LIV|EX
THE FINE WINE MARKET

Orders API v7

Document revision 1.4
Date of Issue: 20 July 2020
Date of revision: 03 Nov 2022

Barney Mullan
Business Analyst

Table of Contents

1. Purpose

To provide the API endpoint 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 is calculated based on publicly advertised prices sourced from leading merchants in the EU and Switzerland. It provides a guide as to the price you are likely to pay for SIB-compliant stock in the market.
Contract Type	Contract type is a generic term for SIB, SEP or Special (X).
SIB	Standard in Bond trade terms (link)
SEP	Standard En Primeur trade terms (link)
Special	Special contract trade terms (link)
Special Now	An offer of stock that is ready for immediate dispatch from Liv-ex warehouses.
Fat Finger	A set of checks designed to prevent the erroneous entry of price or quantity values.

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. PATCH for making partial change to existing resource
    3. DELETE for delete operation

- POST, PATCH 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 PATCH & DELETE
- Authentication mechanism will be custom based on CLIENT_KEY and CLIENT_SECRET
- The Orders 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. 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.

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.

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": "7.0",
        "timestamp": 1549537950898,
        "provider": "Liv-ex"
    }
}
```

Invalid header XML response

```
<Response>
    <Status>Unauthorized</Status>
    <HttpCode>401</Code>
    <Message>Request was unsuccessful.</Message>
    <LivexCode>R001</LivexCode>
    <ApiInfo>
        <Version>7.0</Version>
        <Timestamp>2017-11-04T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
<Response>
```

5. API Listing

5.1 Add Order Service (POST method)

Description

This service will be used to add bid or offer positions to the Liv-ex exchange. The Orders API enables users to create, modify and delete orders. Version 7 of the service supports all Liv-ex contract types (SIB, SEP and Specials).

Note that a successful POST request will be responded with an orderGUID value that should be recorded. This orderGUID reference can then be used in edit (PATCH) and delete (DELETE) requests to manipulate the bid/offer accordingly (an example of the response is included below).

Users integrating with the Orders API may choose to receive PUSH messages that automatically inform them of events take place on the Liv-ex exchange (e.g. a trade taking place for one of their bids or offers). For more information please read the document "PUSH services".

Base URI

exchange/v7/orders

Request Parameters

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

dutyPaid	N (mandatory if 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	N (mandatory if 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	N (mandatory if contractType = x)	States whether the lead time on the offer is different to the standard Liv-ex terms of 2 weeks. Value provided must be between 1 and 16. Special Now offers (where deliveryPeriod would be 0, must be placed via the Order by UID API service). **Type:** integer
condition	N (mandatory if contractType = x)	A free text field that states any issues with the stock or its packaging. **Type:** string
photoGUID	N (mandatory if contractType = x)	The GUID(s) provided in the Photo Upload API (POST) response. **Type:** string
orderType	Y	A valid type of the order. The possible values can be **B** (bid, buying position) and **O** (offer, selling position). **Type:** alphanumeric
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 provided)	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 provided)	Mandatory if LWIN7 is provided. Values are typically 1, 3, 6, 12, 24 **Type:** 2-digit integer
bottleSize	N (mandatory if LWIN7 provided)	Mandatory if LWIN7 is provided. The values must be in ml (milliliters) expressed as a 5-digit term e.g. 75cl = 00750. **Type:** 5-digit integer
currency	Y	Acceptable values are GBP, EUR, USD, HKD, GBP/btt, EUR/btt. The currency used must match the trading currency pre-agreed with Liv-ex. If you are unsure what your pre-agreed currency is, please contact your account manager. Please note if your pre-agreed currency is per bottle, then orders placed using this API will always respect the price value supplied as a price per bottle. **Type:** 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)
specialOrderGUID	N Mandatory if placing order against existing special	The order identification code of the Special position (bid or offer) that the new order should be placed against. **Type:** 128-bit hexadecimal
overrideFatFinger	N default = false	Bypass system checks that prevent keying errors. **Type:** Boolean 'true' or 'false'

Recommendation notes:

1. The POST service permits multiple orders to be supplied as an array/list. Examples of both are shown below. **To ensure consistent performance and minimize risk of failures, Liv-ex recommends sending POST requests one-by-one.**

2. A successful POST will return an orderGUID value. This is the unique identifier used within the Liv-ex system for that order. A record of each orderGUID should made as they are used to edit (PATCH) and delete (DELETE) orders. PUSH updates (if consumed) use orderGUID to identify specific events.

3. specialOrderGUID should be supplied when placing an order against a specific Special (contractType = x) that already exists on the exchange. The new order will inherit the contract terms of the Special position supplied.

4. One or more photoGUID should be supplied when placing a Special offer (contractType = x) with a condition “As per photos”. Photos need to be pre-uploaded via Photo Upload API v1 (POST) first. Different special offers cannot share the same photoGUID.

Sample Request Body

JSON Request (single order)

```
{"orders":[{
   "specialOrderGUID": "",
   "contractType": "SIB",
   "orderType": "o",
   "orderStatus": "L",
   "expiryDate": "2018-12-01",
   "lwin": "1006045",
   "vintage": "2012",
   "bottleInCase": "12",
   "bottleSize": "00750",
   "currency": "GBP",
   "price": 3400,
   "quantity": "1",
   "merchantRef": "PO #123456",
   "overrideFatFinger": false
   }]
}
```

JSON Request (multiple orders)

```
{"orders":[
   {
   "specialOrderGUID": "",
   "contractType": "X",
   "special": {
      "dutyPaid": true,
      "minimumQty": "4",
      "deliveryPeriod": 2,
      "condition": "test",
      "photoGUID": ["102JAY87X7kknr455bn6754jf43643f8L", "102JAY87X7kknr455bn6754jjgzd7673"]
      },
   "orderType": "o",
   "orderStatus": "L",
   "expiryDate": "2020-12-01",
   "lwin": "1006045",
   "vintage": "2012",
   "bottleInCase": "12",
   "bottleSize": "00750",
   "currency": "GBP",
```

```
  "price": 3200,
  "quantity": "4",
  "merchantRef": "place special offer using API",
  "overrideFatFinger": false
  },
  {
  "specialOrderGUID": "",
  "contractType": "SIB",
  "special": {
    "dutyPaid": false,
    "minimumQty": "1",
    "deliveryPeriod": 2,
    "condition": "",
    "photoGUID": []
    },
  "orderType": "o",
  "orderStatus": "L",
  "expiryDate": "2020-12-01",
  "lwin": "1006045",
  "vintage": "2012",
  "bottleInCase": "06",
  "bottleSize": "00750",
  "currency": "GBP",
  "price": 1700,
  "quantity": "1",
  "merchantRef": "place SIB offer with all POST attributes ",
  "overrideFatFinger": false
  }]
}
```


XML Request (single order)


```
<orders>
<order>
<bottleInCase>12</bottleInCase>
 <bottleSize>750</bottleSize>
 <contractType>SIB</contractType>
 <currency>GBP</currency>
 <expiryDate>2020-11-28</expiryDate>
 <lwin>100946620111200750</lwin>
 <merchantRef>Ref</merchantRef>
 <orderStatus>L</orderStatus>
 <orderType>o</orderType>
 <price>800</price>
 <quantity>1</quantity>
 <vintage>2011</vintage>
 <overrideFatFinger>false</overrideFatFinger>
</order>
</orders>
```


XML Request (multiple orders)


```
<orders>
  <order>
    <specialOrderGUID></specialOrderGUID>
    <contractType>X</contractType>
    <special>
      <dutyPaid>true</dutyPaid>
      <minimumQty>1</minimumQty>
      <deliveryPeriod>2</deliveryPeriod>
```

```
      <condition>test</condition>
      <photoGUID>12ac6222e8d6404a8738105fc3285c66</photoGUID>
    </special>
    <orderType>o</orderType>
    <orderStatus>L</orderStatus>
    <expiryDate>2020-09-28</expiryDate>
    <lwin>1006045</lwin>
    <vintage>2015</vintage>
    <bottleInCase>12</bottleInCase>
    <bottleSize>00750</bottleSize>
    <currency>gbp</currency>
    <price>3300</price>
    <quantity>4</quantity>
    <merchantRef>Placing special offer using API</merchantRef>
    <overrideFatFinger>true</overrideFatFinger>
  </order>
  <order>
    <specialOrderGUID></specialOrderGUID>
    <contractType>SIB</contractType>
    <orderType>o</orderType>
    <orderStatus>L</orderStatus>
    <expiryDate>2020-09-28</expiryDate>
    <lwin>1006045</lwin>
    <vintage>2015</vintage>
    <bottleInCase>06</bottleInCase>
    <bottleSize>00750</bottleSize>
    <currency>gbp</currency>
    <price>1700</price>
    <quantity>1</quantity>
    <merchantRef>lace SIB offer with all POST attributes</merchantRef>
    <overrideFatFinger>true</overrideFatFinger>
  </order>
</orders>
```

Sample Response Body

Response Parameters

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

JSON Response

The response is sent per order

```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully.",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "7.0",
```

```
    "timestamp": 1595251376068,
    "provider": "Liv-ex"
  },
  "orders": {
    "order": [
      {
        "merchantRef": "PO #123456",
        "orderGUID": "803ad1ff-6308-4fb9-98bc-117127d82215",
        "orderPlaceDate": 1595251375664,
        "photoGUID": null,
        "errors": null
      }
    ]
  }
}
```


Invalid JSON response


```
{
    "status": "failure",
    "httpCode": "400",
    "internalErrorCode": "R002",
    "message": "Request partially completed",
    "apiInfo": {
        "version": "7.0",
        "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


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<exchangeResponse 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>7.0</Version>
    <Timestamp>2020-07-20T14:30:56.467+01:00</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <orders>
```

```
    <order>
      <merchantRef>Ref</merchantRef>
      <orderGUID>9f5b0404-4b8f-4056-ba80-12b7c74645f9</orderGUID>
      <orderPlaceDate>2020-07-20T14:30:56.255+01:00</orderPlaceDate>
      <errors xsi:nil="true"/>
    </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>7.0</Version>
        <Timestamp>2015-06-04T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <prders>
        <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>
```

5.2 Edit Order Service (PATCH method)

Description

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

The PATCH method allows for one or more attributes to be updated. Only the attribute(s) to be updated need to be supplied in the request (alongside the ID (orderGUID) of the resource to be modified).

Note: orderGUID is the resource identifier number returned when using the POST (add) order service.

Base URI

exchange/v7/orders

Parameters

Name	Mandatory	Description
orderGUID	Y	The identifier of the order resource to be updated. **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
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)
overrideFatFinger	N default = false	Bypass system checks that prevent keying errors. **Type:** Boolean 'true' or 'false'

Sample PATCH request body

JSON

```
{"orders":[{
   "orderGUID": "7040d17d-0389-4461-83ff-a6bfcbaab809",
   "price": 3550,
   "quantity": 7,
   "merchantRef": "editing offer using PATCH method"
   }]
}
```

XML

```
<orders>
   <order>
      <orderGUID>7040d17d-0389-4461-83ff-a6bfcbaab809</orderGUID>
      <merchantRef>editing offer using PATCH method</merchantRef>
      <orderStatus>L</orderStatus>
      <price>3500</price>
      <quantity>2</quantity>
      <expiryDate>2020-09-28</expiryDate>
   </order>
</orders>
```

JSON Response

Response with valid orderGUID


```
{
   "status": "OK",
   "httpCode": "200",
   "message": "Request completed successfully.",
   "internalErrorCode": "R001",
   "apiInfo": {
      "version": "7.0",
      "timestamp": 1595252758378,
      "provider": "Liv-ex"
   },
   "orders": {
      "order": [
         {
            "merchantRef": "editing offer using PATCH meth",
            "orderGUID": "7040d17d-0389-4461-83ff-a6bfcbaab809",
            "orderPlaceDate": 1595252757979,
            "photoGUID": null,
            "errors": null
         }
      ]
   }
}
```


Response with invalid orderGUID


```
{
   "status": "Bad Request",
   "httpCode": "400",
   "message": "Request was unsuccessful.",
   "internalErrorCode": "R000",
```

```
  "apiInfo": {
    "version": "7.0",
    "timestamp": 1537441379582,
    "provider": "Liv-ex"
  },
  "orders": {
    "order": [
      {
        "merchantRef": null,
        "orderGUID": "f2f80a78-de1a-45b7-a20f-08faa19f2418",
        "orderPlaceDate": null,
        "errors": {
          "error": [
            {
              "code": "V056",
              "message": "GUID is not available or does not exist"
            }
          ]
        }
      }
    ]
  }
}
```


XML Response

Response with valid orderGUID


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<exchangeResponse 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>7.0</Version>
    <Timestamp>2020-07-20T14:50:59.363+01:00</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <orders>
    <order>
      <merchantRef>editing offer using PATCH meth</merchantRef>
      <orderGUID>7040d17d-0389-4461-83ff-a6bfcbaab809</orderGUID>
      <orderPlaceDate>2020-07-20T14:50:58.955+01:00</orderPlaceDate>
      <errors xsi:nil="true"/>
    </order>
  </orders>
</exchangeResponse>
```


Response with invalid orderGUID

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<exchangeResponse 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>7.0</Version>
    <Timestamp>2020-07-20T14:58:09.965+01:00</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <orders>
    <order>
      <merchantRef>editing offer using PATCH meth</merchantRef>
      <orderGUID>040d17d-0389-4461-83ff-a6bfcbaab809</orderGUID>
      <orderPlaceDate xsi:nil="true"/>
      <errors>
        <error>
          <code>V056</code>
          <message>GUID is not available or does not exist</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.

Note: orderGUID is the resource identifier number returned when using the POST (add) order service.

Base URI

exchange/v7/orders

Parameters

Name	Mandatory	Description
orderGUID	Y	A valid order GUID for the account.

Recommendation notes:

1. The Orders DELETE service permits multiple GUIDs be supplied as an array/list. **To ensure consistent performance and minimize risk of failures, Liv-ex recommends sending DELETE requests one-by-one.**

Sample JSON DELETE request body

JSON

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

XML

```
<orders>
  <order>
     <orderGUID>2a1dd836-9dca-4ef8-85f7-652cb8b175d2</orderGUID>
  </order>
</orders>
```

JSON Response

Response with valid order ID


```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully.",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "7.0",
    "timestamp": 1595254715057,
    "provider": "Liv-ex"
  },
  "orders": {
    "order": [
      {
        "merchantRef": "test",
        "orderGUID": "64f65268-6136-4d4b-b724-97245bde9cf5",
        "orderPlaceDate": 1595254714473,
        "photoGUID": null,
        "errors": null
      }
    ]
  }
}
```


Response with invalid order ID

```
{
  "status": "Bad Request",
  "httpCode": "400",
  "message": "Request was unsuccessful.",
  "internalErrorCode": "R000",
  "apiInfo": {
    "version": "7.0",
    "timestamp": 1595254831684,
    "provider": "Liv-ex"
  },
  "orders": {
    "order": [
      {
        "merchantRef": null,
        "orderGUID": "64f65268-6136-4d4b-b724-97245bde9cf5",
        "orderPlaceDate": null,
        "photoGUID": null,
        "errors": {
          "error": [
            {
              "code": "V002",
              "message": "Invalid parameter(orderGUID)."
            }
          ]
        }
      }
    ]
  }
}
```

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://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>7.0</Version>
    <Timestamp>2020-07-20T15:19:45.858+01:00</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <orders>
    <order>
      <merchantRef>photo 3</merchantRef>
      <orderGUID>2a1dd836-9dca-4ef8-85f7-652cb8b175d2</orderGUID>
      <orderPlaceDate>2020-07-20T15:19:45.372+01:00</orderPlaceDate>
      <errors xsi:nil="true"/>
    </order>
  </orders>
</exchangeResponse>
```

Response with invalid order ID

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<exchangeResponse 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>7.0</Version>
    <Timestamp>2020-07-20T15:20:04.883+01:00</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <orders>
    <order>
      <merchantRef xsi:nil="true"/>
      <orderGUID>2a1dd836-9dca-4ef8-85f7-652cb8b175d2</orderGUID>
      <orderPlaceDate xsi:nil="true"/>
      <errors>
        <error>
          <code>V002</code>
          <message>Invalid parameter(orderGUID).</message>
        </error>
      </errors>
    </order>
  </orders>
</exchangeResponse>
```

6. Supplementary API services

orders PUSH service – real-time messaging on trade events that involve your orders

heartbeat API – check the exchange is available

orderStatus API – check the status of specific bid/offer positions on the exchange

orderByUID API – sell stock you have stored in the Liv-ex Vine warehouse as Special Now

myPositions API – view and reconcile all your positions with Liv-ex (live or suspended)

bulkOrderAction API – suspend, reactivate and renew positions on the exchange in bulk

photoUpload API – upload photos before placing a special offer

photoView API – check available photos

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.
V017	Merchant does not have the EP limit assigned for vintage <value>.
V018	Mandatory field missing (<value>).
V053	GUID is mandatory for contract type X.
V054	Parent order is not live.
V055	Order details do not match orderGUID.
V056	orderGUID is not available or does not exist.
V064	Invalid / incorrect lwin and vintage : [%s] combination.
V072	Quantity change is not allowed in this order.
V073	Invalid / incorrect dutyPaid: [<value>]. Possible values are 'true', 'false'.
V074	Invalid / incorrect deliveryPeriod: [<value>]. Must be positive integer value. Max value = 16 weeks
V075	Invalid / incorrect minimumQty: [<value>]. Must be a positive integer value.
V077	Invalid / incorrect contractType: [<value>]. Possible values can be 'sib' (Standard In Bond), 'sep' (Standard En Primeur) and 'x' (Special).
V078	Missing attribute: [<value>]. When orderType = 'o' and contractType = 'x', attributes dutyPaid, minimumQty and deliveryPeriod must be included.
V085	Parent and child orders can't have same order type.
V086	Please provide valid special terms of contract to create a special order.
V087	Contract type change is not allowed in this order.
V148	photoGUID [${v}] does not exist or is already in use.

7.2 Trade validation error codes

Code	Error Key	Meaning
TR001	invalid.made.avail	The deliver period supplied is not valid. Must be a positive integer value (max value = 16)
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
TR013	merchant.awrs.urn.not.found	Merchant is placing a duty paid offer but does not have a validated AWRS number
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

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 field that states any issues with the stock or its packaging. **Type:** string
photoGUID	The GUID(s) provided in the Photo Upload API (POST) response. **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 – attribute inconsistencies between Datafeeds and Orders API

For customers using the "Exceloffer2" and "Excelbid2" feeds to pull in the latest bid and offers, note that the **dutyPaid** and **condition** attributes are handled differently in two feeds:

dutyPaid stock is in bond = 'true'; stock is duty paid = 'dutypaid'

condition no condition issue text = 'true'; condition issues = '<free text string>'