Gooru Oneroster APIs

Overview

This document lists all the APIs related to Oneroster.

Authentication for APIs

All Oneroster APIs are authenticated based on the client id and secret shared by Gooru with partners. Partners need to base64 encode the client id and secret in order to pass as Basic authorization header to APIs.

Example: (To generate Base64 Encoded Credentials)

ClientId: `<Will be shared by Gooru team in Email>`
Secret: `<Will be shared by Gooru team in Email>`

Combine client id and secret with ':' -
`<clientId>:<Secret>`

And then encode it by Base64 encoder which will generate a string like below. Use it to pass in a request header.

```
Basic
ODU4NGIxY2QtNjEyNy00M2U1LWFlNjMtMTA3MmRmOTJhOGYzOk1sNWpKbWVrL0Y2WUpFa
HAzVXBpM0xSZ1JKaz0=
```

Data Requirements

All csv files contained in the zip should comply to the IMS Global Oneroster specification

CSV Overview:

Tenants should provide the student information to Gooru as .csv formatted files.

Tenants must continue to use .csv files to exchange roster information with Gooru, below is an outline of the format for the roster data which corresponds to the OneRoster standard.

Tenants can choose to upload class rosters by preparing four (4) files in csv format outlined in this document.

1. orgs.csv
2. users.csv
3. classes.csv
4. enrollments.csv

CSV Format:

The file format MUST be comma separated values format (CSV) for the OneRoster profile. Each field will be separated by commas and line breaks between each row. Double quotes MUST be used with a field that contains a comma.

- All files are required
- The header row is required.
- Some fields are required.
- The primary id is marked in red below, this must be unique per file.

ALL Header fields MUST be supplied in EXACTLY the same order as in the tables below. Optional fields with no data MUST simply be empty in the CSV. Header fields MUST be named the same as per the field header in the tables below. All filenames and header fields are case sensitive.

In order to roster the data into Gooru, tenants should already exist in the Gooru system against which the data will be rostered. In Case the tenant is not created, please get in touch with the Gooru Team.

Organization Rostering

First step in the rostering to create the organization based on the details provided in the "organization.csv". This file should be processed first if present

- New organization will be created if there is no organization present with the same "sourced_id" for the tenant.
- If an organization with the same "sourced_id" found for the tenant rostering the data, then the last modified date from the csv will be compared with the last modified date present in the database. If date from csv file is later than what present in date, class data will be updated otherwise no action will be taken

If this file is not present in the zip uploaded by the tenant, other csv files should contain references to existing organizations (orgSourcedId).

If there is no organization with orgSourcedId, no data will be rostered and error should be returned to the tenant.

There is one to one mapping between the organization and tenant. If any partner wants to roster the data for multiple school districts, all school districts should be first configured in Gooru as a sub tenant of the partner's tenant.

School districts will contain multiple schools having their own organization sourced ids. Data will be mapped in Gooru based on the tenant for which the data is getting rostered.

Partner's roster sync system will be responsible to roster the data against each school district separately.

Support Fields:

field header	required	format	description
sourcedId	Yes	UUID	unique id for the organization. SourcedId is used in other files and must be unique across all organizations. (For example –schools in your district).
status	No	String	"active".

dateLastModified	No	Date	The date that this record was last modified. (format is **YYYY-MM-DD**)
name	Yes	string	name of the organization
type	Yes	String	“school”
identifier	No	string	NCES ID National Center for Education Statistics) for the school / district
metadata.classification	No	String	“charter” | “private” | “public”
metadata.gender	No	String	“female” | “male” | “mixed”
metadata.boarding	No	Boolean	True if school is boarding school
parentSourcedId	No	UUID	SourcedId of the Parent organization

Class Rostering

Based on the data present in the “classes.csv” file, classes are rostered into Gooru.

- New class will be created if there is no existing class present for the organization with same “sourced_id”
- If a class with the same “sourced_id” is found for the organization, the last modified date from the csv will be compared with the last modified date present in the database. If date from csv file is later than what present in date, class data will be updated otherwise no action will be taken

Support Fields:

field header	**required**	**format**	**description**
sourcedId	Yes	UUID	Unique ID for the class. SourcedId is used in other files and must be unique across all classes.
status	No	String	“active”
dateLastModified	No	Date	The date that this record was last modified. (FORMAT IS **YYYY-MM-DD**)
title	Yes	String	Name of this class
grade	No	String	Grade (i.e. 9 or range 9-12)
courseSourcedId	No	UUID	SourcedId of the course of which this class is an instance

classCode	No	String	Human readable code used to help identify this class
classType	Yes	String	“homeroom”, “scheduled”
location	No	String	Human readable description of where the class is physically located
schoolSourcedId	Yes	UUID	SourcedId of the organization which teaches this class
termSourcedId	Yes	UUID	SourcedId of the academicSessions(s) in which the class is taught. If more than one term is needed, use double quotes and delimit with commas, (per RFC 4180) Examples: “1,2” 1 “1,4,8”
subjects	No	String	Subject name(s). If more than one subject is needed, use double quotes, and delimit with commas (per RFC 4180): Examples: “chemistry, physics” physics “music, drama, poetry”

User Rostering

Based on the data present in the “users.csv” file, users data will be rostered into the Gooru system.

- New user will be created if there is no user present for the organization with same “sourced_id” and username
- If a user with the same “sourced_id” and username / user_id found for the organization, the last modified date from the csv will be compared with the last modified date present in the database. If date from csv file is later than what present in date, user data will be updated otherwise no action will be taken

“username” is mandatory while rostering users data. This should be the login name or login id of the user in the client system.

While rostering the record into Gooru, uniqueness of the records is based on the “username” and tenant. If there is already a user with “username” (i.e. reference_id in “users” table of Core DB) for the tenant, error (409 Conflict) will be returned.

Supported Fields:

field header	required	format	description
sourcedId	Yes	UUID	Unique ID for the user. SourcedId is used in other files and must be unique across all users,
status	No	String	“active”
dateLastModified	No	Date	The date that this record was last modified. (FORMAT IS YYYY-MM-DD)
orgSourcedIds	Yes	UUID	SourcedIds of the Organizations to which this user belongs. (Note in most cases, it is expected that users will belong to a single school).
role	Yes	String	“teacher”, “student”
username	Yes	String	Username
userId	No	String	external machine readable id (e.g. LDAP id, LTI id) for this user, to be used if the sourcedId should not be used.
givenName	Yes	String	User’s first name
familyName	Yes	String	User’s surname
identifier	No	String	Identifier for the user with a human readable meaning
email	No	String	Email address for the user
sms	No	String	SMS address for the user
phone	No	String	Phone number for the user
agents	No	String	SourcedIds of the users to which this user has a relationship. Note: In most cases this will be for indicating parental relationships.

Enrollment Rostering

Based on the data present in the “enrollments.csv” file, enrollments will be rostered into the Gooru system.

- It's mandatory to pass organization sourced id, class sourced id and user sourced id in order to enroll the users. Missing any of the details will not create enrollment and specific errors should be returned.

Support Fields:

field header	required	format	description
sourcedId	Yes	UUID	Id of this enrollment
classSourcedId	Yes	UUID	Id of the class
schoolSourcedId	Yes	UUID	Id of the school
userSourcedId	Yes	UUID	Id of the user (teacher or student)
role	Yes	String	“student” | “teacher”
status	No	String	“active”
dateLastModified	No	Date	The date that this record was last modified. (FORMAT IS YYYY-MM-DD)
primary	No	Boolean	MUST only be set to true for ONE teacher for a class.

Upload Roster

This API is used to upload the roster files in zip format. Invalid or extra uploaded files are skipped from rostering. Current scope of this API is to process orgs, users, classes and enrollments only. Other files such as academicSessions, demographics and courses will not be processed and rostered into the system.

All data will be rostered against the client id (tenant) in Gooru. Client applications need to provide the same client id and secret while performing SSO which id used while rostering the users.

This API will not return any details on error / exceptions for the failed records. Status API should be explicitly called with required parameters to get the status / error while rostering files.

Details

HTTP Method	POST
End Point	http://{host}/api/nucleus-oneroster/{version}/upload
Sample	http://oneroster.gooru.org/api/nucleus-oneroster/v1/upload

Headers

Header Name	**Value**
Authorization	Basic <BASE64 ENCODED CREDENTIALS>
Content-Type	multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

- Replace the correct value of encoded credentials in 'Authorization' header
- Boundary in the 'Content-Type' header is a sample. Need to be replaced with actual boundaries while calling the API.

Request Payload

```
Attach the file with request.
```

Success Response

HTTP Status Code: 201 Created
Location header of the response will contain the Upload Id which can be used to get the status of the upload.

Error Response

In case of any error while uploading the Roster, API will not provide any details of error / exceptions for records in the files. For Error response codes, refer to HTTP Response Codes.

Get Upload Status

This API is used to get the status of the upload for a given upload id. For now this API will return total records and success record counts. It may be enhanced in future to return records level details.

Details

HTTP Method	GET
End Point	http://{host}/api/nucleus-oneroster/{version}/upload/{uploadId}/status
Sample	http://oneroster.gooru.org/api/nucleus-oneroster/v1/upload/62f3fbd3-4341-4e30-8efe-3b655074133d/status

Headers

Header Name	**Value**
Authorization	Basic <BASE64 ENCODED CREDENTIALS>

- Replace the correct value of encoded credentials in the ‘Authorization’ header.

Request Payload

No request payload required for this API

Success Response

HTTP Status Code: 200 OK

Possible Status

Status	**Purpose**
in-progress	Denotes the in progress status of the upload.
accepted	Denotes that the upload zip file has been accepted for rostering. This is the status after parsing and validation of all csv files, and still the sync with Gooru main database is pending.
failed	Denotes that the upload has been failed. In case of failure while rostering records, this status will be returned.
completed	Denotes that the upload has been complete. When all records are rostered successfully, this status will be returned.

Responses

Error Response Payload


```
{
	"status": "completed",
	"orgs_errors": [{
		"<<source_id of Org>>": "org.postgresql.util.PSQLException: ERROR:
duplicate key value violates unique constraint"
	}],
	"users_errors": [{}],
	"classes_errors": [{}],
	"enrollments_errors": [{}]
}
```


Completed Response Payload


```
{
	"status": "completed",
	"saved_records": {
		"users": 1055,
		"classes": 1280,
		"enrollments": 7701
	},
	"total_records": {
		"orgs": 0,
		"users": 0,
		"classes": 0,
		"enrollments": 0
	},
	"success_records": {
		"orgs": 0,
		"users": 0,
		"classes": 0,
		"enrollments": 0
	}
}
```


Error Response

For Error response codes, refer to HTTP Response Codes.

Reference

HTTP Response codes

For our standardized APIs we shall be using the following HTTP Status code to send back the information to the client.

HTTP Status Code	HTTP Method	Response Body/Contents	Description
200	GET	Entity which is requested	Operation successful without error
201 Created	POST	No Entity in body except for the location of entity in header	Entity creation successful
204 No Content	PUT, DELETE	No Content	Entity update or delete is successful
400 Bad Request	ALL	Error messages	Malformed, invalid or incorrect request or request parameters which are applicable to this Entity
401 Unauthorized	ALL	No Content	Action requiring authentication or session token may have expired. Note that we don't return www-authenticate headers as we do not want the browser to participate in the auth flow.
403 Forbidden	ALL	Error message	Authentication failure, or invalid API key, or insufficient privileges
404 Not Found	ALL	No Content	Entity not found

405 Not Allowed	ALL	No Content	Entity does not support the requested operation
408 Request Timeout	ALL	No Content	Request has timed out. The system was not able to complete the operation within specified time
413 Request Entity Too Large	POST, PUT	No Content	The representation of payload is too large for server to handle, or is not allowed
500 Internal Server Error	ALL	Error message	Some exceptions happened while processing the request. The message may be technical and may not be suitable for UI consumption.

Do’s and Don’ts

1. Make sure to keep a track of all sourcedIds. These sourcedIds are required when you want to enrol the new user to the old class or old user to new class.
2. CSV files should include all columns in the same sequence, including optional ones irrespective of whether there is data or not.
3. One user can be associated to one organization only.
4. Always class is associated with a teacher account.
5. If you need to associate a Student to an existing class please use the same sourcedid of that respective class.
6. Supported organization type : "school"
7. Supported Users Role : student, teacher
8. Enrollments : make sure teacher related data is available first then student.
9. Update existing Users records: there is no change in the process. But if you need to delete any account, update the status column with appropriate value and sourcedId.
10. Update existing Enrollments:
    a. If you want to add a new user to the existing class, make sure you provide the correct sourcedId in the enrollments.csv file