LIV|EX

THE FINE WINE MARKET

LWIN: Request Status Check API v1

Document Revision 1.0
Date of Issue: 24 January 2020
Date of revision: 28 May2020

Daria Ershova

Table of Contents

1. Purpose

To provide the API end point information and examples of the web services available for LWIN Request Status Check API.

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. This document refers to LWIN7 (7-digit, wine) and LWIN11 (11-digit, wine + vintage) codes only.
Combine	When two LWINs for the same wine have been erroneously created codes are "combined". This preserves the overall data of the product. The dominant LWIN is known as the "leader", whils the LWIN that has been merged away is know as the "follower".

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 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
- The APIs will be accessible at https://api.liv-ex.com/ followed by their specific base URIs

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 user.
CLIENT_SECRET	Y	Password/Secret for the user 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": "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>2019-11-04T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
<Response>
```

5. API Listing

5.1 Request Status Check service (POST method)

Description

This service allows users to check status of their request for a new LWIN7 / LWIN11. One status request is permitted per API call.

Base URI

/ lwin/request/v1/requestStatusCheck

Request Parameters

Name	Mandatory	Description
requestReference	Y	LWIN 7/11 request reference received via LWIN Request services **Type:** integer (11)

Sample Request Body

JSON

```
{
"requestReference": "3911"
}
```

XML

```
<requestStatusCheck>
    <requestReference>3911</requestReference>
</requestStatusCheck>
```

Sample Response Body

The Request Status Check POST service returns current status for the provided requestReference, as well as feedback (if any) and metadata for approved LWIN7 requests.

Response parameters

Name	Description
requestReference	**Type:** integer(11)
requestStatus	**Type:** alphanumeric **Vallues:** "accepted", "rejected", "assigned", "pending"
feedback	**Type:** alphanumeric Max 250 characters
lwin	**Type:** integer(7|11)
producerTitle	Title of producer or owner of wine **Type:** alphanumeric

<table>
<tr><td>producerName</td><td>Producer or owner of wine
Type: alphanumeric</td></tr>
<tr><td>wine</td><td>Name of wine (brand and/or grape and/or technical term)
Type: alphanumeric</td></tr>
<tr><td>country</td><td>Country of origin
Type: alphanumeric</td></tr>
<tr><td>region</td><td>Region of origin (where applicable, specific to local laws)
Type: alphanumeric</td></tr>
<tr><td>subRegion</td><td>Sub-region of origin (where applicable, specific to local laws)
Type: alphanumeric</td></tr>
<tr><td>site</td><td>Site within sub-region (where applicable, specific to local laws)
Type: alphanumeric</td></tr>
<tr><td>parcel</td><td>Parcel within site (where applicable, specific to local laws)
Type: alphanumeric</td></tr>
<tr><td>colour</td><td>Colour of LWIN product
Type: alphanumeric
Values: "white", "red", "rose", null</td></tr>
<tr><td>type</td><td>LWIN beverage type
Type: alphanumeric
Values: "wine", "fortified", "spirit", "rice wine", "beer", "other"</td></tr>
<tr><td>subType</td><td>Subcategory of LWIN type
Type: alphanumeric
Values:
<table>
<tr><td>wine</td><td>"still", "sparkling"</td></tr>
<tr><td>fortified</td><td>"madeira", "port", "sherry"</td></tr>
<tr><td>spirit</td><td>"brandy", "whiskies", "vodka"</td></tr>
<tr><td>rice wine</td><td>"sake"</td></tr>
<tr><td>beer</td><td></td></tr>
<tr><td>other</td><td>"na"</td></tr>
</table></td></tr>
<tr><td>designation</td><td>Officially assigned status (specific to local laws)
Type: alphanumeric</td></tr>
<tr><td>classification</td><td>Officially declared quality level (where applicable, specific to local laws)
Type: alphanumeric</td></tr>
<tr><td>vintageConfiguration</td><td>Vintage pattern of the wine (e.g. single vintage only, production ended)
Type: alphanumeric
Values: "sequential", "nonSequential", "singleVintageOnly".</td></tr>
<tr><td>vintageValues</td><td>Sorting order: youngest vintage → oldest vintage
Type: array, integer(4)</td></tr>
<tr><td>firstVintage</td><td>The first vintage of the LWIN7
Type: integer(4)</td></tr>
<tr><td>finalVintage</td><td>The final vintage of the LWIN7 (only if production ended)
Type: integer(4)</td></tr>
<tr><td>childOf</td><td>Parent wine LWIN7. This should be used to map associated wines to one another. This is a separate relationship to LWIN merge references
Type: integer (7)</td></tr>
<tr><td>displayName</td><td>Type: alphanumeric</td></tr>
<tr><td>dateCreated</td><td>Type: datetime / epoch</td></tr>
</table>

lastUpdateDate	**Type:** datetime / epoch
status	**Type:** alphanumeric **Values:** "live", "deleted", "combined"
combineReference	**Type:** integer(7)

JSON Response

The response is sent per request.

LWIN7


```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1579854458964,
        "provider": "Liv-ex"
    },
    "requestStatusCheck": {
        "requestReference": "9208",
        "requestStatus": "accepted",
        "feedback": null,
        "lwin": "2576932",
        "metadata": {
            "producerTitle": "Bodega",
            "producerName": "Garzom",
            "wine": "Tannat",
            "country": "Uruguay",
            "region": null,
            "subRegion": null,
            "site": null,
            "parcel": null,
            "colour": "Red",
            "type": "Wine",
            "subType": "Still",
            "designation": null,
            "classification": null,
            "vintageConfiguration": "sequential",
            "vintageValues": [
                "2017"
            ],
            "firstVintage": "2016",
            "finalVintage": "2017",
            "childOf": null,
            "displayName": "Bodega Garzon, Tannat",
            "dateCreated": 1579851649000,
            "lastUpdateDate": 1579851649000,
            "status": "live",
            "combineReference": null
        },
        "errors": null
    }
}
```


LWIN11

```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1579854109532,
        "provider": "Liv-ex"
    },
    "requestStatusCheck": {
        "requestReference": "9210",
        "requestStatus": "accepted",
        "feedback": null,
        "lwin": "10082831996",
        "metadata": null,
        "errors": null
    }
}
```


Invalid JSON response


```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1579854185065,
        "provider": "Liv-ex"
    },
    "requestStatusCheck": {
        "requestReference": "3112",
        "errors": {
            "error": [
                {
                    "code": "L035",
                    "message": "Invalid / incorrect requestReference 3112 provided."
                }
            ]
        }
    }
}
```


XML Response

The response is sent per request.

LWIN7


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<requestStatusCheckResponse>
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2020-01-24T08:42:17.661Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <requestStatusCheck>
        <requestReference>9212</requestReference>
        <requestStatus>pending</requestStatus>
        <feedback xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
        <lwin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
```

```
        <metadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
        <errors xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
    </requestStatusCheck>
</requestStatusCheckResponse>
```

LWIN11

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<requestStatusCheckResponse>
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2020-01-24T08:43:52.309Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <requestStatusCheck>
        <requestReference>9213</requestReference>
        <requestStatus>rejected</requestStatus>
        <feedback>was not made</feedback>
        <lwin>25769322013</lwin>
        <metadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
        <errors xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
    </requestStatusCheck>
</requestStatusCheckResponse>
```

Invalid XML Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<requestStatusCheckResponse>
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2020-01-24T08:23:53.266Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <requestStatusCheck>
        <requestReference></requestReference>
        <errors>
            <error>
                <code>L001</code>
                <message>Mandatory field requestReference missing.</message>
            </error>
        </errors>
    </requestStatusCheck>
</requestStatusCheckResponse>
```

6. Response Codes

This section describes the response codes that will be returned by the LWIN services.

Code	Message
R000	Request was unsuccessful
R001	Request completed successfully
R002	Request partially completed

6.1 Request validation error codes

Code	Message
L001	("Mandatory field [%s] missing.")

6.2 HTTP Status codes

HTTP defines a bunch of meaningful status codes that can be returned from our API. These can be leveraged to help our API Merchants/consumers route their responses accordingly:

Code	Message
200 OK	Response to a successful GET, POST, PUT, DELETE. Can also be used for a POST that doesn't result in a creation.
201 Created	Response to a POST that results in a creation.

202 Accepted	The request has been accepted and will be processed later. It is a classic answer to asynchronous calls (for better UX or performances).
204 No Content	Response to a successful request that won't be returning a body (like a DELETE request)
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.