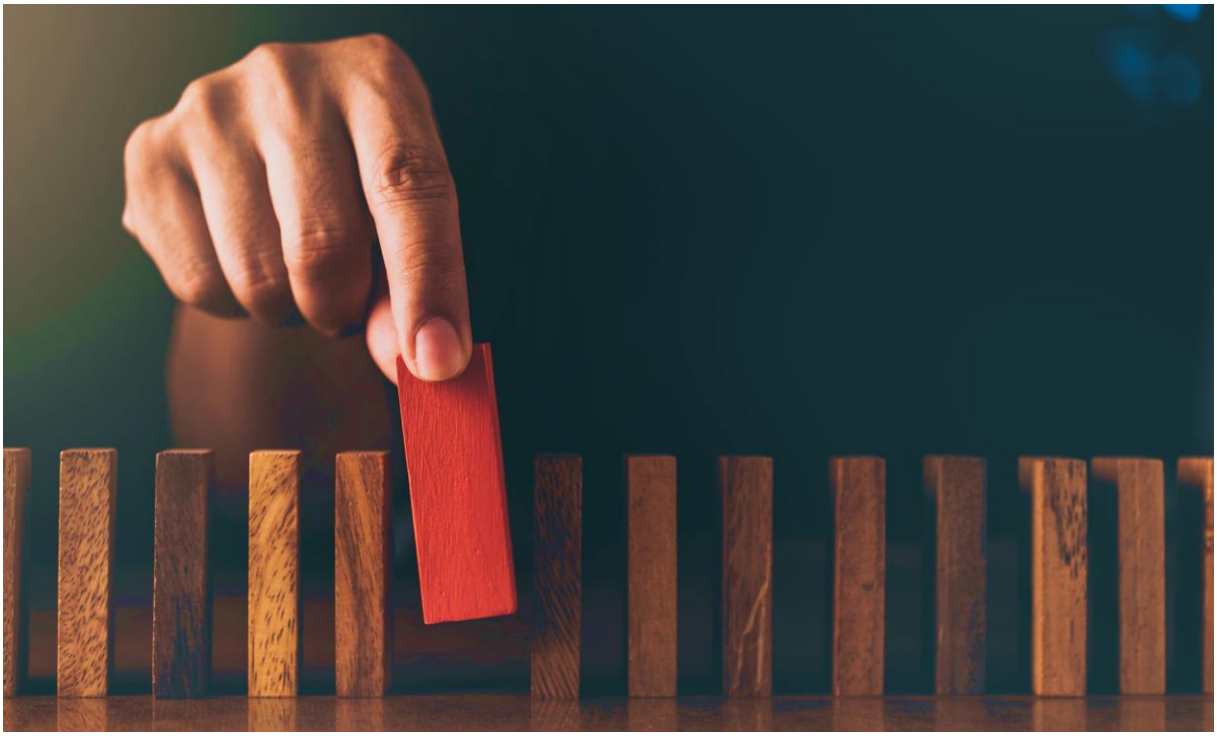
TRIBOO API DOCUMENTATION



Feb, 2022

TABLE OF CONTENT

Introduction

Triboo offers Restful APIs, allowing our clients to perform several actions regarding their platform from a third party software. These API can provide a list of accessible GET data related to courses information or users performance. But also some POST and DELETE command to automate the administration of users by third party software (register users, create users, get users…) and also a "bulk enrolment" which enrolls users to a course.

Additional Information

- The RestAPI authentication uses OAuth2.0 protocol.
- Limitations has been setup (config.) to avoid servers outage caused by multiple calls.

To Get an Access Token

To get an access token for the API, follow these steps.

1. Make sure you have the client ID and client secret strings for your API client.
2. Send a `POST` HTTP request to the `/oauth2/access_token` (try `/oauth2/v1/access_token` if doesn't work) authentication resource. Include `grant_type=client_credentials`, your client ID and client secret in the message body of your `POST` request, for example:

```
curl -XPOST "https:// <platform url>/oauth2/access_token" -d
"grant_type=client_credentials&client_id=…&client_secret=…"
```

3. Find the access token string in the `access_token` value in the JSON response data.

To Call the API Endpoints

Note that pagination can always be controlled by `page_size`, for example:

```
curl -XGET -H "Authorization: Bearer <token>" https://<platform
url>/api/extended/v1/users/?page_size=100
```

API Functions

All calls require `slug`, `apikey` and `secret`.

CREATE USERS

Create an account for one user.

Method

POST

End Point

`/api/extended/v1/users/`

Body (urlencoded)

- The profile field `org` of the created user will be filled with the `org_filter` associated to the API credentials (slug, apikey, secret).

- `platform_role` defines the user role in the platform. The possible values are:

 - `Learner` (default)

 - `Studio Admin`: the user will be added to the group “Studio Admin” (`STUDIO_ADMIN_ACCESS_GROUP`)

- `Platform Admin`: the user will created with `is_staff=True`

- `Super Platform Admin`: the user will created with `is_staff=True` and `is_superuser=True`

- `analytics_access` defines the user's access to Triboo analytics. The possible values are:

- `None` (default)

- `Restricted`: the user will be added to the group "Restricted Triboo Analytics Admin" (`ANALYTICS_LIMITED_ACCESS_GROUP`).

- `Full Access`: if the user isn't Platform Admin or Platform Super Admin, the user will be added to the group "Triboo Analytics Admin" (`ANALYTICS_ACCESS_GROUP`).

- `internal_catalog_access`: if `false`, the user will be added to the group "Catalog Denied Users" (CATALOG_DENIED_GROUP).

- `edflex_catalog_access`: if `false`, the user will be added to the group "EdFlex Denied Users" (`EDFLEX_DENIED_GROUP`).

- `crehana_catalog_access`: if `false`, the user will be added to the group "Crehana Denied Users" (`CREHANA_DENIED_GROUP`).

- `anderspink_catalog_access`: if `false`, the user will be added to the group "Anderspink Denied Users" (`ANDERSPINK_DENIED_GROUP`).

- `learnlight_catalog_access`: if `false`, the user will be added to the group "Learnlight Denied Users" (`LEARNLIGHT_DENIED_GROUP`).

Example:

```
{
    "username": "user8",
    "email": "user8@example.com",
    "first_name": "first8",
    "last_name": "last8",
    "name": "Eight"
}
```

Returns

A `status` code:

- `user_created` in case of success
- `username_already_used` if an account with this same username already exists
- `email_already_used` if an account with the same email already exists
- `error` if any other error

Response Example:


```
{
"username": "user8",
"last_name": "last8",
"lt_exempt_status": true,
"lt_comments": null,
"lt_address": null,
"city": null,
"first_name": "first8",
"lt_ilt_supervisor": null,
"lt_sub_area": null,
"lt_custom_country": null,
"location": "",
"email": "user8@example.com",
"status": "user_created",
"bio": null,
"lt_address_2": null,
"lt_is_tos_agreed": false,
"lt_phone_number": null,
"language": "",
"year_of_birth": null,
"lt_area": null,
"goals": null,
"lt_employee_id": null,
"lt_learning_group": null,
"lt_level": null,
"lt_hire_date": null,
"lt_job_code": null,
"name": "Eight",
"lt_company": null,
"lt_supervisor": null,
"gender": null,
"lt_gdpr": false,
"level_of_education": null,
"country": "",
"lt_department": null,
"lt_job_description": null
}
```

UPDATE USER

Update one user account.

Method

PUT

End Points

`/update_user/<user_id>`

`/update_user_by_username/<username>`

Body (urlencoded)

Same fields as **Create User** but the `username` and `email` are optional here. If a different `username` is sent in the body, we'll try to update it.

```
{
"anderspink_catalog_access": true
}
```

Returns

A `status` code:

- `user_updated` in case of success
- `user_not_found` if the user with this `username` or `user_id` doesn't exist
- `user_inactive` if the user with this username or user_id is inactive (and the user isn't updated)
- `username_already_used` if an account with the new `username` set in the body is already used by another user

- `email_already_used` if an account with the new `email` set in the body is already used by another user
- `error` if any other error

GET USERS

Retrieve basic information about a given user or a given list of users.

Method

GET

End Points

`/api/extended/v1/users/`

`/api/extended/v1/users/<user_id>/`

`/api/extended/v1/users/?user_id=<user_id1,user_id2,…>`

`/api/extended/v1/users_by_username/<username>/`

`/api/extended/v1/users_by_username/?username=<username1,username2,…>`

Returns

For each user requested, it will return all fields available in **Create Users** plus:

- `user_id`
- `is_active`

Example:


```
{
    "email": "user7@example.com",
    "username": "user7",
    "first_name": "first7",
    "last_name": "last7",
    "language": "",
    "location": "",
    "year_of_birth": null,
    "bio": null,
    "goals": null,
    "level_of_education": null,
    "name": "",
    "gender": null,
    "city": null,
    "country": "",
    "lt_custom_country": null,
    "lt_area": null,
    "lt_sub_area": null,
    "lt_address": null,
    "lt_address_2": null,
    "lt_phone_number": null,
    "lt_gdpr": false,
    "lt_company": null,
    "lt_employee_id": null,
    "lt_hire_date": null,
    "lt_level": null,
    "lt_job_code": null,
    "lt_job_description": null,
    "lt_department": null,
    "lt_supervisor": null,
    "lt_learning_group": null,
    "lt_exempt_status": true,
    "lt_is_tos_agreed": false,
    "lt_comments": null,
    "lt_ilt_supervisor": null,
    "analytics_access": null,
    "internal_catalog_access": false,
    "edflex_catalog_access": false,
    "crehana_catalog_access": false,
    "anderspink_catalog_access": false,
    "learnlight_catalog_access": false,
    "user_id": 30,
    "is_active": true
}
```



```
{
    "count": 2,
```

```
    "num_pages": 1,
    "current_page": 1,
    "results": [
        {
            "email": "user6@example.com",
            "username": "user6",
            "first_name": "first6",
            "last_name": "last6",
            "language": "",
            "location": "",
            "year_of_birth": null,
            "bio": null,
            "goals": null,
            "level_of_education": null,
            "name": "",
            "gender": null,
            "city": null,
            "country": "",
            "lt_custom_country": null,
            "lt_area": null,
            "lt_sub_area": null,
            "lt_address": null,
            "lt_address_2": null,
            "lt_phone_number": null,
            "lt_gdpr": false,
            "lt_company": null,
            "lt_employee_id": null,
            "lt_hire_date": null,
            "lt_level": null,
            "lt_job_code": null,
            "lt_job_description": null,
            "lt_department": null,
            "lt_supervisor": null,
            "lt_learning_group": null,
            "lt_exempt_status": true,
            "lt_is_tos_agreed": false,
            "lt_comments": null,
            "lt_ilt_supervisor": null,
            "analytics_access": null,
            "internal_catalog_access": true,
            "edflex_catalog_access": false,
            "crehana_catalog_access": false,
            "anderspink_catalog_access": false,
            "learnlight_catalog_access": false,
            "user_id": 29,
            "is_active": true
        },
        {
            "email": "user7@example.com",
            "username": "user7",
            "first_name": "first7",
            "last_name": "last7",
            "language": "",
            "location": "",
            "year_of_birth": null,
            "bio": null,
```

```
            "goals": null,
            "level_of_education": null,
            "name": "",
            "gender": null,
            "city": null,
            "country": "",
            "lt_custom_country": null,
            "lt_area": null,
            "lt_sub_area": null,
            "lt_address": null,
            "lt_address_2": null,
            "lt_phone_number": null,
            "lt_gdpr": false,
            "lt_company": null,
            "lt_employee_id": null,
            "lt_hire_date": null,
            "lt_level": null,
            "lt_job_code": null,
            "lt_job_description": null,
            "lt_department": null,
            "lt_supervisor": null,
            "lt_learning_group": null,
            "lt_exempt_status": true,
            "lt_is_tos_agreed": false,
            "lt_comments": null,
            "lt_ilt_supervisor": null,
            "analytics_access": null,
            "internal_catalog_access": false,
            "edflex_catalog_access": false,
            "crehana_catalog_access": false,
            "anderspink_catalog_access": false,
            "learnlight_catalog_access": false,
            "user_id": 30,
            "is_active": true
        }
    ],
    "next": null,
    "start": 0,
    "previous": null
}
```


DEACTIVATE USER

Set as inactive the account of a user. This won't delete any data but only deactivate the account (`is_active` set to False). The user won't be able to login.

Method

DELETE

End Points

`/api/extended/v1/users/<user_id>/`

`/api/extended/v1/users/?user_id=<user_id1,user_id2,…>`

`/api/extended/v1/users_by_username/<username>/`

`/api/extended/v1/users_by_username/?username=<username1,username2,…>`

Returns

For each user requested, it will return:

- `username`
- `user_id`
- `status` code:
    - `user_deactivated` in case of success
    - `user_not_found` if the user with this `username` or `user_id` doesn't exist
    - `user_already_inactive` if the user was already inactive
    - `error` if any other error

Example:

```
{
    "username": "user6",
    "status": "user_deactivated",
    "user_id": 29
}
```

```
[
```

```
    {
        "username": "user6",
        "status": "user_deactivated",
        "user_id": 29
    },
    {
        "username": "user7",
        "status": "user_deactivated",
        "user_id": 30
    }
]
```


GET COURSES

Retrieve the list of all the started courses matching the `org_filter` associated to the API credentials (slug, apikey, secret) including their information.

Method

GET

End Point

`/api/extended/v1/courses/`

Returns

For each course:

- `course_id`
- `title`: the course title
- `overview_url`: URL of the course about page
- `start_date`
- `card_image_url`: URL of the course card image
- `banner_image_url`: URL of the course banner image

- `description`
- `instructors`: list of the instructor names
- `duration`
- `language`
- `category`
- `tags`
- `countries`: list of countries for programmatic enrollments (`course_country`)
- `learning_groups`: list of learning groups for programmatic enrollments (`enrollment_learning_groups`)
- `last_update`

Example:


```
{
        "count": 2,
        "num_pages": 1,
        "current_page": 1,
        "results": [{
                        "id": "course-v1:edX+DemoX+Demo_Course",
                        "display_name": "Demonstration Course",
                        "overview_url":
"<http://edx.devstack.lms:18000/courses/course-
v1:edX+DemoX+Demo_Course/about>",
                        "start": "2013-02-05T05:00:00Z",
                        "card_image_url":
"<http://edx.devstack.lms:18000/asset-
v1:edX+DemoX+Demo_Course+type@asset+block@images_course_image.jpg>",
                        "banner_image_url":
"<http://edx.devstack.lms:18000/static/images/pencils.jpg>",
                        "short_description": null,
                        "instructors": [],
                        "effort": null,
                        "language": null,
                        "course_category": null,
                        "tags": [],
                        "countries": [],
                        "learning_groups": [],
                        "modified": "2021-08-10T09:19:05.182673Z"
```

```
                },
                {
                        "id": "course-v1:edX+E2E-101+course",
                        "display_name": "E2E Test Course",
                        "overview_url":
"<http://edx.devstack.lms:18000/courses/course-v1:edX+E2E-
101+course/about>",
                        "start": "2016-01-01T00:00:00Z",
                        "card_image_url":
"<http://edx.devstack.lms:18000/static/images/pencils.jpg>",
                        "banner_image_url":
"<http://edx.devstack.lms:18000/static/images/pencils.jpg>",
                        "short_description": "",
                        "instructors": [],
                        "effort": null,
                        "language": null,
                        "course_category": null,
                        "tags": [],
                        "countries": [],
                        "learning_groups": [],
                        "modified": "2021-08-10T09:51:01.100924Z"
                }
        ],
        "next": null,
        "start": 0,
        "previous": null
}
```


GET USER PROGRESS REPORT

Returns all the courses the user(s) is (are) enrolled in with progress information. Information are returned for active enrollments of active users only.

Method

GET

End Points

`/api/extended/v1/user_progress_report/<user_id>/`

`/api/extended/v1/user_progress_report/?user_id=<user_id1,user_id2,…>/`

`/api/extended/v1/user_progress_report_by_username/<username>/`

`/api/extended/v1/user_progress_report_by_username/?username=<username1,username2,…>`

`/api/extended/v1/user_progress_report_by_email/<email>/`

`/api/extended/v1/user_progress_report_by_email/?email=<email1,email2,…>`

When `supervisor` is used, we'll return the information for all users with `lt_supervisor` set with this `supervisor`.

Returns

For each user matching the request, we'll return:

- `username`
- `user_id`
- `name`: the user's full name
- `courses`, the progress report for each enrollment:
    - `course_id` (ex: course-v1:demo+F07+2020_T2)
    - `course_title`
    - `status`: `not_started`, `in_progress`, `finished` or `failed` (see `CourseStatus`)
    - `progress` (ex: 100)
    - `current_score` (ex: 89)
    - `total_time_spent` (ex: 1403094)
    - `enrollment_date` (date format)

- `completion_date` (date format)
- `badges`, the status of the user for each course badge:
  - `badge`: the badge title as in the Triboo analytics Course Progress report
  - `score` (ex: 60)
  - `success` (true/false)
  - `success_date` (date format)

Note that these informations are based on Triboo analytics Course summary (`LearnerCourseJsonReport`) and Course Progress(`LearnerBadgeJsonReport`) reports.

Example:


```
{
        "username": "janedeer",
        "user_id": 12345,
        "name": "Jane Deer",
        "courses": [{
                "course_id": "course-v1:demo+F07+2020_T2",
                "course_title": "My First Course",
                "status": "finished",
                "progress": 100,
                "current_score": 89,
                "total_time_spent": 1403094,
                "enrollment_date": "2021-08-01 01:00:00.0000",
                "completion_date": "2021-08-08 01:00:00.0000",
                "badges": [{
                        "badge": "Section 1 - badge 1",
                        "score": 90,
                        "success": true,
                        "success_date": "2021-08-08 00:30:00.0000"
                }, {
                        "badge": "Section 2 - badge 2",
                        "score": 88,
                        "success": true,
                        "success_date": "2021-08-08 01:00:00.0000"
                }]
        }]
```

```
}
```


```
{
    "count": 2,
    "num_pages": 1,
    "current_page": 1,
    "results": [
        {
            "user_id": 2,
            "username": "edx",
            "name": "",
            "courses": []
        },
        {
            "user_id": 29,
            "username": "user6",
            "name": "",
            "courses": []
        }
    ],
    "next": null,
    "start": 0,
    "previous": null
}
```

GET COURSE PROGRESS REPORT

Returns all the users that are enrolled in a course with progress information.

Informations are returned for active enrollments of active users only.

Method

GET

End Points

`/api/extended/v1/course_progress_report/<course ID>/`

Params

- `from` (optional, **required** when to is used): in `YYYY-MM-DD` format

- `to` (optional): in `YYYY-MM-DD` format

Example Request:

GET [https://<platform url>/api/extended/v1/course_progress_report/course-v1:edX+AGT101+2020_T1/?from=2021-09-15&to=2022-01-01](https:// <platform url>/api/extended/v1/course_progress_report/course-v1:edX+AGT101+2020_T1/?from=2021-09-15&to=2022-01-01)

Returns

For each user matching the request, we'll return:

- `course_id` (ex: course-v1:demo+F07+2020_T2)

- `course_title`
- `enrollments`, the progress report for each enrollment:
    - `username`
    - `user_id`
    - `name`
    - `status`: `Not Started`, `In Progress`, `Successful` or `Unsuccessful` (see `CourseStatus`)
    - `progress` (ex: 100)
    - `current_score` (ex: 89)
    - `total_time_spent` (ex: 1403094)
    - `enrollment_date` (date format)
    - `completion_date` (date format)

Note that these informations are based on Triboo analytics Course summary (`LearnerCourseJsonReport`) report.

Example Response:


```
{
    "course_id": "course-v1:edX+AGT101+2020_T1",
    "course_title": "Global Security Awareness (English)" ,
    "enrollments": [
        {
            "username": "staf",
            "user_id": 10,
            "name": "",
            "status": "In Progress",
            "progress": 99,
            "current_score": 15,
            "total_time_spent": 352,
            "enrollment_date": "2020-04-28T03:58:04.346006Z",
            "completion_date": null
        },
        {
            "username": "LaetitiaP",
            "user_id": 14,
            "name": "Laetitia Pfaender",
            "status": "Successful",
            "progress": 48,
            "current_score": 100,
            "total_time_spent": 4082,
            "enrollment_date": "2020-05-21T03:14:40.349301Z",
            "completion_date": "2021-10-08T14:41:41.532196Z"
        },
        {
            "username": "Yu",
```

```
            "user_id": 101,
            "name": "Yu",
            "status": "In Progress",
            "progress": 13,
            "current_score": 0,
            "total_time_spent": 772,
            "enrollment_date": "2020-05-20T09:22:23.832814Z",
            "completion_date": null
        },
        {
            "username": "Aaron",
            "user_id": 102,
            "name": "Aaron Zhao",
            "status": "In Progress",
            "progress": 45,
            "current_score": 55,
            "total_time_spent": 15039,
            "enrollment_date": "2020-04-28T03:46:49.254177Z",
            "completion_date": null
        },
        {
            "username": "LT-developer",
            "user_id": 131,
            "name": "",
            "status": "In Progress",
            "progress": 15,
            "current_score": 0,
            "total_time_spent": 10,
            "enrollment_date": "2022-01-13T10:34:27.922044Z",
            "completion_date": null
        }
    ]
}
```

BULK ENROLL

This is a simple API endpoint that allows user with staff role to bulk enroll users to one or many course.

Rationale

Its utility comes from the following abilities not currently made available by the other enrollment-related endpoints:

1. Ability to supply multiple courses for enrollment per-request.
2. Ability to specify users for enrollment using emails (without usernames).
3. Ability to enroll non-existent users in courses before they register or without asking them to login first.
4. Ability to specify whether or not to email users informing them that they've been enrolled per-request.

These features together are important for implementing external integration systems that synchronize/push a high volume of enrollment changes to a Triboo instance.

What does a request to this endpoint look like?

An example request to the endpoint looks like this:

```
POST /api/bulk_enroll/v1/bulk_enroll
```

```
{
        "action": "enroll",
        "auto_enroll": true,
        "email_students": true,
        "courses": "course-v1:triboo+Demo+123",
        "identifiers": "brandon@triboolearnning.com"
}
```

These fields follow suit with the expected input of the `student_update_enrollment` view:

- `action`
  - `enroll` or `unenroll`
  - can be used to specify whether the supplied users should be enrolled or removed from the specified courses.
- `auto_enroll`
  - `true` or `false`
  - allows choosing whether users should be pre-enrolled in courses even if they haven't registered yet.
- `email_students`
  - `true` or `false`
  - allows choosing whether enrolled users should be sent enrollment confirmation emails.
- `courses` is a comma-separated list of serialized course keys.
- `identifiers` is a comma-separated list of emails.

How does it compare to `/api/enrollment/v1/enrollment`?

An example request to the endpoint looks like this:

`POST /api/enrollment/v1/enrollment`

```
{
        "mode": "honor",
        "course_details": {
                "course_id": "course-v1:triboo+Demo+123"
        },
        "is_active": true,
        "username": "bdero"
}
```

- Only one course ID can be supplied per-request.
- `username` must be the username of an existing user, it's not possible to do any of the following:
  1. Pre-enroll users that aren't yet registered.
  2. Supply multiple users.
  3. Supply an emails instead of usernames.

Testing instructions:

1. Using a Client ID/Secret associated with a staff user, fetch an OAuth2 bearer token:

```
# Get a token:
curl -d
"client_id=*****&client_secret=**********&grant_type=client_credent
ials&token_type=jwt"  -X POST  https://DOMAIN.learning-
tribes.com/oauth2/access_token/


# Response:
{"access_token": "********", "id_token":
"************************************", "expires_in": 31535999,
"token_type": "jwt", "scope": "profile openid email permissions"}
```

2. Using the token, construct an authenticated request to the bulk_enroll endpoint (replace "brandon+testenroll@triboolearning.com" with an email address you have access to):

```
# Post the enrollment:
curl -X POST --header "Authorization: JWT ***********" -H "Content-
Type: application/json" -d '{"action": "enroll","auto_enroll":
true,"email_students": true,"courses": "course-
v1:triboo+DemoX+Demo_Course","identifiers":
"brandon+testenroll@triboolearning.com"}'
<https://DOMAIN.learning-tribes.com/api/bulk_enroll/v1/bulk_enroll>

# Response:
{"detail":"Authentication credentials were not provided."}
```

Response Example:

```
{
    "action": "enroll",
    "courses": {
        "course-v1:triboo+DemoX+Demo_Course": {
            "action": "enroll",
            "results": [
                {
                    "identifier":
"brandon+testenroll@triboolearning.com",
                    "after": {
                        "enrollment": false,
```

```
                            "allowed": true,
                            "user": false,
                            "auto_enroll": true
                        },
                        "before": {
                            "enrollment": false,
                            "allowed": false,
                            "user": false,
                            "auto_enroll": false
                        }
                    }
                ],
                "auto_enroll": true
            }
        },
        "email_students": true,
        "auto_enroll": true
    }
```