LIV|EX
THE FINE WINE MARKET

PUSH services

Document revision 1.2
Date of Issue: 04 October 2018
Date of revision: 25 June 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 PUSH notifications

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. PUSH for update operation, delivering POST to consuming systems
- 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
- For PUSH services we require a direct POST URL which should be backed by a service capable of accepting and process JSON or XML payload as POST request.

4. API Listing

4.1 PUSH notifications

Description

A PUSH message will be invoked when triggered by specific Liv-ex system events. The information will be sent to a user's system as a POST 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. Both HTTP and HTTPS urls are supported

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. A POST 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.

HEAD requests have the following user-agent value:

```
Mozilla/5.0 (Macintosh; Intel Mac OS X x.y; rv:42.0) Gecko/20100101 Firefox/42.0
```

Merchant systems must respond to the HEAD 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.

4.1.1 Confirm Trade

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	The GUID of trade event (note that this is different to the GUID of your bid or offer).
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.

4.1.2 Order Update

Individual order update messages are PUSHed for the following reasons:

1	Order Created
2	Order Created via LX3
3	Order Deleted
4	Order Deleted via LX3
5	Order Edited
6	Order Edited via API
7	Order Suspended

8	Order Unsuspended
9	Order Blocked
10	Order Unblocked
11	Order expired by scheduler
12	Special order no longer available. Your position has been deleted
13	Order forex rate updated due to threshold hit

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>1014307200812007 50</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": "1014307200812007 50",
        "price": "240",
        "qty": "1",
        "order_update_date": "2015-06-04T07:22:25"
    }
}
```

Param

Name	Description
order_guid	Bid/offer order GUID that has been updated
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. Response Codes

This section describes the response codes that will be returned by the Exchange

5.1 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.