LIV|EX
THE FINE WINE MARKET

Photo Upload API v1

Document revision 1.0
Date of Issue: 17 July 2020
Date of revision: 17 July 2020

Daria Ershova

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.

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. DELETE for delete operation
- POST 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.

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 and DELETE requests	Content-type is a way to specify the media type of request being sent from the client to the server. The value for the content type will be multipart/form-data for POST requests. The values for the content type will be application/json or application/xml for DELETE requests. If no/ invalid content type is found in the request, then JSON format will be used by default.

Example header

POST

```
CLIENT_KEY: 12A34BC56-DE7F-89G0-H1J2345K678L
CLIENT_SECRET: dummy_password
ACCEPT: application/json
CONTENT-TYPE: multipart/form-data
```

DELETE

```
CLIENT_KEY: 12A34BC56-DE7F-89G0-H1J2345K678L
CLIENT_SECRET: dummy_password
ACCEPT: application/json
CONTENT-TYPE: application/json
```

Invalid header JSON response

```
{
    "status": "Unauthorized",
    "httpCode": "401",
    "message": "Request was unsuccessful",
    "livexCode": "R000"
    "apiInfo": {
        "version": "1.0",
        "timestamp": "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>1.0</Version>
        <Timestamp>2017-11-04T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
<Response>
```

5. Photo Upload API

5.1 Add Photo Service (POST method)

Description

This service will be used to add photos that can be linked to special offers. The Photo Upload API enables users to upload and delete photos.

A successful POST request will be response with a photoGUID values that would be recorded. This photoGUID reference can be used to place a special order with photos via Orders v7 POST and delete uploaded photo via Photo Upload DELETE.

Base URI

/photo/v1/photoUpload

Content Type

multipart/form-data

Request Parameters

Name	Mandatory	Description
photo	Y	Several files can be submitted in one request. Maximum payload size is 20MB. **Type:** .png, .jpeg and .jpg
reference	N	An optisonal free text field that can be used to attach a useful reference to the image files being uploaded. Strings exceeding 300 characters will be truncated. **Type:** alphanumeric (max 300 characters)

Sample Response Body

Name	Description

photoGUID	The GUID asigned to each uploaded photo. The photoGUID is required when sending Orders v7 POST request for placing special offers. **Type:** 128-bit hexadecimal **Example:** "43f4ac34d99f4637aeee1a247aea1937"
inputFileName	User's input file name. **Type:** alphanumeric
reference	Reference provided in the request **Type:** alphanumeric
highResolutionPhotoUrl	Link to a high resolution photo **Type:** alphanumeric
lowResolutionPhotoUrl	Link to a low resolution photo **Type:** alphanumeric
expiryDate	**Type:** alphanumeric, ISO 8601. Epoch time if JSON

JSON Response


```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1595000562875,
    "provider": "Liv-ex"
  },
  "photoUpload": [
    {
      "photoGUID": "5c0a1e4e00f141b491ebe2f405e086c6",
      "inputFileName": "14BRA750DOM_01.png",
      "reference": "test 2",
      "highResolutionPhotoUrl":
"https://dcoygdx2udjif.cloudfront.net/org/5c0a1e4e00f141b491ebe2f405e086c6.png?Expires=1595004162&Signature
",
      "lowResolutionPhotoUrl":
"https://dcoygdx2udjif.cloudfront.net/mod/5c0a1e4e00f141b491ebe2f405e086c6.png?Expires=1595004162&Signatur
e=lvV4S0xtvPx9 ",
      "expiryDate": 1595138400000,
      "errors": null
    }
  ],
  "errors": null
}
```


Multi-status JSON response

```
{
  "status": "Multi-Status",
  "httpCode": "207",
  "message": "Few requests were unsuccessful",
  "internalErrorCode": "R002",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1594975173598,
    "provider": "Liv-ex"
  },
  "photoUpload": [
    {
      "photoGUID": "12ac6222e8d6404a8738105fc3285c66",
      "inputFileName": "14BRA750DOM_01.png",
      "reference": "test",
      "highResolutionPhotoUrl":
"https://dcoygdx2udjif.cloudfront.net/org/12ac6222e8d6404a8738105fc3285c66.png?Expires=1594978773&Signature
=OQl9vBraCEuv- ",
      "lowResolutionPhotoUrl":
"https://dcoygdx2udjif.cloudfront.net/mod/12ac6222e8d6404a8738105fc3285c66.png?Expires=1594978773&Signatur
e=ncWq-QvdrruGATQ ",
      "expiryDate": 1595138400000,
      "errors": null
    },
    {
      "photoGUID": null,
      "inputFileName": "",
      "reference": "test",
      "highResolutionPhotoUrl": null,
      "lowResolutionPhotoUrl": null,
      "expiryDate": null,
      "errors": {
        "error": [
          {
            "code": "V156",
            "message": "Invalid file type: []. Supported types are '.png', '.jpeg' and '.jpg'."
          }
        ]
      }
    }
  ],
  "errors": null
}
```


XML Response


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<photoResponse>
  <Status>OK</Status>
  <HttpCode>200</HttpCode>
  <Message>Request completed successfully</Message>
  <InternalErrorCode>R001</InternalErrorCode>
  <ApiInfo>
    <Version>1.0</Version>
    <Timestamp>2020-07-17T15:45:27.667Z</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <photoUpload>
    <photoGUID>f79562ee2d27449eb19d1701dba9511c</photoGUID>
    <inputFileName>10112471997 1.JPG</inputFileName>
    <reference>XML Test</reference>
```

```
<highResolutionPhotoUrl>https://dcoygdx2udjif.cloudfront.net/org/f79562ee2d27449eb19d1701dba9511c.JPG?Expire
s=1595004327&amp;Signature=dM3s6uC64~s55oDPD23YBqvClar9Krh7LSLbvHIqp6DtN6a9tBZ6EEGBa7mA-
RwildogC8fB0JDWJJSzFOVD </highResolutionPhotoUrl>
<lowResolutionPhotoUrl>https://dcoygdx2udjif.cloudfront.net/mod/f79562ee2d27449eb19d1701dba9511c.JPG?Expir
es=1595004327&amp;Signature=mf-3wby5whd4pMauaDef7wgWd7mw78gWlfGtPSzr7t1MyObVWPSyJK
</lowResolutionPhotoUrl>
      <expiryDate>2020-07-19T06:00:00Z</expiryDate>
   </photoUpload>
</photoResponse>
```

Invalid XML Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<photoResponse>
   <Status>Multi-Status</Status>
   <HttpCode>207</HttpCode>
   <Message>Few requests were unsuccessful</Message>
   <InternalErrorCode>R002</InternalErrorCode>
   <ApiInfo>
      <Version>1.0</Version>
      <Timestamp>2020-07-17T15:46:42.654Z</Timestamp>
      <Provider>Liv-ex</Provider>
   </ApiInfo>
   <photoUpload>
      <photoGUID>c7b5a4ea8a4c428091e44d58732f24f4</photoGUID>
      <inputFileName>10112471997 1.JPG</inputFileName>
      <reference>XML Test</reference>

<highResolutionPhotoUrl>https://dcoygdx2udjif.cloudfront.net/org/c7b5a4ea8a4c428091e44d58732f24f4</highResol
utionPhotoUrl>

<lowResolutionPhotoUrl>https://dcoygdx2udjif.cloudfront.net/mod/c7b5a4ea8a4c428091e44d58732f24f4.JPG?Exp</l
owResolutionPhotoUrl>
      <expiryDate>2020-07-19T06:00:00Z</expiryDate>
   </photoUpload>
   <photoUpload>
      <inputFileName></inputFileName>
      <reference>XML Test</reference>
      <errors>
         <error>
            <code>V156</code>
            <message>Invalid file type: []. Supported types are '.png', '.jpeg' and '.jpg'.</message>
         </error>
      </errors>
   </photoUpload>
</photoResponse>
```

5.2 Delete Photo Service (DELETE method)

Description

This webservice will be used to delete photo(s) of a merchant.

Note: phtotoGUID is the resource identifier number returned when using the POST (upload) photo upload service.

Base URI

/photo/v1/photoUpload

Parameters

Name	Mandatory	Description
photoGUID	Y	Valid photoGUID(s) which is not linked to any special offers. **Type:** 128-bit hexadecimal **Example:** "43f4ac34d99f4637aeee1a247aea1937"

Sample Request Body

JSON

```
{
  "photoUpload": {
  "photoGUID": ["441da267203f4d17921dc6fa874c7cfa", "504e14771cf047f287d1014f816eb126"]
  }
}
```

XML

```
<photoUpload>
 <photoGUID>3978d827111e4715870872708239033 2</photoGUID>
 <photoGUID>ae96324a11b340148d1076ccf422e540</photoGUID>
</photoUpload>
```

Response parameters

Name	Description
photoGUID	**Type:** 128-bit hexadecimal **Example:** "43f4ac34d99f4637aeee1a247aea1937"

Sample Response Body

JSON Response

Multi-status JSON response

```
{
  "status": "Multi-Status",
  "httpCode": "207",
  "message": "Few requests were unsuccessful",
  "internalErrorCode": "R002",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1594971597041,
    "provider": "Liv-ex"
  },
```

```
  "photoUpload": [
    {
      "photoGUID": "441da267203f4d17921dc6fa874c7cfa",
      "error": {
        "code": "V148",
        "message": "photoGUID [441da267203f4d17921dc6fa874c7cfa] does not exist or is already in use."
      }
    },
    {
      "photoGUID": "504e14771cf047f287d1014f816eb126",
      "error": null
    }
  ],
  "errors": null
}
```


Invalid JSON response


```
{
  "status": "Multi-Status",
  "httpCode": "207",
  "message": "Few requests were unsuccessful",
  "internalErrorCode": "R002",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1595001362772,
    "provider": "Liv-ex"
  },
  "photoUpload": [
    {
      "photoGUID": "441da267203df7921dc6fa874c7cfa",
      "error": {
        "code": "V148",
        "message": "photoGUID [441da267203df7921dc6fa874c7cfa] does not exist or is already in use."
      }
    }
  ],
  "errors": null
}
```

XML Response

Multi-status XML response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<photoResponse>
  <Status>Multi-Status</Status>
  <HttpCode>207</HttpCode>
  <Message>Few requests were unsuccessful</Message>
  <InternalErrorCode>R002</InternalErrorCode>
  <ApiInfo>
    <Version>1.0</Version>
    <Timestamp>2020-07-17T07:40:26.726Z</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <photoUpload>
    <photoGUID>3978d827111e47158708727082390332</photoGUID>
    <error>
      <code>V148</code>
      <message>photoGUID [3978d827111e47158708727082390332] does not exist or is already in use.</message>
    </error>
  </photoUpload>
  <photoUpload>
    <photoGUID>437b48f876d24bbf955f31c6d4c536ad</photoGUID>
  </photoUpload>
  <photoUpload>
    <photoGUID>ae96324a11b340148d1076ccf422e540</photoGUID>
    <error>
      <code>V148</code>
      <message>photoGUID [ae96324a11b340148d1076ccf422e540] does not exist or is already in use.</message>
    </error>
  </photoUpload>
</photoResponse>
```

Invalid XML response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<photoResponse>
  <Status>Multi-Status</Status>
  <HttpCode>207</HttpCode>
  <Message>Few requests were unsuccessful</Message>
  <InternalErrorCode>R002</InternalErrorCode>
  <ApiInfo>
    <Version>1.0</Version>
    <Timestamp>2020-07-17T15:57:52.091Z</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <photoUpload>
    <photoGUID>3978d827111e47158708727082390332</photoGUID>
    <error>
      <code>V148</code>
      <message>photoGUID [3978d827111e47158708727082390332] does not exist or is already in use.</message>
    </error>
  </photoUpload>
</photoResponse>
```

6. Response Codes

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

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

6.1 Request validation error codes

Code	Message
V148	photoGUID [${v}] does not exist or is already in use.
V156	Invalid file type: [%s]. Supported types are '.png', '.jpeg' and '.jpg'.

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
413 Payload Too Large	If payload exceeds 20 MB.
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.