Student Learning & Licensure API Instructions

Table of Contents

Basics

The application programming interface (API) allows developers at member organizations to send data that is needed by SL&L directly from their local systems. The data will be used to add user accounts, manage things like courses and assignments, and bring in user profile information. This data helps programs power assessment and analyze the results that SL&L helps to measure.

The API can reduce the need for manual adding of information or the use of spreadsheets to get the needed data into SL&L, thus eliminating possible errors, decreasing manual tasks, and giving the user ability to determine how frequently data is updated in the SL&L platform.

An organization's API Developers can write applications to request that SL&L make a change to the data (such as add, edit, or delete) using defined methods and parameters listed in this document, and SL&L will respond with the result. The changes that are allowed and use cases for why an organization may want to use the API to make those changes are also included in this document.

The developer can choose whatever language they want to use to connect with SL&L. However, Watermark has code available for Perl and Ruby as a starting point.

Things a SL&L API Developer Should Know

The SL&L API developer must be familiar with authenticating requests of an API using OAuth signatures and HMAC-SHA1. This developer must also be able to create a mechanism for extracting data from the local system(s), transform it to the required, pre-defined format, and transmit it using the API information in this document. If your IT team doesn't have someone who can do this, you will use our manual data import to feed information into SL&L.

The transmission process will involve creating a secure connection between the local systems and SL&L. This will require familiarity with authenticating requests of an API using OAuth signatures and HMAC-SHA1.

This mechanism should be run on a regular basis, so the developer will need to consider how they will access local resources capable of connecting to the SL&L API. This may involve scheduling the task or responding to local user operations and generating the API request, depending on the use case.

Testing and Go Live

This documentation will list endpoints and URLs associated with a non-production environment, https://sll-testing.watermarkinsights.com. When appropriate, production endpoints and URLs will be provided by the implementation team.

The test instance is not connected to a production database, so API developers can use it to test scripts and make sample requests of the API that will not affect production data. The non-production environment is good for testing

scripts or creating and modifying sample records to confirm results in the interface, but none of the changes are permanent or will inform production data in any way.

Data rate limits and processing capabilities in testing are limited compared to the production environment, so it is not recommended for large import tests or load testing. Request rate limits, the maximum number of records per request, and other questions about capacity can be discussed with the implementation technical team.

Getting Started

To help you get started, your Implementation Team will:

- Discuss the planned usages by the colleges, departments and programs and present recommendations on the information to be included in the API integration
- Provide a non-production SL&L environment for testing
- Demonstrate SL&L functions and explain data dependencies and requirements

Use Cases

The SL&L API offers endpoints to allow administrators to push data into the application from a variety of sources and use it to inform and power the experiences of their users. These endpoints let administrators create accounts, manage courses and groups, update enrollments, and populate user profile information for reporting. Below are descriptions of some common use cases and what API interactions are required to support them, so API Developers know where to start when coding for colleges, departments, or programs who will be using SL&L.

User Creation

Using the API, member accounts are mass created in SL&L based on information from some campus information system (e.g. Student Information System, Active Directory, or other databases) in a batch operation by sending an array of records.

Relevant endpoint: POST /api/memberships/users

Some administrators want user accounts created ad hoc. Using the API, a post creates a single account in response to the submission of a local form, an order completed by the university, or another system event such as workshop registration or program enrollment.

Relevant endpoint: POST /api/memberships/users

Profile Updates

Add student data like Major, Academic Program, and test scores stored in other systems to the user profile information in SL&L to be used to disaggregate and aggregate assessment and standards-based reports.

Relevant endpoint: POST /api/memberships/users

Hierarchy definition from an external system

Use data from other systems that contain organizational charts or hierarchy structures to inform structure in SL&L.

Relevant endpoint: POST /api/memberships/hierarchy

User Academic Data/Term Based Data from SIS Term Creation

Create new terms or update existing terms that can then be aligned with groups or used to add academic data for students.

Relevant endpoints: POST /api/memberships/terms

Course Shells synced from LMS

When new courses are added and records for them are created in the SIS or the LMS, add them as a “Group Template” to SL&L at the appropriate level in the hierarchy. This will allow the course to have SL&L measurement tools available and ensure that college, department or program administration have correct administrator access to manage their use.

Relevant endpoints:

- *POST /api/memberships/hierarchy*
- *POST /api/memberships/group_templates*

Courses, Internships or Cohort Groups from SIS or other source data

Feed course sections, internships or cohort groups to SL&L from other campus systems, like the SIS, LMS or Internship or Cohort management systems. Instances of these courses or groups are connected to the group template to apply pre-set system permissions and resources.

Relevant endpoints:

- *POST /api/memberships/group_templates*
- *POST /api/memberships/groups*

Roster Management and Enrollment Data from SIS

Add or drop users from groups using API posts.

Relevant endpoints:

- *POST /api/memberships/users*
- *POST /api/memberships/groups*
- *POST /api/memberships/group_memberships*

User Academic Data/Term Based Data from SIS

Add student academic data like Cumulative GPA, Term GPA, Attending Status, Primary Major, Primary Major Concentration, Additional Major, Additional Major Concentration, Minor, and Earned Credit Hours stored in other systems to the user profile information in SL&L to be used to disaggregate and aggregate assessment and standards-based reports for specific terms.

Relevant endpoints:

POST /api/memberships/academic_data

NOTE: Relevant users and terms need to exist in the system to import academic data.

Relevant endpoints to add users and terms are:

POST /api/memberships/users POST /api/memberships/terms

Authorization of Requests Using OAuth

To ensure that the request is secure and that only the organization's own application is allowed to make changes in SL&L, requests are all signed using OAuth 1.0a as the authentication mechanism.

An article on the details of the OAuth mechanism is available here:

https://oauth.net/core/1.0a/

The OAuth Key and OAuth secret that are generated in the SL&L admin's account are used in the signature. The OAuth key must be included in each request. Please note that these values are sensitive and should never be shared with anyone.

1. Go to https://sll-testing.watermarkinsights.com.
2. Log in with your email/password.
3. Click on **Settings**.
4. Click the **OAuth Configuration** tab.

Every OAuth parameter needs to be included in the signature. Below are the parameters and examples for authorizing a request. These example parameters should only be used to verify the developer's own code produces properly formatted strings and a correct signature. The example keys, secrets and signatures in this document will not be accepted by the SL&L application servers and are included here for illustrative purposes. The proper key, secret, and parameters must be generated by the local application for the testing or production environments.

HTTP Method	POST
URL	https://sll-testing.watermarkinsights.com/api/memberships/users
oauth_consumer_key	SBIJQWSNRTNATLY4RADYNRCDNLE
oauth_nonce	sDULoQDmaw

oauth_signature_method	HMAC-SHA1
oauth_timestamp	1475077240
oauth_version	1.0a

Create the Base String

The Signature Base String is a concatenation of the HTTP method, base URL (which is discussed later in this guide), and parameter strings into a single string. To do this:

1. Convert the HTTP method (post) to uppercase.
2. Append the '&' character to the string.
3. Percent-encode the URL and append it to the string.
4. Append the '&' character to the string.
5. Sort parameters by name and percent-encode the parameter string and append it to the string.

For example, when creating the base string for a user creation request with the OAuth parameters above, it will produce the following:

```
POST&https%3A%2F%2Fsll-testing.watermarkinsights.com%2Fapi%2Fmemberships%2Fuser
s&oauth_consumer_key%3DSBIJQWSNRTNATLY4RADYNRCDNLE%26oauth_nonce%3DsD
ULoQDmaw%26oauth_signature_method%3DHMAC-SHA1%26oauth_timestamp%3D147
5077240%26oauth_version%3D1.0a
```

The request will include JSON as the body, but that data should not be included in the base string.

Calculate the Signing Key

The base string and signing key are fed into the HMAC-SHA1 algorithm to generate the signature, which is discussed in the next section.

Since SL&L does not use access tokens, the signing key will only consist of the percent-encoded consumer secret, which is generated on the same page in SL&L as the Consumer Key. Use that percent-encoded string, followed by an ampersand character (&):

```
O25WUE8REKJPSVU8WMNRXMAVGYHWX1LQ7TMVDB_A-WXUNL2E9NKP8Q&
```

It is very important to keep this value private to your application. If you think your values have been compromised, regenerate your tokens. To regenerate your tokens, go to your **OAuth Configuration settings** page, click **Edit**, and then click **Regenerate Configuration**.

The example above is a simple string for illustrative purposes, but always percent-encode and append the ampersand character in the local application as the secret that SL&L generates will be unique and may include characters that require encoding.

Calculate the Signature

The signature is calculated by passing the Signature Base String and Signing Key to the HMAC-SHA1 hashing algorithm. There are implementations of HMAC-SHA1 available for every popular language. It is highly recommended to use an existing library to handle the hashing even if the code to write the signature is created from scratch.

An example that uses an existing library in Ruby is shown below.

```
require 'base64'
require 'cgi'
require 'openssl'

 def sign(key, base_string)
    digest = OpenSSL::Digest.new('sha1')
    hmac = OpenSSL::HMAC.digest(digest, key, base_string)
    Base64.encode64(hmac).chomp.gsub(/\n/, '')
  end
```

The algorithm is explained in depth here:

https://en.wikipedia.org/wiki/Hash-based_message_authentication_code

The output of the HMAC signing function is a binary string. This needs to be base64 encoded to produce the signature string. That value, when converted to base64, is the OAuth signature for this request. For example, when creating the signature for a user creation request with the OAuth parameters above, it will produce the following:

```
OAuth Signature: Gtm%2B8MA216hb8AdKrCGikSTeOf4%3D
```

The resulting request header, when put together, appears below. Note that all of the parameters (except the secret) are included in the request header and the calculated signature is sent.

```
 /api/memberships/users HTTP/1.1
 sll-testing.watermarkinsights.com
nt-Type: application/json rization:
OAuth
oauth_consumer_key="SBIJQWSNRTNATLY4RADYNRCDNLE",oauth_signature_method="H
MAC-SHA1",oauth_timestamp="1475077240",oauth_nonce="sDULoQDmaw",oauth_vers
ion="1.0a",oauth_signature="Gtm%2B8MA216hb8AdKrCGikSTeOf4%3D"
```

Endpoints, the Request Body, Import Details and Responses

Each import has a unique URL and endpoint that needs to be used when making a request. The URL used in the base string will need to include the endpoint.

The base URL in this document refers to a non-production test instance, as mentioned previously.

https://sll-testing.watermarkinsights.com

This URL should be used for testing. The production host URL will have been provided to you at the outset of implementation. The application domain is the same (https://sll.watermarkinsights.com) but the subdomain value will be different.

Endpoints

Usage	HTTP Method	Endpoint
User Creation and Profile Updates	POST	/api/memberships/users
Create University / Organizational Hierarchy	POST	/api/memberships/hierarchy
Add New Group Templates / Course Shells	POST	/api/memberships/group_templates
Create a Group Instance / Course Section	POST	/api/memberships/groups
Manage Group Membership / Enrollments	POST	/api/memberships/group_memberships
Manage Term Creation	POST	/api/memberships/terms
Manage User Academic Data	POST	api/memberships/academic_data

The Request Body

The Request Body contains the data import fields. This data must be in JSON format. It can include multiple items as an array. Here is an example of the Request Body, with the required parameters for a single user import:

```
[{
      "user_id": "9976550",
      "first_name": "Joe",
      "middle_name": "A",
      "last_name": "Studyman",
      "gender": "male",
      "email":
       "joe.studyman@livetext.
      com", "role": "student",
      "password": "password"
            }
}]
```

Import Details and Dependencies

Each of the imports has documents which cover the required and optional fields, acceptable data types and character limits, and contain examples of the data SL&L can process. The requirements for the API are the same as when using the CSV import tool in the admin account or creating the items manually in the admin interface.

Refer to the following documents, which list those requirements:

User Import Spec Hierarchy Spec

Course Import Spec Course Section Spec Memberships Import Spec

Academic Import Data Spec Terms Import Spec

Complete the Request

A complete and properly formatted user creation request using the parameters and values above would look like this:

```
POST /api/memberships/users HTTP/1.1
Host: sll-testing.watermarkinsights.com
Content-Type: application/json
Authorization: OAuth
     oauth_consumer_key="SBIJQWSNRTNATLY4RADYNRCDNLE",oauth_signature_meth
     od="HMAC-SHA1",oauth_timestamp="1475077240",oauth_nonce="sDULoQDmaw",
     oauth_version="1.0a",oauth_signature="Gtm%2B8MA216hb8AdKrCGikSTeOf4%3 D"
 [{
     "user_id": "9976550",
     "first_name": "Joe",
     "middle_name": "A", "last_name":
     "Studyman", "gender": "male",
     "email": "joe.studyman@livetext.com", "role":
     "student",
     "password": "password"
 }]
```

Server Responses, Errors and Result Logs

Requests will have an immediate response from the SL&L application in the form of a server response code that has basic information about if the request was fulfilled. Further information, like the details of the changes made as a result of the request, are available by logging in to the admin account.

Successful Requests

A successful POST request will return an HTTP (status code 200 OK) and response information in JSON format. This is a response from the API that the connection was established, the request was received, and that it was processed as expected. A summary of the resulting data changes in SL&L must be accessed through the admin account interface, which is discussed later in this guide. The request above would result in an HTTP status 200 OK and the following as the body in JSON:

```
{
          "id": "5888f2d3944138179800004e8",
          "enrollment_target": "users", "created_at":
          "2017-01-25T18:47:47.885Z",
          "status": "created",
          "updated_at": "2017-01-25T18:47:47.885Z",
          "credential": {
                    "credential_key": "SBIJQWSNRTNATLY4RADYNRCDNLE"
                     },
          "import_result": {},
          "public_url":
               "https://eportfolio-showcase.s3.amazonaws.com/enrollment_l
               ogs/190dd613573824b363534cbb4627007a?AWSAccessKeyId=AKIAIS
               SETHSQKPJC3PRA\u0026Expires=1485373667\u0026Signature=xkFr
               ZnpO2N2jg5Hcc67OTSPQef8%3D",
          "type": "json"
}
```


Here are the individual response details:

FIELD	VALUE	DESCRIPTION
id	5888f2d3944138179800004e8	The unique ID of the API import request.
enrollment_targe t	users	The API import request endpoint.
created_at	2017-01-25T18:47:47.885Z	Timestamp of when the import started.
status	created	The current status of the import request. This should always be “created”.
updated_at	2017-01-25T18:47:47.885Z	Same as created_at
credential_key	SBIJQWSNRTNATLY4RADYNRCDNL E	OAuth Consumer Key used by the request.

Error Responses and Likely Causes

While not exhaustive, some common error responses and possible causes are listed below.

Server error responses are not logged in the organization's SL&L admin account because they generally represent a failure to connect to the specific organization's endpoint and cannot be associated with the application making the request. Developers should consider logging server responses in their application for debugging purposes.

Error 404 - Not Found

This is a client error (caused by a problem with the organization's application that is making the request) that is generally accompanied by a JSON response in the body which states:

```
{
    "response": "resource not found"
}
```

The most likely cause for this error is that the host URL or endpoint is incorrect. To resolve this, confirm the correct endpoint is being used from the table above and check with the Implementation Technical Team to confirm the host URL is correct for the SL&L test or production environment to which the organization's application is trying to connect.

It may also be because the consumer key is wrong. SL&L identifies which organization and what application is making the request using the consumer key. If an application sends a key that is not recognized, SL&L does not have a valid endpoint for the organization, so it will return the "resource not found" error as a response to the request.

Error 401 - Unauthorized

This is another client error, generally accompanied by a JSON response in the body that states:

```
{
    "message": "invalid signature"
}
```

This error is most common because the consumer secret is wrong. Log in to the admin account and confirm the consumer secret is accurate.

Another common reason for this error is that the base string is not calculated correctly. If this happens, when the base string and key are passed through the HMAC-SHA1 algorithm and the result is different from what SL&L calculated, the connection is rejected. The key is probably correct, otherwise, an "Error 404 - Not Found" would have been received as noted above, so the base string is probably the cause. Refer to the section earlier in this guide on calculating the base string and make sure to use an existing HMAC-SHA1 library for hashing.

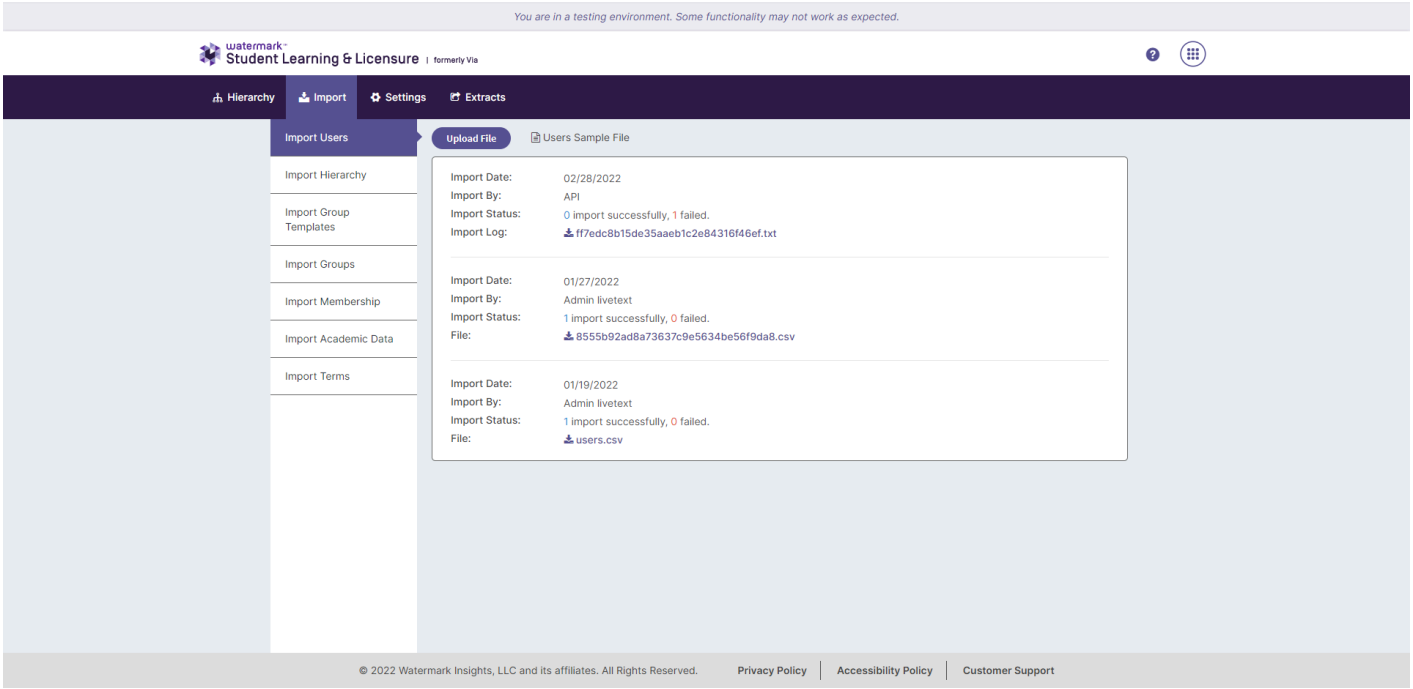
Error 500 - Internal Server Error

This is a generic server error message from the SL&L application side which indicates the problem likely exists on that end. Most likely this was a temporary condition and retrying the request will be successful. If not, please note details about the request, such as the environment (testing or production), the time and date, the endpoint, and general information about the data changes that were requested and send that to support@watermarkinsights.com so they can look into it and determine next steps.

Request Result Logs

To find the results of the API import request, log into SL&L and go to the following URL (non-production):

https://sll-testing.watermarkinsights.com/import#/import/users



The results information includes the following:

TEXT	SAMPLE VALUE	DESCRIPTION
Import Date	03/18/2018	Date of import request
Import By	API	The method used to import the data. For API requests this will be “API”. Manual imports will have the username of the individual who performed the operation

Import Status	0 import successfully, 1 failed.	This will list how many records were successfully imported and how many failed.
Data Source	link	A link to the import request data. (JSON or CSV)
Failed Information	Line Number: 1 User ID: Reasons: 1. Username: can't be blank 2. Email: is already taken 3. Password: can't be blank	This field only displays if at least one row failed. It gives relevant information about which row failed and the reason(s).

Sample Code

Watermark can provide sample code in Perl and Ruby as a starting point for development.

Making the Move to Production

When testing is complete, the developer will need to change the host URL and update the Consumer Key and Secret using values generated in the production application.

In SL&L, only the subdomain is different, and the production subdomain will have been provided to you at the outset of implementation.

Keep in mind that the calculation of the base string and the headers of the request need to include the host URL, so it is important to note that the URL subdomain is different on production from the host used in testing. When moving to production, the API developer will need to change the Host as well as the key and secret. If all three items are not updated correctly, the likely result will be a 404 error as mentioned previously in this document.

Document Version History

Version	Date	Description
1.0.2017.1	1/15/2017	Initial Version
1.1.2018.2	2/27/2018	Edited content Changed document title Fixed typos in URL, base string URL and example signature that would result in errors when deployed Updated layout, colors, brand and copyright information
2.0.2018.3	3/19/2018	Restructured and added content • Use Cases • Things an API Developer should know • Server responses, errors and result logs • Testing and Production Go Live Eliminated duplication, consolidated documents, removed redundant and confusing parameter examples and tables.
2.0.2018.4	4/20/2018	Edited content • Added external reference links for HMAC-SHA1 and OAuth 1.0a • Updated support resources • Updated Implementation Services details
2.0.2020.5	2/8/2021	Edited content • Added Term Creation/Academic Data endpoint • Updated SL&L Testing URLs
2.0.2022.6	3/1/2022	Edited content • Updated the product name
2.0.2022.7	6/16/2022	Edited content • Update the URLs from vialivetext.com to sll.watermarkinsights.com