Indices Components API v1

Document Revision 1.1
Date of Issue: 02 September 2021
Date of revision: 14 March 2024

Barnabas Mullan

Senior Business Analyst

Table of Contents

1. Purpose

To provide the API end point information and examples of the web services available for the Indices Components API.

2. Glossary of Terms

Term	Meaning
Liv-ex Index	Liv-ex has created several indices to track the price of fine wine, including the industry benchmark Liv-ex 100. The Liv-ex indices which track the prices of the most traded fine wines on the market, using the Liv-ex Mid Price.
Mid price	The mid-point between the highest live bid and lowest live offer on the market, validated against additional data including transaction prices. Mid-prices are used to calculate the Liv-ex indices and value the world's leading fine wine funds.

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
- 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 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.
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>
```

5. API Listing

5.1 Indices Components service (POST method)

Description

This service will be used to request a list of components that make up a specified Liv-ex Index.

Base URI

data/v1/indexComponents

URI Pagination Parameters

Name	Mandatory	Description
?limit	N default = 250	Values: Any value between '1' and '1000' **Type:** integer
?offset	N default = 1	**Type:** alphanumeric

Example base URI including pagination: data/v1/indexComponents?offset=1&limit=250

Request Parameters

Name	Mandatory	Description
indexName	Y	Values: any valid Liv-ex Index name, i.e. “Liv-ex Fine Wine 100”. This parameter is not case sensitive. Type: alphanumeric

Sample Request Body

JSON Request

```
{
    "indexComponents":{
        "indexName":"Liv-ex fine Wine 100"
    }
}
```

XML Request

```
<indexComponentsRequest>
    <indexComponents>
        <indexName>Liv-ex Fine Wine 100</indexName>
    </indexComponents>
</indexComponentsRequest>
```

Sample Response Body

Response parameters

Name	Description
indexName	The name of the Liv-ex index Type: alphanumeric
subIndexOf	Notes the parent index of the specific Liv-ex Index Type: alphanumeric
lwin	LWIN11 Type: alphanumeric
lwinName	Vintage-specific LWIN wine name description e.g. ‘Chateau Grand-Puy-Lacoste 5eme Cru Classe, Pauillac’ Type: alphanumeric
lwinRegion	LWIN11 region Type: alphanumeric
vintage	Vintage Type: alphanumeric

iwp	Link to the individual wine page (IWP) of the LWIN on the Liv-ex exchange application Type: alphanumeric
monthOnMonthChange	Percentage change in MID price vs 1 calendar month previous Type: double
yearToDateChange	Percentage change in MID price vs the start of the calendar year Type: double
oneYearChange	Percentage change in MID price vs 1 calendar year previous Type: double
twoYearChange	Percentage change in MID price vs 2 calendar years previous Type: double
fiveYearChange	Percentage change in MID price vs calendar years previous Type: double

JSON Response

The response is sent per request. The example below shows the response to a request for the index components of the Liv-ex Fine Wine 100, truncated down to one response cluster with anonymized data.


```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1630591334772,
        "provider": "Liv-ex"
    },
    "pageInfo": {
        "totalResults": 100,
        "limit": 1,
        "offset": 1
    },
    "indexComponents": {
        "indexName": "Liv-ex Fine Wine 100",
        "subIndexOf": null,
        "list": [
            {
                "lwin": 10112471995,
                "lwinName": "Chateau Haut-Brion Premier Cru Classe, Pessac-Leognan",
                "lwinRegion": "Bordeaux",
                "vintage": 1995,
                "iwp": "https://app.liv-ex.com/#/wine-page?lwin11=10112471995",
                "monthOnMonthChange": 000.00,
                "yearToDateChange": 000.00,
                "oneYearChange": 000.00,
                "twoYearChange": 000.00,
                "fiveYearChange": 000.00
            },
        ]
    },
    "errors": null
}
```


Invalid JSON response


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

```
    "internalErrorCode": "R000",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1630597737225,
        "provider": "Liv-ex"
    },
    "pageInfo": null,
    "indexComponents": null,
    "errors": {
        "error": [
            {
                "code": "V169",
                "message": "Invalid / incorrect internalIndex name: [Liv-ex fine Wine 10]."
            }
        ]
    }
}
```


XML Response

The response is sent per request. The example below shows the response to a request for the index components of the Liv-ex Fine Wine 100, truncated down to one response cluster with anonymized data.


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<indexComponentsResponse>
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2021-09-02T15:50:03.129Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <pageInfo>
        <totalResults>81</totalResults>
        <limit>100</limit>
        <offset>1</offset>
    </pageInfo>
    <indexComponents>
        <indexName>Liv-ex Fine Wine 100</indexName>
        <subIndexOf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
        <list>
            <lwin>10062052009</lwin>
            <lwinName>Chateau Ausone Premier Grand Cru Classe A, Saint-Emilion Grand
Cru</lwinName>
            <lwinRegion>Bordeaux</lwinRegion>
            <vintage>2009</vintage>
            <iwp>https:// app.liv-ex.com/#/wine-page?lwin11=10062052009</iwp>
            <monthOnMonthChange xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
            <yearToDateChange>000.00</yearToDateChange>
            <oneYearChange>-000.00</oneYearChange>
            <twoYearChange>-000.00</twoYearChange>
            <fiveYearChange>000.00</fiveYearChange>
        </list>
    </indexComponents>
</indexComponentsResponse>
```


Invalid XML Response

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<indexComponentsResponse>
    <Status>Bad Request</Status>
    <HttpCode>400</HttpCode>
    <Message>Request was unsuccessful</Message>
    <InternalErrorCode>R000</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2021-09-02T15:52:46.017Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <errors>
        <error>
            <code>V169</code>
            <message>Invalid / incorrect internalIndex name: [Liv-ex Fine Wine 10].</message>
        </error>
    </errors>
</indexComponentsResponse>
```


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
V169	Invalid / incorrect internalIndex name: [%s].

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.