LIV|EX

THE FINE WINE MARKET

Critic Data Change Since API v1

Document Revision 1.0
Date of Issue: 16 February 2021
Date of revision: 16 February 2021

Table of Contents

1. Purpose

To provide the API end point information and examples of the web services available for Critic Data Change Since.

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 service supports ISO 8601.
- The service only support HTTPS protocol for client and server communications.
- The API will support the following methods:
    - POST 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

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.

Parameter

Name	Mandatory	Description
CLIENT_KEY	Y	A valid GUID which will be unique for each user.
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.

Example header

```
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": "Unauthorized",
    "internalErrorCode": null,
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1554364615297,
        "provider": "Liv-ex"
    }
}
```

Invalid header (XML response)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://aby-qa-api.liv-ex.com/v1 https://aby-qa-api.liv-
ex.com/schema/v1/services.xsd">
    <Status>Unauthorized</Status>
    <HttpCode>401</HttpCode>
    <Message>Unauthorized</Message>
    <InternalErrorCode xsi:nil="true"/>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2019-04-04T12:02:37.092+01:00</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
</Response>
```

5. API Listing

5.1 Critic Data Change Since service (POST method)

Description

This service allows users to request latest critic reviews for a specified timeframe. The user must hold a valid license with each publication to be granted access to scores and data. The API service returns scores, notes and details of the critic, publication, and review date.

The data available via this API is shared with Liv-ex by each publication. Depending on their busines model they may not permit historical notes to be available or tasting notes to be included. Therefore, the data available on a given wine may vary depending on the publication requested.

Base URI

/critic/data/v1/criticDataChangeSince

URI Pagination Parameters

Name	Mandatory	Description
?limit	N default = 50	Values: Any value between '1' and '50' **Type:** integer
?offset	N default = 1	**Type:** alphanumeric

Example base URI including pagination: /critic/data/v1/criticDataChangeSince?offset=1&limit=25

Request Parameters

Name	Mandatory	Description
timeframe	N Default = 1 day	The time window of critic data change to return. **Type:** alphanumeric **Values:** '1day', '1week', '2week', '1month', '3month'
changeSince	N	Type: time YYYY-MM-DD HH:mm Maximum 3 months If both timeframe and changeSince values are provided, changeSince value will be ignored
publication	Y	The name of the publication to retrieve scores from. Case insensitive. Only one value is permitted per request. For notes from all publications subscribed use value "allSubscribed" A list of valid publication names can be called via a separate GET API. Please contact Liv-ex for more details. **Type:** alphanumeric **Example:** "vinous", "allSubscribed"
reviewer	N	The name of the reviewer. Only one value is permitted per request. Case insensitive. **Type:** alphanumeric **Example:** "neal martin"

Sample Request Body

JSON Request

```
{
"criticDataChangeSince": {
"changeSince": "2021-02-01 00:00",
"publication": "allSubscribed"
}
}
```

XML Request

```
<criticDataChangeSinceRequest>
    <criticDataChangeSince>
        <timeframe>1month</timeframe>
        <publication>Vinous</publication>
        <reviewer>Neal Martin</reviewer>
    </criticDataChangeSince>
</criticDataChangeSinceRequest>
```

Sample Response Body

The Critic Data Change Since service will respond with HTTP Code 200 OK in a successful response.

Response parameters

Name	Description
reviewDate	**Type:** alphanumeric, ISO 8601. Epoch time if JSON **Example (JSON):** "1549537950898" **Example (XML):** "2019-02-07T11:12:30"
lwin	LWIN11 code of product **Type:** alphanumeric **Example:** "12345672008"
publication	The name of the publication. **Type:** alphanumeric **Example:** "Vinous"
reviewer	The name of the reviewer **Type:** alphanumeric **Example:** "Neal Martin"
scoreRaw	The score as published **Type:** alphanumeric **Example:** "93-96" "17++"
scoreFrom	The lower range of the score. If rawScore is not a range the actual value will be stated **Type:** alphanumeric **Example**: "93"
scoreTo	The upper range of the score. If rawScore is not a range the actual value will be stated **Type:** alphanumeric **Example**: "96"

scoreMedian	The median value of the score if a range. If rawScore is not a range the actual value will be stated **Type:** alphanumeric **Example**: "94.5"
drinkFrom	The lower range of the drinking window published (if available) **Type:** alphanumeric **Example:** "2015"
drinkTo	The upper range of the drinking window published (if available) **Type:** alphanumeric **Example:** "2020
tastingNote	The reviewer's review **Type:** alphanumeric
externalReference	The name of the article in which the review featured **Type:** alphanumeric
externalLink	HTTP link to the review on the publication's website **Type:** alphanumeric
externalId	3rd party reference id for the score and tasting note **Type:** alphanumeric

JSON Response

The response is sent per request.

```
{
    "status": "OK",
    "httpCode": "200",
    "message": "Request completed successfully",
    "internalErrorCode": "R001",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1613489323805,
        "provider": "Liv-ex"
    },
    "pageInfo": {
        "totalResults": 4,
        "limit": 50,
        "offset": 1
    },
    "criticDataChangeSince": [
        {
            "reviewDate": 1613488655000,
            "lwin": "10495252016",
            "publication": "JRT",
            "reviewer": "Test",
            "scoreRaw": "94",
            "scoreFrom": "94.0",
            "scoreTo": "94.0",
            "scoreMedian": "94.0",
            "drinkFrom": "2022",
            "drinkTo": "2050",
            "tastingNote": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
            "externalReference": "Test",
            "externalLink": "https://vinous.com/wines/",
            "externalId": "475844"
        },
        {
            "reviewDate": 1613481024000,
            "lwin": "13131522010",
            "publication": "Test",
            "reviewer": "Test 2",
            "scoreRaw": "97",
            "scoreFrom": "97.0",
            "scoreTo": "97.0",
            "scoreMedian": "97.0",
            "drinkFrom": "2026",
            "drinkTo": "2038",
            "tastingNote": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
            "externalReference": "Test",
            "externalLink": "https://vinous.com/wines/",
            "externalId": "480775"
        },
        {
            "reviewDate": 1613479595000,
            "lwin": "12229002019",
            "publication": "Vinous",
            "reviewer": "Neal Martin",
            "scoreRaw": "(89-91)",
            "scoreFrom": "89.0",
            "scoreTo": "91.0",
            "scoreMedian": "90.0",
            "drinkFrom": "2023",
            "drinkTo": "2030",
            "tastingNote": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
            "externalReference": "Test",
            "externalLink": "https://vinous.com/wines/",
            "externalId": "498716"
        },
        {
            "reviewDate": 1612876224000,
            "lwin": "13131522010",
            "publication": "Test",
            "reviewer": "Test",
            "scoreRaw": "97",
            "scoreFrom": "97.0",
            "scoreTo": "97.0",
            "scoreMedian": "97.0",
            "drinkFrom": "2026",
            "drinkTo": "2038",
            "tastingNote": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
            "externalReference": "Test",
            "externalLink": "https://vinous.com/wines/",
            "externalId": "480775"
        }
    ],
    "errors": null
}
```

Invalid JSON response

```
{
    "status": "Bad Request",
    "httpCode": "400",
    "message": "Request was unsuccessful",
    "internalErrorCode": "R000",
    "apiInfo": {
        "version": "1.0",
        "timestamp": 1613489745517,
        "provider": "Liv-ex"
    },
    "pageInfo": null,
    "criticDataChangeSince": null,
    "errors": {
        "error": [
            {
                "code": "V164",
                "message": "Wrong changeSince format. Requested date should be a valid 
date in 'YYYY-MM-DD HH:mm' format."
            }
        ]
    }
}
```

XML Response

The response is sent per request.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<criticDataChangeSinceResponse>
    <Status>OK</Status>
    <HttpCode>200</HttpCode>
    <Message>Request completed successfully</Message>
    <InternalErrorCode>R001</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2021-02-16T15:36:41.969Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <pageInfo>
        <totalResults>1</totalResults>
        <limit>50</limit>
        <offset>1</offset>
    </pageInfo>
    <criticDataChangeSince>
        <criticDataChangeSince>
            <reviewDate>2021-02-16T15:17:35Z</reviewDate>
            <lwin>10495252016</lwin>
            <publication>JRT</publication>
            <reviewer>Test</reviewer>
            <scoreRaw>94</scoreRaw>
            <scoreFrom>94.0</scoreFrom>
            <scoreTo>94.0</scoreTo>
            <scoreMedian>94.0</scoreMedian>
            <drinkFrom>2022</drinkFrom>
            <drinkTo>2050</drinkTo>
            <tastingNote>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do 
eiusmod tempor incididunt ut labore et dolore magna aliqua.</tastingNote>
            <externalReference>Test</externalReference>
            <externalLink>https://vinous.com/wines/</externalLink>
            <externalId>475844</externalId>
        </criticDataChangeSince>
    </criticDataChangeSince>
    <errors xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</criticDataChangeSinceResponse>
```

Invalid XML Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<criticDataChangeSinceResponse>
    <Status>Bad Request</Status>
    <HttpCode>400</HttpCode>
    <Message>Request was unsuccessful</Message>
    <InternalErrorCode>R000</InternalErrorCode>
    <ApiInfo>
        <Version>1.0</Version>
        <Timestamp>2021-02-16T15:38:28.901Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
    <criticDataChangeSince xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
    <errors>
        <error>
            <code>V140</code>
            <message>You do not have permission to access data from Vinous. Please
contact your account manager.</message>
        </error>
    </errors>
</criticDataChangeSinceResponse>
```

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
V000	("Mandatory field missing")
V035	("No records found")
V139	("Our records show your subscription to [publication_name] has ended. Please contact the publication and/or your account manager.")
V140	("You do not have permission to access data from [publication_name]. Please contact your account manager.")
V141	("Invalid / incorrect publication: [<value>].")
V142	("Invalid / incorrect reviewer: [<value>].")
V143	("Invalid / incorrect includeHistoric: [<value>]. Possible values are 'true' or 'false'.")

V144	("Invalid / incorrect publication and reviewer combination.")
V164	("Wrong changeSince format. Requested date should be a valid date in 'YYYY-MM-DD HH:mm' format.")
V166	("Invalid / incorrect timeframe: %s. Possible values are '1day', '1week', '2week', '1month', '3month'.")

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.