LIV|EX

THE FINE WINE MARKET

Merchant Names API v1

Document Revision 1.0
Date of Issue: 08 July 2022
Date of revision: 08 July 2022

Barnabas Mullan
Business Analyst

Table of Contents

1. Purpose 3
2. Technical Standards 3
3. Request Header 3
4. API Listing 4
4.1 Merchant Names service (GET method) 4
5. Response Codes 5
5.1 HTTP Status codes 6

1. Purpose

To provide the API end point information and examples of the web services available for the Merchant Names API

2. 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 contents 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:
    - GET for read 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

3. 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.

e.g.

Example header

```
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>2021-07-01T11:12:30</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
<Response>
```

4. API Listing

4.1 Merchant Names service (GET method)

Description

This service will be used to retrieve a list of merchant names which can be used as the merchantNames request parameter in the List Comparison API.

Base URI

data/v1/merchantNames

Sample Response Body

Response parameters

Name	Description
merchantName	Name of a merchant on your peer list. Type: Alphanumeric

JSON Response

The response is sent per request. The example below shows the response for the JSON Request example.


```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1657284853091,
        "provider": "Liv-ex"
    },
    "merchantNames": [
        {
            "merchantName": "Example 1"
        },
        {
            "merchantName": "Example 2"
        },
        {
            "merchantName": "Example 3"
        }
    ],
    "errors": null
}
```


XML Response

The response is sent per request. The example below shows the response for the XML Request example.


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<merchantNamesResponse>
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2022-07-08T12:54:13.091Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <merchantNamesList>
        <merchantNames>
            <merchantName>Example 1</merchantName>
        </merchantNames>
        <merchantNames>
            <merchantName>Example 2</merchantName>
        </merchantNames>
        <merchantNames>
            <merchantName>Example 3</merchantName>
        </merchantNames>
    </merchantNamesList>
</merchantNamesResponse>
```


5. 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

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.