored	You are currently only allowed to create special offers. Please change your offer

7.3 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)
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.

8. Appendix – Special contracts types

Special contracts (contractType = 'X') carry four attributes that define the tax status, minimum volume, lead time and condition of a specific offer. Attributes can be combined in various ways depending on the status of the stock.

Special attribute	Meaning
dutyPaid	States whether the stock offered on the Special contract has a tax status of duty paid or not. If set to 'false' stock should be considered In Bond (IB). **Type:** Boolean 'true' or 'false'
minimumQty	States whether the seller has placed a minimum volume of units on the trade. E.g the seller has 50 units on offer with a minimumQty value of 10. **Type:** integer
deliveryPeriod	States whether the lead time on the offer is different to the standard Liv-ex terms of 2 weeks. If deliveryPeriod = 0, the offer is **Special Now** i.e. the stock is n the Liv-ex warehouse, has been checked and is ready for immediate dispatch. **Type:** integer
condition	A free text fied that states any issues with the stock or its packaging. **Type:** string

Some wine offered under a Special contract can match or exceed the Liv-ex SIB terms. Offers listed as 'Special – Now' on the exchange are the equivalent of Standard In Bond (SIB) but have the added benefit of being ready for immediate dispatch from Liv-ex warehouses.

The following combination of attributes would filter to these specific type of Special offers:

- dutyPaid = false
- minimumQty = null
- deliveryPeriod = 0
- condition = null

Offers with these flags are In Bond (IB), have no minimum quantity terms or condition issues and have already been landed and checked in the Liv-ex warehouses.

Important – feed inconsistencies

The **dutyPaid** and **condition** attributes are handled differently in the Exceloffer2 and Excelbid2 datafeeds. New, revised versions of the feeds will make then consistent with other Liv-ex services.

dutyPaid	stock is in bond = ‘true’; stock is duty paid = ‘dutypaid’
condition	no condition issue text = ‘true’; condition issues = ‘<free text string>’