# edX Studio Documentation

***Release 0.1***

**EdX Doc Team**

September 21, 2016

Contents

CHAPTER 1

# Read Me

The edX Platform API documentation is created using RST files and Sphinx. You, the user community, can help update and revise this documentation project on GitHub:

https://github.com/edx/edx-platform/tree/master/docs/en_us/platform_api/source

To suggest a revision, fork the project, make changes in your fork, and submit a pull request back to the original project: this is known as the GitHub Flow.

CHAPTER 2

# Other edX Resources

Course teams, researchers, developers, learners: the edX community includes groups with a range of reasons for using the platform and objectives to accomplish. To help members of each group learn about what edX offers, reach goals, and solve problems, edX provides a variety of information resources.

To help you find what you need, browse the edX offerings in the following categories.



All members of the edX community are encouraged to make use of any of the resources described in this preface. We welcome your feedback on these edX information resources. Contact the edX documentation team at docs@edx.org.

## 2.1 The edX Partner Portal

The edX Partner Portal is the destination for partners to learn, connect, and collaborate with one another. Partners can explore rich resources and share success stories and best practices while staying up-to-date with important news and updates.

To use the edX Partner Portal, you must register and request verification as an edX partner. If you are an edX partner and have not used the edX Partner Portal, follow these steps.

1. Visit partners.edx.org, and select **Create New Account**.
2. Select **Request Partner Access**, then fill in your personal details.
3. Select **Create New Account**. You will receive a confirmation email with your account access within 24 hours.

After you create an account, you can sign up to receive email updates about edX releases, news from the product team, and other announcements. For more information, see *Release Announcements by Email*.

### 2.1.1 Course Team Support in the edX Partner Portal

EdX partner course teams can get technical support in the edX Partner Portal. To access technical support, submit a support ticket, or review any support tickets you have created, go to partners.edx.org and select **Course Staff Support** at the top of the page. This option is available on every page in the Partner Portal.

## 2.2 The Open edX Portal

The Open edX Portal is the destination for all edX users to learn about the edX roadmap, as well as hosting, extending the edX platform, and contributing to Open edX. In addition, the Open edX Portal provides product announcements, the Open edX blog, and other rich community resources.

All users can view content on the Open edX Portal without creating an account and logging in.

To comment on blog posts or the edX roadmap, or subscribe to email updates, you must create an account and log in. If you do not have an account, follow these steps.

1. Visit open.edx.org/user/register.
2. Fill in your personal details.
3. Select **Create New Account**. You are then logged in to the Open edX Portal.

### 2.2.1 Release Announcements by Email

To receive and share product and release announcements by email, you can subscribe to announcements on one of the edX portal sites.

1. Create an account on the Open edX Portal or the edX Partner Portal as described above.
2. Select **Community** and then **Announcements**.
3. Under **Subscriptions**, select the different types of announcements that you want to receive through email. You might need to scroll down to see these options.
4. Select **Save**.

You will now receive email messages when new announcements of the types you selected are posted.

## 2.3 System Status

For system-related notifications from the edX operations team, including outages and the status of error reports. On Twitter, you can follow @edxstatus.

Current system status and the uptime percentages for edX servers, along with the Twitter feed, are published on the edX Status web page.

## 2.4 Resources for Course Teams

Course teams include faculty, instructional designers, course staff, discussion moderators, and others who contribute to the creation and delivery of courses on edx.org or edX Edge.

### 2.4.1 The edX Learning Series

The courses in the edX Learning Series provide foundational knowledge about using the edX platform. These courses are available on edx.org.

#### edX101: Overview of Creating a Course

The edX101 course is designed to provide a high-level overview of the course creation and delivery process using Studio and the edX LMS. It also highlights the extensive capabilities of the edX platform.

#### StudioX: Creating a Course with edX Studio

After you complete edX101, StudioX provides more detail about using Studio to create a course, add different types of content, and configure your course to provide an optimal on-line learning experience.

#### VideoX: Creating Video for the edX Platform

VideoX presents strategies for creating videos for course content and course marketing. The course provides step-by-step instructions for every stage of video creation, and includes links to exemplary sample videos created by edX partner institutions.

### 2.4.2 Documentation

Documentation for course teams is available on the docs.edx.org web page.

- Building and Running an edX Course is a comprehensive guide with concepts and procedures to help you build a course in edX Studio, and then use the Learning Management System (LMS) to run a course.

  When you are working in edX Studio, you can access relevant sections of this guide by selecting **Help** on any page.

- Using edX Insights describes the metrics, visualizations, and downloadable .csv files that course teams can use to gain information about student background and activity.

- The edX Release Notes summarize the changes in each new version of deployed software.

These guides open in your web browser. The left side of each page includes a **Search docs** field and links to the contents of that guide. To open or save a PDF version, select **v: latest** at the lower right of the page, then select **PDF**.

---

**Note:** If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

---

### 2.4.3 Email

To receive and share information by email, course team members can:

- Subscribe to announcements and other new topics in the edX Partner Portal or the Open edX Portal. For information about how to subscribe, see Release Announcements through the Open edX Portal.

- Join the openedx-studio Google group to ask questions and participate in discussions with peers at other edX partner organizations and edX staffers.

### 2.4.4 Wikis and Web Sites

The edX product team maintains public product roadmaps on *the Open edX Portal* and *the edX Partner Portal*.

The edX Partner Support site for edX partners hosts discussions that are monitored by edX staff.

## 2.5 Resources for Researchers

Data for the courses on edx.org and edX Edge is available to the "data czars" at our partner institutions, and then used by database experts, statisticians, educational investigators, and others for educational research.

### 2.5.1 Documentation

The edX Research Guide is available on the docs.edx.org web page.

This guide opens in your web browser, with a **Search docs** field and links to that guide's contents on the left side of each page. To open or save a PDF version, select **v: latest** at the lower right of the page, and then select **PDF**.

---

**Note:** If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

---

### 2.5.2 Email

To receive and share information by email, researchers can join the openedx-analytics Google group to ask questions and participate in discussions with peers at other edX partner organizations and edX staffers.

### 2.5.3 Wikis

The edX Analytics team maintains the Open edX Analytics wiki, which includes links to periodic release notes and other resources for researchers.

The edx-tools wiki lists publicly shared tools for working with the edX platform, including scripts for data analysis and reporting.

## 2.6 Resources for Developers

Software engineers, system administrators, and translators work on extending and localizing the code for the edX platform.

### 2.6.1 Documentation

Documentation for developers is available on the docs.edx.org web page.

- The edX Platform Developer's Guide includes guidelines for contributing to Open edX, options for extending the Open edX platform, using the edX public sandboxes, instrumenting analytics, and testing.
- Installing, Configuring, and Running the Open edX Platform provides procedures for getting an edX developer stack (devstack) and production stack (fullstack) operational.

- Open edX XBlock Tutorial guides developers through the process of creating an XBlock, and explains the concepts and anatomy of XBlocks.
- Open edX XBlock API Guide provides reference information about the XBlock API.
- edX Open Learning XML Guide provides guidelines for building edX courses with Open Learning XML (OLX). Note that this guide is currently an alpha version.
- edX Data Analytics API provides reference information for using the data analytics API to build applications to view and analyze learner activity in your course.
- edX Platform APIs provide reference information for building applications to view course information and videos and work with user and enrollment data.

---

**Note:** If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

---

### 2.6.2 GitHub

These are the main edX repositories on GitHub.

- The edx/edx-platform repo contains the code for the edX platform.
- The edx/edx-analytics-dashboard repo contains the code for edX Insights.
- The edx/configuration repo contains scripts to set up and operate the edX platform.

Additional repositories are used for other projects. Our contributor agreement, contributor guidelines and coding conventions, and other resources are available in these repositories.

### 2.6.3 Community Discussions

The Community Discussions page in the Open edX Portal lists different ways that you can ask, and answer, questions.

### 2.6.4 Wikis and Web Sites

The Open edX Portal is the entry point for new contributors.

The edX Engineering team maintains an open Confluence wiki, which provides insights into the plans, projects, and questions that the edX Open Source team is working on with the community.

The edx-tools wiki lists publicly shared tools for working with the edX platform, including scripts and helper utilities.

## 2.7 Resources for Open edX

Hosting providers, platform extenders, core contributors, and course staff all use Open edX. EdX provides release-specific documentation, as well as the latest version of all guides, for Open edX users. The following documentation is available.

- Open edX Release Notes provides information on the contents of Open edX releases.
- Building and Running an Open edX Course is a comprehensive guide with concepts and procedures to help you build a course in Studio, and then use the Learning Management System (LMS) to run a course.

  When you are working in Studio, you can access relevant sections of this guide by selecting **Help** on any page.

- Open edX Learner's Guide helps students use the Open edX LMS to take courses. This guide is available on the docs.edx.org web page. Because learners are currently only guided to this resource through the course, we encourage course teams to provide learners with links to this guide as needed in course updates or discussions.
- Installing, Configuring, and Running the Open edX Platform provides information about installing and using devstack and fullstack.
- The edX Platform Developer's Guide includes guidelines for contributing to Open edX, options for extending the Open edX platform, using the edX public sandboxes, instrumenting analytics, and testing.
- Open edX XBlock Tutorial guides developers through the process of creating an XBlock, and explains the concepts and anatomy of XBlocks.
- Open edX XBlock API Guide provides reference information on the XBlock API.
- EdX Open Learning XML Guide provides guidelines for building edX courses with Open Learning XML (OLX). Note that this guide is currently an alpha version.
- EdX Data Analytics API provides reference information for using the data analytics API to build applications to view and analyze learner activity in your course.
- EdX Platform APIs provide reference information for building applications to view course information and videos and work with user and enrollment data.

---

**Note:** If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

---

## 2.8 Resources for Learners

### 2.8.1 Documentation

The EdX Learner's Guide and the Open edX Learner's Guide are available on the docs.edx.org web page. Because learners are currently only guided to this resource through the course, we encourage course teams to provide learners with links to these guides as needed in course updates or discussions.

### 2.8.2 In a Course

All edX courses have a discussion forum where you can ask questions and interact with other students and with the course team: select **Discussion**. Many courses also offer a wiki for additional resources and materials: select **Wiki**.

Other resources might also be available, such as a course-specific Facebook page or Twitter feed. Be sure to check the **Home** page for your course as well as the **Discussion** and **Wiki** pages.

From time to time, the course team might send email messages to all students. While you can opt out of these messages, doing so means that you can miss important or time-sensitive information. To change your preferences for course email, select **edX** or **edX edge** at the top of any page. On your dashboard of current courses, locate the course and then select **Email Settings**.

### 2.8.3 From edX

To help you get started with the edX learning experience, edX offers a course (of course!). You can find the edX Demo course on the edX web site. EdX also maintains a list of frequently asked questions and answers.

If you still have questions or suggestions, you can get help from the edX support team: select **Contact** at the bottom of any edX web page or send an email message to info@edx.org.

For opportunities to meet others who are interested in edX courses, check the edX Global Community meetup group.

CHAPTER 3

# Change Log

## 3.1 2015

| Date | Change |
|---|---|
| 10 June 2015 | Added documentation for the profile images API. |
| 11 May 2015 | Updated the *User API* to Version 1.0.<br><br>Added the *User API User Preferences Resource*. |
| 23 April 2015 | Updated the example responses in the profile images API documentation. |
| 2 April 2015 | Added documentation for the course structure API, *Enrollment API Version 1.0* and edX Platform User API Version 0 sections. |
| 29 January 2015 | Added documentation about changing a user's status in a course to the profile images API documentation. |

CHAPTER 4

# Overview of the edX Platform APIs

The edX Platform APIs are a rapidly growing and evolving set of capabilities that enable you to build web, desktop, and mobile applications that work with your Open edX instance.

The edX Platform APIs use REST design principles and support the JSON data- interchange format. The APIs also use edX OAuth 2.0 for *authentication*.

## 4.1 Supported edX Platform API Modules

The following edX Platform APIs are currently supported.

- *Courses API Version 1.0*
- *Enrollment API Version 1.0*
- *User API Version 1.0*

CHAPTER 5

# EdX API Authentication

## 5.1 OAuth 2.0

The edX Platform APIs use OAuth 2.0 for authentication. OAuth 2.0 is an open standard used by many systems that require secure user authentication. See the OAuth 2.0 Standard for more information.

## 5.2 Registering with Your Open edX Instance

To use the edX Platform API with courses on your instance of Open edX, you must register your application with the Open edX server. See the OAuth 2.0 specification for details.

CHAPTER 6

# Supported APIs

## 6.1 Courses API Version 1.0

### 6.1.1 Courses API Overview

Use the Courses API to obtain information about edX courses.



#### Courses API Version and Status

The Courses API is currently at version 1.0.

#### Courses API Resources and Endpoints

The Courses API includes the *Courses* and *Blocks* resources, and supports the following tasks, methods, and endpoints.

##### Courses Resource

| Task | Method | Endpoint |
|---|---|---|
| *Get a List of Courses* | GET | /api/courses/v1/courses/ |
| *Get Details for a Course* | GET | /api/courses/v1/courses/{course_key}/ |

##### Blocks Resource

| Task | Method | Endpoint |
|---|---|---|
| *Get a List of Course Blocks in a Course* | GET | /api/courses/v1/blocks/ with required `course_id` parameter |
| *Get a List of Course Blocks in a Block Tree* | GET | /api/courses/v1/blocks/{usage_id} |

### 6.1.2 Courses API Courses Resource

With the Courses API **Courses** resource, you can complete the following tasks.



#### Get a List of Courses

The endpoint to get a list of courses is `/api/courses/v1/courses/`.

##### Use Case

Get a list of the courses that are visible to a specific user. If a username is not supplied, an anonymous user is assumed. Users with course staff permissions can specify other users' usernames.

##### Request Format

GET /api/courses/v1/courses/

Example:

GET /api/courses/v1/courses/?username=anjali

##### Query Parameters

- username (optional) - The username of the user for whom the course data is being accessed. If the request is made for an anonymous user, username is not required. Only users with course staff permissions can specify other users' usernames.
- org (optional) - A code for an organization; case-insensitive. Example: "HarvardX". If `org` is specified, the list of courses includes only courses that belong to the specified organization.
- mobile (optional) - If specified, the list of courses includes only courses that are designated as `mobile_available`.

##### Response Values

The following fields are returned with a successful response. All date/time fields are in ISO 8601 notation.

- effort: A textual description of the weekly hours of effort expected in the course.
- end: The date and time that the course ends.
- enrollment_end: The date and time that enrollment ends.
- enrollment_start: The date and time that enrollment begins.
- id: A unique identifier of the course; a serialized representation of the opaque key identifying the course.
- media: An object that contains named media items, including the following objects.
  - course_image: An image to show for the course.

        * uri: The location of the image.
    - course_video: A video about the course.
        * uri: The location of the video.
- name: The name of the course.
- number: The catalog number of the course.
- org: The name of the organization that owns the course.
- overview: An HTML textual description of the course.
- short_description: A textual description of the course.
- start: The date and time that the course begins.
- start_display: The start date of the course, formatted for reading.
- start_type: Indicates how `start_display` was set. Possible values are:
    - "string": Manually set by the course author.
    - "timestamp": Generated from the `start` timestamp.
    - "empty": No start date is specified.
- pacing: The type of pacing for this course. Possible values are `instructor` and `self`.
- course_id: The course key. This field might be returned but is deprecated. You should use `id` instead.

#### Example Response Showing The List of Courses Visible to a User


```
{

 "media": {
    "course_image": {
        "uri": "/c4x/edX/example/asset/just_a_test.jpg",
        "name": "Course Image"
    }
  },
  "description": "An example course.",
  "end": "2015-09-19T18:00:00Z",
  "enrollment_end": "2015-07-15T00:00:00Z",
  "enrollment_start": "2015-06-15T00:00:00Z",
  "id": "edX/example/2012_Fall",
  "name": "Example Course",
  "number": "example",
  "org": "edX",
  "start": "2015-07-17T12:00:00Z",
  "start_display": "July 17, 2015",
  "start_type": "timestamp"
}
```


### Get Details for a Course

The endpoint to get the details for a course is `/api/courses/v1/courses/{course_key}/`.

#### Use Case

Get the details for a course that you specify using a course key.

#### Request Format

GET /api/courses/v1/courses/{course_key}

Example: GET /api/courses/v1/courses/edX%2FDemoX%2FDemo_Course

#### Query Parameters

- username (optional) - The username of the user for whom the course data is being accessed. If the request is made for an anonymous user, username is not required. Only users with course staff permissions can specify other users' usernames.

#### Response Values

*Response values* for this endpoint are the same as those for *Get a List of Courses*.

#### Example Response Showing Details of a Specified Course

The following example response is returned from this request:

GET /api/courses/v1/courses/edX%2FDemoX%2FDemo_Course


```
{

  "effort": null,
  "end": "2015-08-08T00:00:00Z",
  "enrollment_start": "2015-01-01T00:00:00Z",
  "enrollment_end": "2015-05-01T00:00:00Z",
  "id": "edX/DemoX/Demo_Course",
  "media": {
      "course_image":    {
          "uri": "/c4x/edX/DemoX/asset/images_course_image.jpg"
          },
      "course_video": {
          "uri": null
          },
   },
   "name": "edX Demonstration Course",
   "number": "DemoX",
   "org": "edX",
   "short_description": null,
   "start": "2015-02-05T05:00:00Z",
   "start_display": "Feb. 5, 2015",
   "start_type": "timestamp",
   "pacing": "instructor",
   "course_id": "edX/DemoX/Demo_Course",
   "overview": "<p>Include your long course description here.</p>"
}
```

### 6.1.3 Courses API Blocks Resource

With the Courses API **Blocks** resource, you can complete the following tasks.



#### Get a List of Course Blocks in a Course

The endpoint to get a list of course blocks in a course is `/api/courses/v1/blocks/` with required `course_id` parameter.

##### Use Case

Get a list of course blocks in a specified course. Response results depend on the viewing user's permissions level within a course, as well as group membership and individual allowances (such as due date extensions), if any.

##### Request Format

GET /api/courses/v1/blocks/?course_id=<course_id>

Example:

GET /api/courses/v1/blocks/?course_id=edX%2FDemoX%2FDemo_Course &all_blocks=true&requested_fields=graded,format,studen

##### Query Parameters

- all_blocks: (boolean) Provide a value of `true` to return all blocks, including those that are visible only to specific learners (for example, based on group or cohort membership, or randomized content from content libraries). Returns all blocks only if the requesting user has course staff permissions. If `all_blocks` is not specified, you must specify the username for the user whose course blocks are requested.
- block_counts: (list) Specify the types of blocks for which to return a count. Example: `block_counts=video,problem`.
- block_types_filter: (list) Specify the types of blocks to be included in results. Values are the names of any XBlock type in the system, for example: `sequential`, `vertical`, `html`, `problem`, `video` or `discussion`. Example: `block_types_filter=problem,html`.
- course_id: (string, required) The URL-encoded ID of the course whose block data you are requesting. Example: `course_id=edX%2FDemoX%2FDemo_Course`.
- depth: (integer or `all`) Specify how far in the course blocks hierarchy to traverse down. A value of `all` specifies the entire hierarchy. The default value is `0`. Example: `depth=all`.
- requested_fields: (list) Specify the fields to return for each block, in addition to `id`, `type`, and `display_name`, which are always returned. For the list of possible fields, see the fields listed under `blocks` in the *Response Values* section. Example: `requested_fields=graded,format,student_view_multi_device`.
- return_type: (string) Specify the data type in which the block data is returned. Supported values are `dict` and `list`. The default value is `dict`.

- student_view_data: (list) Specify the types of blocks for which to return the `student_view_data` response value, which consists of a JSON representation of the block's data. Example: `student_view_data=video`.
- username: (string) Required, unless `all_blocks` is specified. Specify the username for the user whose course blocks are requested. Only users with course staff permissions can specify other users' usernames. If a username is specified, results include blocks that are visible to that user, including those based on group or cohort membership or randomized content assigned to that user. Example: `username=anjali`.

#### Response Values

The following fields are returned with a successful response.

- root: The ID of the root node of the requested course block structure.
- blocks: A dictionary or list, based on the value of the `return_type` query parameter. Maps block usage IDs to a collection of information about each block, as described in the following fields.
  - id: (string) The usage ID of the block.
  - type: (string) The type of block. Values are the names of any XBlock type in the system, including custom blocks. Examples are: `course`, `chapter`, `sequential`, `vertical`, `html`, `problem`, `video`, and `discussion`.
  - display_name: (string) The display name of the block.
  - children: (list) If the block has child blocks, an ordered list of IDs of the child blocks. Returned only if `children` is included in the `requested_fields` query parameter.
  - block_counts: (dict) For each block type specified in the `block_counts` query parameter, the aggregate number of blocks of that type within the root block and all of its descendants. For example, if you specify `block_counts=video,problem` as a query parameter, in the `block_counts` response value the number of video blocks and problem blocks in the specified block and in its children, is returned.
  - graded: (boolean) Whether or not the block or any of its descendants is graded. Returned only if `graded` is included in the `requested_fields` query parameter.
  - format: (string) The assignment type of the block. Possible values can be `Homework`, `Lab`, `Midterm Exam`, and `Final Exam`. Returned only if `format` is included in the `requested_fields` query parameter.
  - student_view_data: (dict) The JSON data for this block, if the specified block type implements the `student_view_data` method. The JSON data can be used to natively render the XBlock. Returned only if the `student_view_data` query parameter contains this block's type. See also `student_view_multi_device` and `student_view_url`.
  - student_view_multi_device: (boolean) This value indicates whether or not the HTML of the student view that is rendered at the `student_view_url` supports responsive web layouts, touch-based inputs, and interactive state management for a variety of device sizes and types, including mobile and touch devices. Returned only if `student_view_multi_device` is included in the `requested_fields` query parameter.
  - student_view_url: (string) The URL to retrieve the HTML rendering of the block's student view. The HTML can include CSS and Javascript code. This field can be used in combination with the `student_view_multi_device` field to determine whether a block can be viewed on a device. This URL can be used as a fallback if the `student_view_data` for this block type is not supported by the client or by the block.

- lms_web_url: (string) The URL to the navigational container of the XBlock on the web LMS. This URL can be used as a fallback if the `student_view_data` and `student_view_url` fields are not supported.
- lti_url: (string) The block URL for an LTI consumer. Returned only if the `ENABLE_LTI_PROVIDER` Django setting is set to `True`.

#### Example Response Showing a List of Course Blocks in a Specified Course

The following example response is returned from this request:

GET /api/courses/v1/blocks/?course_id=edX/DemoX/Demo_Course&all_blocks=true&block_counts=video,html,problem&requested_fields=graded,format,student_view_data,student_view_url,student_view_multi_device&student_view_data=video,html,problem

Only the top level block in the course is returned, because the `depth` parameter was not specified.

```
{
    "root": "i4x://edX/DemoX/course/Demo_Course",
    "blocks": {
        "i4x://edX/DemoX/course/Demo_Course": {
            "display_name": "edX Demonstration Course",
            "graded": false,
            "student_view_url": "https://courses.edx.org/xblock/i4x://edX/DemoX/
             course/Demo_Course",
            "student_view_multi_device": false,
            "lms_web_url": "https://courses.edx.org/courses/edX/DemoX/Demo_Course/
             jump_to/i4x://edX/DemoX/ course/Demo_Course",
            "type": "course",
            "id": "i4x://edX/DemoX/course/Demo_Course",
            "block_counts": {
                "problem": 23,
                "html": 32,
                "video": 5
            }
        }
    }
}
```

### Get a List of Course Blocks in a Block Tree

The endpoint to get a list of course blocks in a specified block tree is `/api/courses/v1/blocks/{usage_id}/`.

#### Use Case

Get a list of course blocks in a specified block tree. Response results depend on the specified user's permissions level within a course, as well as group membership and individual allowances (such as due date extensions), if any.

#### Request Format

GET /api/courses/v1/blocks/{usage_id}/

Example:

GET /api/courses/v1/blocks/i4x%3A%2F%2FedX%2FDemoX%2Fvertical %2F2152d4a4aadc4cb0af5256394a3d1fc7?all_blocks=true

#### Query Parameters

*Query parameters* for this endpoint are the same as those for *Get a List of Course Blocks in a Course*.

#### Response Values

*Response values* for this endpoint are the same as those for *Get a List of Course Blocks in a Course*.

#### Example Response Showing a List of Course Blocks in a Block Tree

The following example response is returned from this request:

GET /api/courses/v1/blocks/i4x%3A%2F%2FedX%2FDemoX%2Fvertical %2F2152d4a4aadc4cb0af5256394a3d1fc7?all_blocks=true


```
{
    "root": "i4x://edX/DemoX/vertical/2152d4a4aadc4cb0af5256394a3d1fc7",
    "blocks": {
        "i4x://edX/DemoX/discussion/e5eac7e1a5a24f5fa7ed77bb6d136591": {
            "display_name": "",
            "lms_web_url": "https://courses.edx.org/courses/edX/DemoX/Demo_Course/
             jump_to/i4x://edX/DemoX/discussion/e5eac7e1a5a24f5fa7ed77bb6d136591",
            "type": "discussion",
            "id": "i4x://edX/DemoX/discussion/e5eac7e1a5a24f5fa7ed77bb6d136591",
            "student_view_url": "https://courses.edx.org/xblock/i4x://edX/DemoX/
             discussion/e5eac7e1a5a24f5fa7ed77bb6d136591"
        },
        "i4x://edX/DemoX/vertical/2152d4a4aadc4cb0af5256394a3d1fc7": {
            "display_name": "Pointing on a Picture",
            "lms_web_url": "https://courses.edx.org/courses/edX/DemoX/Demo_Course/
             jump_to/i4x://edX/DemoX/vertical/2152d4a4aadc4cb0af5256394a3d1fc7",
            "type": "vertical",
            "id": "i4x://edX/DemoX/vertical/2152d4a4aadc4cb0af5256394a3d1fc7",
            "student_view_url": "https://courses.edx.org/xblock/i4x://edX/DemoX/
             vertical/2152d4a4aadc4cb0af5256394a3d1fc7"
        },
        "i4x://edX/DemoX/problem/c554538a57664fac80783b99d9d6da7c": {
            "display_name": "Pointing on a Picture",
            "lms_web_url": "https://courses.edx.org/courses/edX/DemoX/Demo_Course/
             jump_to/i4x://edX/DemoX/problem/c554538a57664fac80783b99d9d6da7c",
            "type": "problem",
            "id": "i4x://edX/DemoX/problem/c554538a57664fac80783b99d9d6da7c",
            "student_view_url": "https://courses.edx.org/xblock/i4x://edX/DemoX/
             problem/c554538a57664fac80783b99d9d6da7c"
        }
    }
}
```

## 6.2 Enrollment API Version 1.0

### 6.2.1 Enrollment API Overview

Use the Enrollment API to view user and course enrollment information and to enroll a user in a course.



#### Enrollment API Version and Status

The Enrollment API is currently at version 1.0. We plan to make significant enhancements to this API.

#### Enrollment API Endpoints

The Enrollment API supports the following tasks, methods, and endpoints.

| Task | Method | Endpoint |
|---|---|---|
| *Get the user's enrollment status in a course* | GET | /api/enrollment/v1/enrollment/{user_id},{course_id} |
| *Get the user's enrollment information for a course* | GET | /api/enrollment/v1/course/{course_id} |
| *View a user's enrollments* | GET | /api/enrollment/v1/enrollment |
| *Enroll a user in a course* | POST | /api/enrollment/v1/enrollment{"course_details":{"course_id":"{course_id}"}} |

### 6.2.2 Enrollment API Enrollment Resource

With the Enrollment API **Enrollment** resource, you can complete the following tasks.



#### Get the User's Enrollment Status in a Course

**class** `enrollment.views.`**`EnrollmentView`**(***kwargs*)

**Use Case**

Get the user's enrollment status for a course.

**Example Request**

GET /api/enrollment/v1/enrollment/{username},{course_id}

**Response Values**

If the request for information about the user is successful, an HTTP 200 "OK" response is returned.

The HTTP 200 response has the following values.

•course_details: A collection that includes the following values.

– course_end: The date and time when the course closes. If null, the course never ends.

– course_id: The unique identifier for the course.

– course_modes: An array of data about the enrollment modes supported for the course. If the request uses the parameter include_expired=1, the array also includes expired enrollment modes.

Each enrollment mode collection includes the following values.

* currency: The currency of the listed prices.

* description: A description of this mode.

* expiration_datetime: The date and time after which users cannot enroll in the course in this mode.

* min_price: The minimum price for which a user can enroll in this mode.

* name: The full name of the enrollment mode.

* slug: The short name for the enrollment mode.

* suggested_prices: A list of suggested prices for this enrollment mode.

– course_end: The date and time at which the course closes. If null, the course never ends.

– course_start: The date and time when the course opens. If null, the course opens immediately when it is created.

– enrollment_end: The date and time after which users cannot enroll for the course. If null, the enrollment period never ends.

– enrollment_start: The date and time when users can begin enrolling in the course. If null, enrollment opens immediately when the course is created.

– invite_only: A value indicating whether students must be invited to enroll in the course. Possible values are true or false.

• created: The date the user account was created.

• is_active: Whether the enrollment is currently active.

• mode: The enrollment mode of the user in this course.

• user: The ID of the user.

**Example response showing the user's enrollment status in a course**

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS

{
    "created": "2014-11-19T04:06:55Z",
    "mode": "honor",
    "is_active": true,
    "course_details": {
        "course_id": "edX/DemoX/Demo_Course",
        "enrollment_end": null,
        "course_modes": [
            {
                "slug": "honor",
                "name": "Honor Code Certificate",
```

```
                "min_price": 0,
                "suggested_prices": [],
                "currency": "usd",
                "expiration_datetime": null,
                "description": null
            }
        ],
        "enrollment_start": null,
        "invite_only": false
    },
    "user": "staff"
}
```

### Get the User's Enrollment Information for a Course

**class** enrollment.views.**EnrollmentCourseDetailView**(***kwargs*)

**Use Case**

Get enrollment details for a course.

Response values include the course schedule and enrollment modes supported by the course. Use the parameter include_expired=1 to include expired enrollment modes in the response.

**Note:** Getting enrollment details for a course does not require authentication.

**Example Requests**

GET /api/enrollment/v1/course/{course_id}

GET /api/enrollment/v1/course/{course_id}?include_expired=1

**Response Values**

If the request is successful, an HTTP 200 "OK" response is returned along with a collection of course enrollments for the user or for the newly created enrollment.

Each course enrollment contains the following values.

- course_end: The date and time when the course closes. If null, the course never ends.
- course_id: The unique identifier for the course.
- course_modes: An array of data about the enrollment modes supported for the course. If the request uses the parameter include_expired=1, the array also includes expired enrollment modes.

  Each enrollment mode collection includes the following values.

  - currency: The currency of the listed prices.
  - description: A description of this mode.
  - expiration_datetime: The date and time after which users cannot enroll in the course in this mode.
  - min_price: The minimum price for which a user can enroll in this mode.
  - name: The full name of the enrollment mode.
  - slug: The short name for the enrollment mode.
  - suggested_prices: A list of suggested prices for this enrollment mode.
- course_start: The date and time when the course opens. If null, the course opens immediately when it is created.

- enrollment_end: The date and time after which users cannot enroll for the course. If null, the enrollment period never ends.
- enrollment_start: The date and time when users can begin enrolling in the course. If null, enrollment opens immediately when the course is created.
- invite_only: A value indicating whether students must be invited to enroll in the course. Possible values are true or false.

**Example response showing a user's course enrollment information**

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS

{
    "course_id": "edX/DemoX/Demo_Course",
    "enrollment_end": null,
    "course_modes": [
        {
            "slug": "honor",
            "name": "Honor Code Certificate",
            "min_price": 0,
            "suggested_prices": [],
            "currency": "usd",
            "expiration_datetime": null,
            "description": null
        }
    ],
    "enrollment_start": null,
    "invite_only": false
}
```

### View a User's Enrollments or Enroll a User in a Course

**class** enrollment.views.**EnrollmentListView**(***kwargs*)

**Use Cases**

- Get a list of all course enrollments for the currently signed in user.
- Enroll the currently signed in user in a course.

  Currently a user can use this command only to enroll the user in the default course mode. If this is not supported for the course, the request fails and returns the available modes.

  This command can use a server-to-server call to enroll a user in other modes, such as "verified", "professional", or "credit". If the mode is not supported for the course, the request will fail and return the available modes.

  You can include other parameters as enrollment attributes for a specific course mode. For example, for credit mode, you can include the following parameters to specify the credit provider attribute.

  - namespace: credit
  - name: provider_id
  - value: institution_name

**Example Requests**

GET /api/enrollment/v1/enrollment

POST /api/enrollment/v1/enrollment {

"mode": "credit", "course_details":{"course_id": "edX/DemoX/Demo_Course"}, "enrollment_attributes":[{"namespace": "credit","name": "provider_id","value": "hogwarts",},]

}

**POST Parameters**

A POST request can include the following parameters.

- user: Optional. The username of the currently logged in user. You cannot use the command to enroll a different user.
- mode: Optional. The course mode for the enrollment. Individual users cannot upgrade their enrollment mode from the default. Only server-to-server requests can enroll with other modes.
- is_active: Optional. A Boolean value indicating whether the enrollment is active. Only server-to-server requests are allowed to deactivate an enrollment.
- course details: A collection that includes the following information.
  - course_id: The unique identifier for the course.
- email_opt_in: Optional. A Boolean value that indicates whether the user wants to receive email from the organization that runs this course.
- enrollment_attributes: A dictionary that contains the following values.
  - namespace: Namespace of the attribute
  - name: Name of the attribute
  - value: Value of the attribute
- is_active: Optional. A Boolean value that indicates whether the enrollment is active. Only server-to-server requests can deactivate an enrollment.
- mode: Optional. The course mode for the enrollment. Individual users cannot upgrade their enrollment mode from the default. Only server-to-server requests can enroll with other modes.
- user: Optional. The user ID of the currently logged in user. You cannot use the command to enroll a different user.

**GET Response Values**

If an unspecified error occurs when the user tries to obtain a learner's enrollments, the request returns an HTTP 400 "Bad Request" response.

If the user does not have permission to view enrollment data for the requested learner, the request returns an HTTP 404 "Not Found" response.

**POST Response Values**

If the user does not specify a course ID, the specified course does not exist, or the is_active status is invalid, the request returns an HTTP 400 "Bad Request" response.

If a user who is not an admin tries to upgrade a learner's course mode, the request returns an HTTP 403 "Forbidden" response.

If the specified user does not exist, the request returns an HTTP 406 "Not Acceptable" response.

**GET and POST Response Values**

If the request is successful, an HTTP 200 "OK" response is returned along with a collection of course enrollments for the user or for the newly created enrollment.

Each course enrollment contains the following values.

- course_details: A collection that includes the following values.
  - course_end: The date and time when the course closes. If null, the course never ends.
  - course_id: The unique identifier for the course.
  - course_modes: An array of data about the enrollment modes supported for the course. If the request uses the parameter include_expired=1, the array also includes expired enrollment modes.

    Each enrollment mode collection includes the following values.

    - currency: The currency of the listed prices.
    - description: A description of this mode.
    - expiration_datetime: The date and time after which users cannot enroll in the course in this mode.
    - min_price: The minimum price for which a user can enroll in this mode.
    - name: The full name of the enrollment mode.
    - slug: The short name for the enrollment mode.
    - suggested_prices: A list of suggested prices for this enrollment mode.
  - course_start: The date and time when the course opens. If null, the course opens immediately when it is created.
  - enrollment_end: The date and time after which users cannot enroll for the course. If null, the enrollment period never ends.
  - enrollment_start: The date and time when users can begin enrolling in the course. If null, enrollment opens immediately when the course is created.
  - invite_only: A value indicating whether students must be invited to enroll in the course. Possible values are true or false.
- created: The date the user account was created.
- is_active: Whether the enrollment is currently active.
- mode: The enrollment mode of the user in this course.
- user: The username of the user.

**Example response showing a user who is enrolled in two courses**

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, POST, HEAD, OPTIONS

[
    {
        "created": "2014-09-19T18:08:37Z",
        "mode": "honor",
        "is_active": true,
        "course_details": {
```

```
            "course_id": "edX/DemoX/Demo_Course",
            "enrollment_end": null,
            "course_modes": [
                {
                    "slug": "honor",
                    "name": "Honor Code Certificate",
                    "min_price": 0,
                    "suggested_prices": [],
                    "currency": "usd",
                    "expiration_datetime": null,
                    "description": null
                }
            ],
            "enrollment_start": null,
            "invite_only": false
        },
        "user": "honor"
    },
    {
        "created": "2014-09-19T18:09:35Z",
        "mode": "honor",
        "is_active": true,
        "course_details": {
            "course_id": "ArbisoftX/BulkyEmail101/2014-15",
            "enrollment_end": null,
            "course_modes": [
                {
                    "slug": "honor",
                    "name": "Honor Code Certificate",
                    "min_price": 0,
                    "suggested_prices": [],
                    "currency": "usd",
                    "expiration_datetime": null,
                    "description": null
                }
            ],
            "enrollment_start": "2014-05-01T04:00:00Z",
            "invite_only": false
        },
        "user": "honor"
    }
]
```


**Example response showing that a user has been enrolled in a new course**


```
{
    "course_details": {
        "course_id": "edX/DemoX/Demo_Course"
    }
}
```


## 6.3 User API Version 1.0

### 6.3.1 User API Overview

Use the User API to view and update account and preference information.

#### User API Version and Status

The User API is currently at version 1.0. We plan on making significant enhancements to this API.

#### User API Resources and Endpoints

The User API supports the following resources, tasks, methods, and endpoints.

##### User Accounts API Resource

| Task | Method | Endpoint |
|---|---|---|
| *Get a user's account information* | GET | /api/user/v1/accounts/{username}/[?view=shared] |
| *Update your account information* | PATCH | /api/user/v1/accounts/{username}/{"key":"value"} |

##### User Preferences API Resource

| Task | Method | Endpoint |
|---|---|---|
| *Get a user's preferences information* | GET | /api/user/v1/preferences/{username}/ |
| *Update a user's preferences information* | PATCH | /api/user/v1/preferences/{username}/ |
| *Get a specific preference* | GET | /api/user/v1/preferences/{username}/{preference_key} |
| *Update a specific preference* | PUT | /api/user/v1/preferences/{username}/{preference_key} |
| *Delete a specific preference* | DELETE | /api/user/v1/preferences/{username}/{preference_key} |

### 6.3.2 User API User Accounts Resource

With the User API **User Accounts** resource, you can complete the following tasks.



#### Get and Update a User's Account Information

**class** `user_api.accounts.views.`**`AccountViewSet`**`(`***kwargs*`)`

**Use Cases**

Get or update a user's account information. Updates are supported only through merge patch.

**Example Requests**

GET /api/user/v1/accounts?usernames={username1,username2}[?view=shared] GET /api/user/v1/accounts/{username}/[?view=shared]

PATCH /api/user/v1/accounts/{username}/{"key":"value"} "application/merge-patch+json"

**Response Values for GET**

If no user exists with the specified username, an HTTP 404 "Not Found" response is returned.

If the user makes the request for her own account, or makes a request for another account and has "is_staff" access, an HTTP 200 "OK" response is returned. The response contains the following values.

- bio: null or textual representation of user biographical information ("about me").
- country: An ISO 3166 country code or null.
- date_joined: The date the account was created, in the string format provided by datetime. For example, "2014-08-26T17:52:11Z".
- email: Email address for the user. New email addresses must be confirmed via a confirmation email, so GET does not reflect the change until the address has been confirmed.
- gender: One of the following values:
  - null
  - "f"
  - "m"
  - "o"
- goals: The textual representation of the user's goals, or null.
- is_active: Boolean representation of whether a user is active.
- language: The user's preferred language, or null.
- language_proficiencies: Array of language preferences. Each preference is a JSON object with the following keys:
  - "code": string ISO 639-1 language code e.g. "en".
- level_of_education: One of the following values:
  - "p": PhD or Doctorate
  - "m": Master's or professional degree
  - "b": Bachelor's degree
  - "a": Associate's degree
  - "hs": Secondary/high school
  - "jhs": Junior secondary/junior high/middle school
  - "el": Elementary/primary school
  - "none": None
  - "o": Other
  - null: The user did not enter a value
- mailing_address: The textual representation of the user's mailing address, or null.
- name: The full name of the user.
- profile_image: A JSON representation of a user's profile image information. This representation has the following keys.
  - "has_image": Boolean indicating whether the user has a profile image.

- "image_url_*": Absolute URL to various sizes of a user's profile image, where '*' matches a representation of the corresponding image size, such as 'small', 'medium', 'large', and 'full'. These are configurable via PROFILE_IMAGE_SIZES_MAP.

- requires_parental_consent: True if the user is a minor requiring parental consent.
- username: The username associated with the account.
- year_of_birth: The year the user was born, as an integer, or null.
- account_privacy: The user's setting for sharing her personal profile. Possible values are "all_users" or "private".
- accomplishments_shared: Signals whether badges are enabled on the platform and should be fetched.

For all text fields, plain text instead of HTML is supported. The data is stored exactly as specified. Clients must HTML escape rendered values to avoid script injections.

If a user who does not have "is_staff" access requests account information for a different user, only a subset of these fields is returned. The returns fields depend on the ACCOUNT_VISIBILITY_CONFIGURATION configuration setting and the visibility preference of the user for whom data is requested.

Note that a user can view which account fields they have shared with other users by requesting their own username and providing the "view=shared" URL parameter.

**Response Values for PATCH**

Users can only modify their own account information. If the requesting user does not have the specified username and has staff access, the request returns an HTTP 403 "Forbidden" response. If the requesting user does not have staff access, the request returns an HTTP 404 "Not Found" response to avoid revealing the existence of the account.

If no user exists with the specified username, an HTTP 404 "Not Found" response is returned.

If "application/merge-patch+json" is not the specified content type, a 415 "Unsupported Media Type" response is returned.

If validation errors prevent the update, this method returns a 400 "Bad Request" response that includes a "field_errors" field that lists all error messages.

If a failure at the time of the update prevents the update, a 400 "Bad Request" error is returned. The JSON collection contains specific errors.

If the update is successful, updated user account data is returned.

**Example response showing a user's account information**

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS, PATCH

{
  "username": "John",
  "name": "John Doe",
  "language": "",
  "gender": "m",
  "year_of_birth": 1987,
  "level_of_education": "m",
  "goals": "Professional Development",
  "country": US,
```

```
  "mailing_address": "123 Main Street, Anytown, MA 02144",
  "email": "johndoe@company.com",
  "date_joined": "2015-03-18T13:42:40Z",
  "account_privacy": "all_users"
}
```

### 6.3.3 User API User Preferences Resource

With the User API **User Preferences** resource, you can complete the following tasks.



#### Get and Update the User's Preferences Information

**class** user_api.preferences.views.**PreferencesView**(***kwargs*)

**Use Cases**

Get or update the user's preference information. Updates are only supported through merge patch. Preference values of null in a patch request are treated as requests to remove the preference.

**Example Requests**

GET /api/user/v1/preferences/{username}/

PATCH /api/user/v1/preferences/{username}/ with content_type "application/merge-patch+json"

**Response Values for GET**

If no user exists with the specified username, an HTTP 404 "Not Found" response is returned.

If a user without "is_staff" access requests preferences for a different user, an HTTP 404 "Not Found" message is returned.

If the user makes the request for her own account, or makes a request for another account and has "is_staff" access, an HTTP 200 "OK" response is returned. The response contains a JSON dictionary with a key/value pair (of type String) for each preference.

The list of preferences depends on your implementation. By default, the list includes the following preferences.

•account_privacy: The user's setting for sharing her personal profile. Possible values are "all_users" or "private".

•pref-lan: The user's preferred language, as set in account settings.

**Response Values for PATCH**

Users can only modify their own preferences. If the requesting user does not have the specified username and has staff access, the request returns an HTTP 403 "Forbidden" response. If the requesting user does not have staff access, the request returns an HTTP 404 "Not Found" response to avoid revealing the existence of the account.

If no user exists with the specified username, an HTTP 404 "Not Found" response is returned.

If "application/merge-patch+json" is not the specified content type, a 415 "Unsupported Media Type" response is returned.

If validation errors prevent the update, this method returns a 400 "Bad Request" response that includes a "field_errors" field that lists all error messages.

If a failure at the time of the update prevents the update, a 400 "Bad Request" error is returned. The JSON collection contains specific errors.

If the update is successful, an HTTP 204 "No Content" response is returned with no additional content.

**Example response showing the user's preference information**

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS, PATCH

{
  "pref-lang": "en",
  "account_privacy": "private"
}
```

### Get, Update, or Delete a Specific Preference

**class** user_api.preferences.views.**PreferencesDetailView**(***kwargs*)

**Use Cases**

Get, create, update, or delete a specific user preference.

**Example Requests**

GET /api/user/v1/preferences/{username}/{preference_key}

PUT /api/user/v1/preferences/{username}/{preference_key}

DELETE /api/user/v1/preferences/{username}/{preference_key}

**Response Values for GET**

If the specified username or preference does not exist, an HTTP 404 "Not Found" response is returned.

If a user without "is_staff" access requests preferences for a different user, a 404 error is returned.

If the user makes the request for her own account, or makes a request for another account and has "is_staff" access, an HTTP 200 "OK" response is returned that contains a JSON string.

**Response Values for PUT**

Users can only modify their own preferences. If the requesting user does not have the specified username and has staff access, the request returns an HTTP 403 "Forbidden" response. If the requesting user does not have staff access, the request returns an HTTP 404 "Not Found" response to avoid revealing the existence of the account.

If the specified preference does not exist, an HTTP 404 "Not Found" response is returned.

If the request is successful, a 204 "No Content" status is returned with no additional content.

**Response Values for DELETE**

Users can only delete their own preferences. If the requesting user does not have the specified username and has staff access, the request returns an HTTP 403 "Forbidden" response. If the requesting user does not have staff access, the request returns an HTTP 404 "Not Found" response to avoid revealing the existence of the account.

If the specified preference does not exist, an HTTP 404 “Not Found” response is returned.

If the update is successful, an HTTP 204 “No Content” response is returned with no additional content.

**Example response to a request for the user’s account_privacy setting**

```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS, PATCH

"private"
```

CHAPTER 7

# Deprecated APIs

## 7.1 Course Structure API Version 0 (Deprecated)

The course structure API version 0 is deprecated. EdX platform developers should not implement new client functions that use the course structure API version 0.

You can get information about the courses offered by an edX platform installation by using the `/api/courses/v1/courses/` REST endpoint.

You can get information about the parameters and return values of `/api/courses/v1/courses` from the Django REST framework web page for that endpoint. For example, https://courses.edx.org/api/courses/v1/courses/ provides information about the courses offered by edx.org.

**Note:** The documentation available at docs.edx.org does not include information about the `/api/courses/v1/courses/` REST endpoint. Developer documentation for the `/api/courses/v1/courses/` is planned in upcoming documentation releases.

## 7.2 Mobile API Version 0.5 (Deprecated)

The mobile API version 0.5 is deprecated. EdX platform developers should not implement new client functions that use the mobile API version 0.5.

You can get information about the courses offered by an edX platform installation by using the `/api/courses/v1/courses/` REST endpoint.

You can get information about the parameters and return values of `/api/courses/v1/courses` from the Django REST framework web page for that endpoint. For example, https://courses.edx.org/api/courses/v1/courses/ provides information about the courses offered by edx.org.

**Note:** The documentation available at docs.edx.org does not include information about the `/api/courses/v1/courses/` REST endpoint. Developer documentation for the `/api/courses/v1/courses/` is planned in upcoming documentation releases.

You can get and update information about learners by using the edX platform user API. For more information about getting and updating learner information in the user API, see *Get and Update a User's Account Information*.

## 7.3 Profile Images API Version 1.0 (Deprecated)

The profile images API version 1.0 is deprecated. EdX platform developers should not implement new client functions that use the profile images API version 1.0.

You can get and update learners' profile images by using the edX platform user API. The `profile_image` object is included in the JSON information for a learner. For more information about getting and updating profile images in the user API, see *Get and Update a User's Account Information*.

## A



## E



## P