Line Manager API v1

Document Revision 1.0
Date of Issue: 05/10/2023

Fred Haselton
Business Analyst

Table of Contents

1. Purpose

To provide the API end point information and examples of the Line Manager API v1. The web service handles POST, PATCH and DELETE methods to facilitate the creation of new lines onto a list, editing of existing lines, or deleting of existing lines.

2. Glossary of Terms

Term	Meaning
listID	Identifier for each individual product list uploaded to the Liv-ex system, created via the List Manager POST API
lineID	Identifier for each individual line within the list, created via the Line Manager POST API
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 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:
    - POST for create operation
    - PATCH for edit operation
    - DELETE for delete operation
- 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 DELETE
- Authentication mechanism will be custom based on CLIENT_KEY and CLIENT_SECRET.
- For any PUSH services we require a direct POST URL which should be backed by

as service capable of accepting and process XML payload as POST request.

- 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.

Name	Mandatory	Description
CLIENT_KEY	Y	A valid user GUID which will be unique for each merchant
CLIENT_SECRET	Y	Password/Secret for the user CLIENT_KEY
ACCEPT	Y	Accept header is a way for the user 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	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 (JSON)

```
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": 1550676412005,
        "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-uat-
api.liv-ex.com/v1 https://aby-uat-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>2019-02-20T15:28:48.623Z</Timestamp>
        <Provider>Liv-ex</Provider>
    </ApiInfo>
</Response>
```

5. API Listing

5.1 Add New Line Service – POST method

Description

The service will be used to append new line information into a list, based on the input listID. A successful POST request will be responded with a lineID value that should be recorded. This lineID can then be used in edit (PATCH) and delete (DELETE) requests to manipulate the line accordingly.

Base URI

listAnalysis/v1/lineManager

Request parameters

When making a request, it is mandatory to provide a listID. In addition, a minimum of inputLwin or a combination of lxHeader Wine and lxHeader Vintage is required for a successful request.

Name	Mandatory	Description
listID	Y	Unique identifying key attached to each list, created using the Line Manager API POST method Type: 128-bit hexadecimal
lxHeader	N	Liv-ex header names

		Available headers: Producer, Wine, Country, Region, Sub region, Colour, Designation, Classification, Vintage, Quantity, Pack size, Bottle size, Price per pack, Price per bottle, Cost price per bottle, Cost price per pack, LWIN. Type: alphanumeric array
userHeader	N	Your field/column name. Type: alphanumeric Character limit 50
value	N	Field value At least one value per line should be provided Type: alphanumeric Character limit 500
inputLwin	N	LWIN7/11/16/18 value Type: alphanumeric
yourProductID	N	Your product ID Type: alphanumeric

Response parameters

Name	Description
listID	List ID provided in the API request Type: 128-bit hexadecimal
lineID	Line ID generated by the system Type: 128-bit hexadecimal
lxHeader	Liv-ex headers provided in the API request Type: alphanumeric
userHeader	Customer's field/column name provided in the request Type: alphanumeric Character limit 50
value	Field value provided in the request Type: alphanumeric
inputLwin	input LWIN (blank/7/11/16/18) Type: alphanumeric

matchedLwin	The LWIN matched to the database based on the inputLwin given Type: alphanumeric
lwinName	The name of the associated matchedLwin Type: alphanumeric
yourProductID	Your product ID Type: alphanumeric

Sample Request Body

JSON Request


```
{
  "lineManager": {
    "listID": "cd347e26-33dc-4fb2-849c-767d27d85895",
    "createLineRequest": [
      {
        "inputLwin": "10118722010",
        "yourProductID": "SKU123456",
        "customLineData": [
          {
            "lxHeader": ["vintage"],
            "userHeader": "vintage",
            "value": "2010"
          },
          {
            "lxHeader": ["wine"],
            "userHeader": "wine",
            "value": "chateaux lafite roth"
          }
        ]
      },
      {
        "inputLwin": "1234567",
        "yourProductID": "SKU123456",
        "customLineData": [
          {
            "lxHeader": ["vintage"],
            "userHeader": "vintage",
            "value": "2020"
          },
          {
```

```
                "lxHeader": ["wine"],
                "userHeader": "wine",
                "value": "chateaux latour"
            }
        ]
    }
  ]
 }
}
```


XML Request


```
<root>
  <lineManager>
    <listID>6f6f6b22-df99-4218-8d57-1f44b658633a</listID>
    <createLineRequest>
      <lineDetails>
        <inputLwin>10118722010</inputLwin>
        <yourProductID>SKU123456</yourProductID>
        <customLineData>
          <customLineDetails>
            <lxHeader>vintage</lxHeader>
            <userHeader>vintage</userHeader>
            <value>2010</value>
          </customLineDetails>
          <customLineDetails>
            <lxHeader>wine</lxHeader>
            <userHeader>wine</userHeader>
            <value>chateaux lafite roth</value>
          </customLineDetails>
        </customLineData>
      </lineDetails>
      <lineDetails>
        <inputLwin>1234567</inputLwin>
        <yourProductID>SKU123456</yourProductID>
        <customLineData>
          <customLineDetails>
            <lxHeader>vintage</lxHeader>
            <userHeader>vintage</userHeader>
            <value>2020</value>
          </customLineDetails>
          <customLineDetails>
            <lxHeader>wine</lxHeader>
```

```
                <userHeader>wine</userHeader>
                <value>chateaux latour</value>
            </customLineDetails>
        </customLineData>
      </lineDetails>
    </createLineRequest>
  </lineManager>
</root>
```


Sample Response Body

JSON response


```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1690285419547,
    "provider": "Liv-ex"
  },
  "lineManagerResponse": {
    "listID": "cd347e26-33dc-4fb2-849c-767d27d85895",
    "lineDetails": [
      {
        "lineID": "77b00bb6-9cce-4729-a064-11ade162c816",
        "inputLwin": 10118722010,
        "matchedLwin": 10118722010,
        "yourProductID": "SKU123456",
        "lwinName": "Chateau Lafite Rothschild Premier Cru Classe, Pauillac",
        "customLineData": [
          {
            "lxHeader": [
              "vintage"
            ],
            "userHeader": "vintage",
            "value": "2010"
          },
          {
            "lxHeader": [
```

```
                "wine"
              ],
              "userHeader": "wine",
              "value": "chateaux lafite roth"
            }
          ]
        },
        {
          "lineID": "53689f77-b0f9-41ee-9808-ca0ba9380477",
          "inputLwin": 1234567,
          "matchedLwin": null,
          "yourProductID": "SKU123456",
          "lwinName": null,
          "customLineData": [
            {
              "lxHeader": [
                "vintage"
              ],
              "userHeader": "vintage",
              "value": "2020"
            },
            {
              "lxHeader": [
                "wine"
              ],
              "userHeader": "wine",
              "value": "chateaux latour"
            }
          ]
        }
      ]
    },
    "errors": null
}
```


XML Response


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<root>
  <Status>OK</Status>
  <HttpCode>200</HttpCode>
  <Message>Request completed successfully</Message>
  <InternalErrorCode>R001</InternalErrorCode>
```

```
  <ApiInfo>
    <Version>1.0</Version>
    <Timestamp>2023-07-25T14:19:53.610Z</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <lineManagerResponse>
    <lineDetails>
      <customLineData>
        <lxHeader>vintage</lxHeader>
        <userHeader>vintage</userHeader>
        <value>2010</value>
      </customLineData>
      <customLineData>
        <lxHeader>wine</lxHeader>
        <userHeader>wine</userHeader>
        <value>chateaux lafite roth</value>
      </customLineData>
      <inputLwin>10118722010</inputLwin>
      <lineID>6338ebee-1cb1-4676-8b77-267eafe05bc4</lineID>
      <lwinName>Chateau Lafite Rothschild Premier Cru Classe, Pauillac</lwinName>
      <matchedLwin>10118722010</matchedLwin>
      <yourProductID>SKU123456</yourProductID>
    </lineDetails>
    <lineDetails>
      <customLineData>
        <lxHeader>vintage</lxHeader>
        <userHeader>vintage</userHeader>
        <value>2020</value>
      </customLineData>
      <customLineData>
        <lxHeader>wine</lxHeader>
        <userHeader>wine</userHeader>
        <value>chateaux latour</value>
      </customLineData>
      <inputLwin>1234567</inputLwin>
      <lineID>b5f2dfbc-4c10-4c1c-aa0e-f888ce3c0367</lineID>
      <yourProductID>SKU123456</yourProductID>
    </lineDetails>
    <listID>6f6f6b22-df99-4218-8d57-1f44b658633a</listID>
  </lineManagerResponse>
</root>
```

5.2 Edit Existing Line Service – PATCH method

Description

This service will be used to amend the editable metadata of a line (input LWIN, custom column values, product ID's). A successful PATCH response will return with the updated metadata of the line. Any live lines that belong to your company can be edited.

Base URI

listAnalysis/v1/lineManager

Request parameters

Name	Mandatory	Description
lineID	Y	Line ID (assigned via Line Manager POST call) Type: alphanumeric
userHeader	N	Reference for the columnName to be updated Type: alphanumeric
value	N	New field value At least one value per line should be provided when userHeader is specified Type: alphanumeric Character limit 500
inputLwin	N	Updated input LWIN (blank/7/11/16/18) Type: integer
yourProductID	N	Your product ID Type: alphanumeric

Response Parameter

Name	Description
listID	Unique identifying key attached to each list, created using the Line Manager API POST method

	Type: 128-bit hexadecimal
lineID	Line ID (assigned via Line Manager POST call) Type: 128-bit hexadecimal
userHeader	Reference for the columnName to be updated Type: alphanumeric
value	New field value Type: alphanumeric Character limit 500
inputLwin	Type: alphanumeric
matchedLwin	Type: alphanumeric
yourProductID	Your product ID Type: alphanumeric
lineStatus	Type: alphanumeric

Sample Request Body

JSON request


```
{
  "lineManager" : {
  "editLineRequest":
     {
     "lineID":"77b00bb6-9cce-4729-a064-11ade162c816",
     "inputLwin":"100002720101200750",
      "yourProductID":"1",
      "customLineData":
        [
          {
           "userHeader":"wine",
           "value":"chateu lafite 2010"
          },
          {
           "userHeader":"vintage",
           "value":"2010"
          }
        ]
     }
   }
}
```

XML Request


```
<root>
  <lineManager>
    <editLineRequest>
      <lineID>77b00bb6-9cce-4729-a064-11ade162c816</lineID>
      <inputLwin>100002720101200750</inputLwin>
      <yourProductID>1</yourProductID>
      <customLineData>
        <customLineDetails>
          <userHeader>wine</userHeader>
          <value>chateu lafite 2010</value>
        </customLineDetails>
        <customLineDetails>
          <userHeader>vintage</userHeader>
          <value>2017</value>
        </customLineDetails>
      </customLineData>
    </editLineRequest>
  </lineManager>
</root>
```


JSON Response


```
{
  "status": "OK",
  "httpCode": "200",
  "message": "Request completed successfully",
  "internalErrorCode": "R001",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1690367541643,
    "provider": "Liv-ex"
  },
  "lineManagerResponse": {
    "listID": "cd347e26-33dc-4fb2-849c-767d27d85895",
    "lineID": "77b00bb6-9cce-4729-a064-11ade162c816",
    "inputLwin": "100002720101200750",
    "matchedLwin": "100002720101200750",
    "yourProductID": "1",
    "lineStatus": "live",
    "customLineData": [
      {
```

```
            "userHeader": "vintage",
            "value": "2010"
          },
          {
            "userHeader": "wine",
            "value": "chateu lafite 2010"
          }
        ]
    },
    "errors": null
}
```


XML Response


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<root>
  <Status>OK</Status>
  <HttpCode>200</HttpCode>
  <Message>Request completed successfully</Message>
  <InternalErrorCode>R001</InternalErrorCode>
  <ApiInfo>
    <Version>1.0</Version>
    <Timestamp>2023-07-26T10:35:32.772Z</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <lineManagerResponse>
    <listID>cd347e26-33dc-4fb2-849c-767d27d85895</listID>
    <lineID>77b00bb6-9cce-4729-a064-11ade162c816</lineID>
    <inputLwin>100002720101200750</inputLwin>
    <matchedLwin>100002720101200750</matchedLwin>
    <yourProductID>1</yourProductID>
    <lineStatus>live</lineStatus>
    <customLineData>
      <userHeader>vintage</userHeader>
      <value>2010</value>
    </customLineData>
    <customLineData>
      <userHeader>wine</userHeader>
      <value>chateu lafite 2010</value>
    </customLineData>
  </lineManagerResponse>
</root>
```

5.3 Delete Existing Line Service – DELETE method

Description

The service will be used to delete lineID's from a list. In the response, the API should return a confirmation of both the input lineID('s) and confirmation that It has been deleted successfully. It is only possible to delete lines that belong to the account calling the API.

Base URI

listAnalysis/v1/lineManager

Request parameters

Name	Mandatory	Description
lineID	Y	Line ID (assigned via Line Manager POST call) Type: array

Response parameters

Name	Mandatory	Description
lineID	Y	Line ID (assigned via Line Manager POST call) Type: array

Sample Request Body

JSON response

```
{
  "lineManager": {
    "lineID": [
      "022f1405-2c28-42de-a23a-2f730b3f0c18",
      "022f1405-2c28-42de-a23a-2f730b3f0c19",
      "022f1405-2c28-42de-a23a-2f730b3f0c20"
    ]
  }
}
```

XML Request

```
<root>
  <lineManager>
    <lineID>022f1405-2c28-42de-a23a-2f730b3f0c18</lineID>
    <lineID>022f1405-2c28-42de-a23a-2f730b3f0c19</lineID>
    <lineID>022f1405-2c28-42de-a23a-2f730b3f0c20</lineID>
```

```
  </lineManager>
</root>
```


Sample Response Body

JSON response


```
{
  "status": "Multi-Status",
  "httpCode": "207",
  "message": "Few requests were unsuccessful",
  "internalErrorCode": "R002",
  "apiInfo": {
    "version": "1.0",
    "timestamp": 1690283762160,
    "provider": "Liv-ex"
  },
  "lineManagerDeleteResponse": {
    "lineDetails": [
      {
        "lineID": "022f1405-2c28-42de-a23a-2f730b3f0c19",
        "errors": {
          "error": [
            {
              "code": "V185",
              "message": "Invalid / incorrect lineID: [022f1405-2c28-42de-a23a-
2f730b3f0c19]. Please provide a valid lineID value."
            }
          ]
        }
      },
      {
        "lineID": "022f1405-2c28-42de-a23a-2f730b3f0c18",
        "errors": {
          "error": []
        }
      },
      {
        "lineID": "022f1405-2c28-42de-a23a-2f730b3f0c20",
        "errors": {
          "error": [
            {
```

```
                "code": "V185",
                "message": "Invalid / incorrect lineID: [022f1405-2c28-42de-a23a-
2f730b3f0c20]. Please provide a valid lineID value."
            }
        ]
      }
    }
  ]
},
"errors": null
}
```


XML response


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<root>
  <Status>Multi-Status</Status>
  <HttpCode>207</HttpCode>
  <Message>Few requests were unsuccessful</Message>
  <InternalErrorCode>R002</InternalErrorCode>
  <ApiInfo>
    <Version>1.0</Version>
    <Timestamp>2023-07-25T11:24:18.495Z</Timestamp>
    <Provider>Liv-ex</Provider>
  </ApiInfo>
  <lineManagerDeleteResponse>
    <lineDetails>
      <errors>
        <error>
          <code>V185</code>
          <message>Invalid / incorrect lineID: [022f1405-2c28-42de-a23a-
2f730b3f0c19]. Please provide a valid lineID value.</message>
        </error>
      </errors>
      <lineID>022f1405-2c28-42de-a23a-2f730b3f0c19</lineID>
    </lineDetails>
    <lineDetails>
      <errors/>
      <lineID>022f1405-2c28-42de-a23a-2f730b3f0c18</lineID>
    </lineDetails>
    <lineDetails>
      <errors>
        <error>
```

```
            <code>V185</code>
            <message>Invalid / incorrect lineID: [022f1405-2c28-42de-a23a-
2f730b3f0c20]. Please provide a valid lineID value.</message>
          </error>
        </errors>
        <lineID>022f1405-2c28-42de-a23a-2f730b3f0c20</lineID>
      </lineDetails>
    </lineManagerDeleteResponse>
</root>
```


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	Trigger
V018	Mandatory field missing(%s).	Validation trigger when attempting to pass a request without the mandatory field(s) being supplied.
V174	Invalid/incorrect listID: [%]. Please provide a valid listID value.	When a listID is supplied while using the PATCH or DELETE service that does not exist, does not match the listIDs associated with the calling account or has a status other than live.
V181	Invalid/incorrect lxHeader: [%]. Please provide a valid lxHeader value.	When an incorrect lxHeader value is supplied. Correct values can be sourced by calling the Headers GET service.
V182	Either provide with an inputLwin value or add values with lxHeaders = “Wine”,”Vintage” assigned.	When the following are true: No valid inputLwin is provided. No values have been submitted that are associated with lxHeader ‘wine’ & lxHeader ‘vintage’.
V183	Invalid userHeader submission [%]. userHeader values must be unique.	When a line is submitted with a duplicated userHeader parameter.
V184	Incompatible lxHeader mapping provided. Please use the same mapping as the original API call.	When either are true:

		When a secondary or subsequent API call is made and the relationships are different to the relationships specified in the first API call. When a first call of the API provides inconsistent mappings of lxHeaders to userHeader across different lines.
V185	Invalid/incorrect lineID: [%]. Please provide a valid lineID value.	When a lineID is submitted that is invalid or does not exist.
V186	Provided data, inputLwin, and yourProductID can not be blank. Please supply at least one parameter to be edited.	When none of the non-mandatory parameters are requested in the PATCH response.
V187	Invalid userHeader submission [%]. userHeader not found within list.	When a userHeader is submitted in a PATCH request that is not present in the custom list's mapping
V188	Invalid / incorrect inputLwin: [%].Please provide a valid LWIN7, LWIN11, LWIN16 or LWIN18 code or leave blank to remove current inputLwin.	When user supplies an LWIN code which is not blank, 7,11,16 or 18 digits in length, or when pack size is 00. When bottle size is 00000.

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.