edX Research Guide

Release

December 12, 2015

Contents

This document is intended for researchers, data czars, and administrative teams at edX partner institutions who use the edX data exports to gain insight into their courses and students.

CHAPTER 1

For Your Information

1.1 Change Log

1.1.1 October-December 2015

Date	Change
1 Dec 2015	Added new events for *discussion forum* voting to the Events in the Tracking Logs section.
	Added new events for interactions with *Office Mixes* to the Events in the Tracking Logs section.
	Updated events in the *Open Response Assessment Events* topic to reflect the addition of file types other than images.
23 Nov 2015	Updated the *Events in the Tracking Logs* section and the *Alphabetical Event List* to correct the names of several events.
10 Nov 2015	Corrected the description for the `problem_show` event in the *Problem Interaction Events* section.
4 Nov 2015	Updates made throughout this guide to replace Python terminology (dictionary, integer, float) with JSON terminology (object, number) where appropriate.
27 Oct 2015	Added new events for interactions with *Oppia explorations* to the Events in the Tracking Logs section.

1.1.2 July-September 2015

Date	Change
16 Sept 2015	Added new events for *teams* to the Events in the Tracking Logs section.
2 Sept 2015	Added new events for *digital certificates*.
6 Aug 2015	Updated the *Data Delivered in Data Packages* section to include approximate sizes for the files in data packages.
8 Jul 2015	Added new events for *polls and surveys* to the Events in the Tracking Logs section.
1 Jul 2015	Added new events for *problem hints* to the Events in the Tracking Logs section.

1.1.3 April-June 2015

Date	Change
10 Jun 2015	Added information about events for *pre-roll videos* to the *Events in the Tracking Logs* section.
8 Jun 2015	Added descriptions of the `video_show_cc_menu` and `video_hide_cc_menu` events to the *video interaction events* section.
19 May 15	Added information about new course team report events to the *Events in the Tracking Logs* section.
11 May 2015	Updated the descriptions of the `pause_video`, `play_video`, and `stop_video` *video interaction events* to include the effects of a **Video Start Time** or **Video Stop Time**.
22 Apr 2015	Added information about the new `student_languageproficiency` table and two new columns in the `auth_userprofile` table to *Student Info and Progress Data*.
6 Apr 2015	Added a section to describe the *course structure* file.

1.1.4 January-March 2015

Date	Change
18 Mar 2015	Added information about library events for students to the *Events in the Tracking Logs* section.
11 Mar 2015	Added information about additional video interaction events that are now emitted by the edX mobile app, and reorganized the *Video Interaction Events* in the Tracking Logs section.
5 Mar 2015	Added new events for contributions to discussion forums to the *Events in the Tracking Logs* section.
	Added events for the display of *Google components* to the Tracking Logs section.
3 Mar 2015	Updated the *Preface* to include information about the *The edX Partner Portal* and the *The Open edX Portal*.
23 Feb 2015	Added new common fields for HTTP header values and new events for video caption use to the *Events in the Tracking Logs* section.
13 Feb 2015	Added the `edx.course.enrollment.mode_changed` event to the *Events in the Tracking Logs* section.
4 Feb 2015	Added information about the `module.usage_key` member field in the common `context` field to the *Events in the Tracking Logs* section.
16 Jan 2015	Added the *Institution-wide Data* section with information about the CSV file of student email preference settings.

1.1.5 October-December 2014

Date	Change
12/24/14	Added information about video events that the edX mobile app emits to the *Events in the Tracking Logs* section.
12/18/14	Updated descriptions of the video events in the *Events in the Tracking Logs* section.
11/26/14	Expanded the background information on content experiments in *Testing Events for Content Experiments*.
11/13/14	Updated the `student_courseenrollment.mode` description.
11/5/14	Corrected descriptions for `play_video` and `pause_video` in *Video Interaction Events*.
10/28/14	Added best practices for passphrases to the *Keys and Credentials for Data Transfers* section.
10/23/14	Added examples of the format used to identify course components to the *Student Info and Progress Data* and *Events in the Tracking Logs* sections.
	Updated the `child_render` event to reflect the name change for the `child_id` member field. See *Events in the Tracking Logs*.
10/20/14	Updated the *Data Delivered in Data Packages* section to remove instructions for downloading weekly event files.
10/16/14	Updated video events with new fields relating to mobile device use in the *Events in the Tracking Logs* section.
10/07/14	Added new student and course team events relating to cohort use to the *Events in the Tracking Logs* section.
	Removed information about XML course formats. See the edX Open Learning XML Guide for information about building XML courses.

1.1.6 July-September 2014

Date	Change
09/30/14	Added information about the data that is available to course teams to the *Data Delivered in Data Packages* section.
09/18/14	Added descriptions of two columns added to the *auth_userprofile table*.
09/08/14	Added cautions to the *Keys and Credentials for Data Transfers* section.
09/04/14	Updated the *Discussion Forums Data* section to include the `thread_type` field for CommentThreads and the `endorsement` field for Comments.
08/25/14	Removed information on course grading. See Establishing a Grading Policy For Your Course in *Building and Running an edX Course*.
	Removed information on the XML for drag and drop. See Drag and Drop Problem in *Building and Running an edX Course*.
08/12/14	Added the *Open Response Assessment Events* section to the *Events in the Tracking Logs* section.
08/01/14	Added the *Data Delivered in Data Packages* section with information to help data czars locate and download data package files.
07/10/14	Added the *Keys and Credentials for Data Transfers* section with information to help new data czars set up credentials for secure data transfers.

1.1.7 April-June 2014

Date	Change
06/27/14	Made a correction to the `edx.forum.searched` event name in the *Events in the Tracking Logs* section.
	Added the `stop_video` event to the *Events in the Tracking Logs* section.
	Updated the `seek_video` event in the *Events in the Tracking Logs* section.
06/23/14	Added a *Preface* with resources for course teams, developers, researchers, and students.
05/23/14	Added descriptions of the enrollment upgrade events to the *Events in the Tracking Logs* section.
05/22/14	Added descriptions of five video- and problem-related events to the *Events in the Tracking Logs* section.
	Added the new `edx.forum.searched` event to the *Events in the Tracking Logs* section.
05/06/14	Added enrollment event types to the *Events in the Tracking Logs* section.
05/05/14	Removed information on the Poll module. See Poll Tool for OLX in *Building and Running an edX Course*.
	Removed information on the Word Cloud tool. See Word Cloud Tool in *Building and Running an edX Course*.
	Removed information on CustomResponse XML and Python Script. See Write-Your-Own-Grader Problem in the *Building and Running an edX Course* guide.
	Removed information on Formula Equation Input. See *partnercoursestaff:Math Expression Input* in the *Building and Running an edX Course* guide.
04/29/14	Corrected misstatement on how *Discussion Forums Data* is sent in data packages.
04/25/14	Added new event types to the *Events in the Tracking Logs* section for interactions with PDF files.

1.1.8 January-March 2014

Date	Change
03/31/14	Added new fields for the server `problem_check` event type to the *Events in the Tracking Logs* section.
	Reformatted the *Events in the Tracking Logs* section to improve readability.
03/28/14	Added the *Data Czar/Data Team Selection and Responsibilities* section.
03/24/14	Added the `user_api_usercoursetag` table to the *Student Info and Progress Data* section and the `assigned_user_to_partition` and `child_render` event types to the *Events in the Tracking Logs* section.
03/19/14	Provided alternative formatting for the examples in the *Discussion Forums Data* section.
03/13/14	Updated the *Student Info and Progress Data* section.
02/24/14	Added descriptions of new fields to the *Wiki Data* section.
02/21/14	Added descriptions of new fields to the *Discussion Forums Data* section.
02/14/14	Added the `seek_video` and `speed_change_video` event types to the *Events in the Tracking Logs* section.

1.2 Read Me

The *edX Research Guide* 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/docs/en_us/data/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. All pull requests need approval from edX. For more information, contact edX at docs@edx.org.

1.3 Preface

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.

1.3.1 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.

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.

1.3.2 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.

1.3.3 Release Announcements through the Open edX Portal

To receive and share product and release announcements by email, subscribe to announcements on the Open edX Portal.

1. Create an account on the Open edX Portal as described above.
2. Go to https://open.edx.org/announcements.
3. Under **Announcement Type** in the **Subscriptions** block, select the type of announcements that you want to receive through email.
4. Select **Save**.

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

Note: EdX partners can complete the same steps on the **Announcements** page in the edX Partner Portal.

1.3.4 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.

1.3.5 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.

edX101: Overview of Creating a Course

The edX101 course was built in Studio and is available for enrollment on edx.org. This course takes one to two hours to complete, and is designed to provide a high-level overview of the course creation and delivery process. It also highlights the extensive capabilities of the edX platform.

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

- 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.

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 edX documentation. This is a known limitation.

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.

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.

1.3.6 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.

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 edX documentation. This is a known limitation.

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.

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.

1.3.7 Resources for Developers

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

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 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 edX documentation. This is a known limitation.

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.

Email and IRC

To receive and share information by email, developers can join these Google groups to ask questions and participate in discussions with peers and edX staffers.

- For conversations about the code in Open edX, join edx-code.
- For conversations about running Open edX, join openedx-ops.
- For conversations about globalization and translation, join openedx-translation.

Additional Google groups are occasionally formed for individual projects.

Note: Please do not report security issues in public. If you have a concern, please email security@edx.org.

EdX engineers often monitor the Freenode #edx-code IRC channel.

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 pull request dashboard is a visualization of the count and age of the pull requests (PRs) assigned to teams at edX. Select the bars in this chart to get more information about the PRs.

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

1.3.8 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 courseware, we encourage course teams to provide learners with links to this guide as needed in course updates or discussions.
- Installing, Configuring, and Running the 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 edX documentation. This is a known limitation.

1.3.9 Resources for Learners

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 courseware, we encourage course teams to provide learners with links to these guides as needed in course updates or discussions.

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, or opportunities for Google Hangouts. Be sure to check the **Course Info** 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**.

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.

1.4 edX Browser Support

The edX Platform runs on the following browsers.

- Chrome
- Safari
- Firefox
- Internet Explorer

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

The edX Platform is routinely tested and verified on the current and previous version of each of these browsers. We generally encourage the use of and fully support only the latest version.

This information is updated as new major operating system and browser versions are released. All point releases are supported unless noted; occasional exceptions are based on specific bug fixes or feature updates.

1.4.1 edX Learning Management System

The following table shows operating system and browser support for the edX learning management system (LMS), which learners and course teams use to interact with course content.

	Chrome	Safari	Firefox	IE 11	IE 10
Windows 8	Yes	N/A	Yes	Yes	Yes
Mac OSX Mavericks or Yosemite	Yes	Yes	Yes	N/A	N/A

For more information about the LMS, see Building and Running an edX Course.

1.4.2 edX Studio

The following table shows operating system and browser support for edX Studio, which course teams use to build a course.

	Chrome	Safari	Firefox	IE 11	IE 10
Windows 8	Yes	N/A	Yes	Provisional	Provisional
Mac OSX Mavericks or Yosemite	Yes	Yes	Yes	N/A	N/A

For more information about Studio, see Building and Running an edX Course.

1.4.3 edX Insights

The following table shows operating system and browser support for edX Insights, which course teams use to review and download data about their courses and learners.

	Chrome	Safari	Firefox	IE 11	IE 10
Windows 8	Yes	N/A	Yes	Provisional	Provisional
Mac OSX Mavericks or Yosemite	Yes	Yes	Yes	N/A	N/A

For more information about edX Insights, see Using edX Insights.

CHAPTER 2

Data Czar/Data Team Selection and Responsibilities

A data czar is the single representative at a partner institution who has the credentials to download and decrypt edX data packages. The data czar is responsible for transferring data securely to researchers and other interested parties after it is received. Due to the sensitivity of this data, the responsibility for these activities is restricted to one individual.

Important: As a best practice for working with student data, edX strongly recommends a single data czar at each partner institution. However, if an additional individual is given this responsibility at your institution, be sure to work with edX to set up individual credentials for that additional data czar.

At each partner institution, the data czar is the primary point of contact for information about edX data.

- *Skills and Experience of Data Czars*
- *Resources and Information*

At some institutions, only the data czar works on research projects that use the course data in edX data packages. At other institutions, the data czar works with a team of additional contributors, or is responsible only for making a secure transfer of the data to the research team. Typically, the data team includes members in the following roles (or a data czar with these skill sets):

- Database administrators work with the SQL and NoSQL data files and write queries on the data.
- Statisticians and data analysts mine the data.
- Educational researchers pose questions and interpret the results of queries on the data.

See *Skills and Experience of Other Contributors*.

All of the individuals who are permitted to access the data should be trained in, and comply with, their institution's secure data handling protocols.

2.1 Skills and Experience of Data Czars

The individuals who are selected by a partner institution to be edX data czars typically have experience working with sensitive student data, are familiar with encryption/decryption and file transfer protocols, and can validate, copy, move, and store large files. The data czar is responsible for ensuring compliance with your institution's and country's regulations with respect to the sharing of this data.

2.1.1 General Skills

- Ability to set up and manage data access.

- Knowledgeable of general data privacy and security best practices.
- Experience with management of sensitive student data.

2.1.2 Technical Skills

- Familiarity with PGP and GPG encryption and decryption.
- Ability to download large files from Amazon Web Service (AWS) Simple Storage Service (S3).
- Experience working with archive files in TAR, GZ, and ZIP formats.
- Familiarity with SQL and noSQL (Mongo) databases.
- Familiarity with CSV and JSON file formats.
- Experience copying, moving, and storing large files in bulk.
- Ability to validate the data and files received and distributed.

2.2 Resources and Information

The edX Analytics team adds every data czar to a Google Group and mailing list called course-data.

EdX also hosts an Open edX Analytics wiki that is available to the public. The wiki provides links to the engineering roadmap, information about operational issues, and release notes describing past releases.

2.3 Skills and Experience of Other Contributors

In addition to the data czar, each partner institution assembles a team of contributors to their research projects. This team can include database administrators, software engineers, data specialists, and educational researchers. The team can be large or small, but collectively its members need to be able to work with SQL and NoSQL databases, write queries, and convert the data from raw formats into standard research packages, such as CSV files, spreadsheets, or other desired formats.

2.3.1 General Skills

- Attention to detail.
- Experience setting up and testing a data conversion pipeline.
- Ability to identify interesting features in a complex and rich data set.
- Familiarity with anonymization and obfuscation techniques.
- Familiarity with data privacy and security best practices.
- Experience managing sensitive student data.

2.3.2 Technical Skills

- Familiarity with CSV, MongoDB, JSON, Unicode, XML, HTML.
- Ability to set up, query, and administer both SQL and noSQL databases.

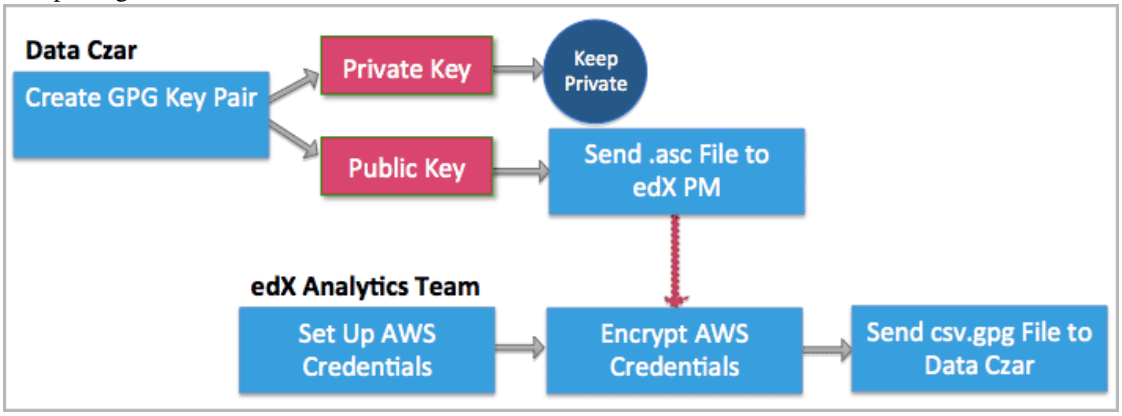
- Experience with console/bash scripts.
- Basic or advanced scripting (for example, using Python or Ruby) to convert, join, and aggregate data from different data sources, handle JSON serialization, and Unicode specificities.
- Experience with data mining and data aggregation across a rich, varied data set.
- Ability to write parsing scripts that properly handle JSON serialization and Unicode.

CHAPTER 3

Keys and Credentials for Data Transfers

EdX transfers course data to the data czars at our partner institutions in regularly generated data packages. Data packages can only be accessed by a single contact at each university, referred to as the "data czar".

The data czar who is selected at each institution sets up keys for securely transferring files from edX to the partner institution. Meanwhile, the Analytics team at edX sets up credentials so that the data czar can log in to the site where data packages are stored.



After these steps for setting up credentials are complete, the data czar can download data packages on an ongoing basis.

3.1 Data Czar: Create Keys for Encryption and Decryption

To assure the security of data packages, the edX Analytics team encrypts all files before making them available to a partner institution. As a result, when you receive a data package (or other files) from the edX Analytics team, you must decrypt the files that it contains before you use them.

The cryptographic processes of encrypting and decrypting data files require that you create a pair of keys: the public key in the pair, which you send to the edX Analytics team, is used to encrypt data. You use your corresponding private key to decrypt any files that have been encrypted with that public key.

To create the keys needed for this encryption and decryption process, you use GNU Privacy Guard (GnuPG or GPG). Essentially, you install a cryptographic application on your local computer and then supply your email address and a secret passphrase (a password).

Important:

- The email address that you supply when you create your keys must be your official email address at your edX partner institution.
- After you specify the passphrase, be sure to take any steps necessary to assure that you can use it in the future. To minimize security risks, GPG does not provide a mechanism for supplying you with a reminder hint.
- Do not reveal your passphrase to anyone else.

The result is the public key that you send to edX to use in encrypting data files for your institution, and the private key which you keep secret and use to decrypt the encrypted files that you receive. Creating these keys is a one-time process that you coordinate with your edX partner manager. Instructions for creating the keys on Windows or Macintosh follow.

For more information about GPG encryption and creating key pairs, see the Gpg4win Compendium.

3.1.1 Create Keys: Windows

1. Go to the Gpg4win website and download the most recent version of Gpg4win.
2. Install Gpg4win and then open the Kleopatra Gpg4win application. A wizard presents a series of dialog boxes to collect information from you and generate your public key (called a certificate in Kleopatra).
   (a) When you are prompted to specify the type of key pair you want, click **Create personal OpenPGP key pair**.
   (b) When you are prompted for your email address, be sure to enter *your official university or institution email address*. EdX cannot use public keys that are based on personal or other non-official email addresses to encrypt data.
   (c) When you are prompted for a passphrase, enter a strong passphrase. Be sure to select a passphrase that you can remember, or use a secure method of retaining it for reuse in the future: you use this passphrase when you decrypt your data packages.
3. When Kleopatra presents the **Key Pair Successfully Created** dialog box, click **Send Certificate by EMail** to send the public key (and only the public key) to your edX partner manager.
4. Optionally, click **Make a Backup Copy of Your Key Pair** to store both of the keys on a removable data storage device.

Important: Do not reveal your passphrase, or share your private key, with anyone else. If you need another person to be able to transfer and decrypt files, work with edX to set her or him up as an additional data czar. Data czars must create and use their own passphrases.

3.1.2 Create Keys: Macintosh

1. Go to the GPG Tools website. Scroll down to the **GPG Suite** section of the page and click **Download GPG Suite**.
2. When the download is complete, click the .dmg file to begin the installation.

   When installation is complete, GPG Keychain Access opens a web page with First Steps and a dialog box.
3. Enter your name and email address. Be sure to enter *your official university or institution email address*. EdX cannot use public keys that are based on personal or other non-official email addresses to encrypt data.
4. Click **Generate key**. A dialog box opens to prompt you for a passphrase.

5. Enter a strong passphrase. Be sure to select a passphrase that you can remember, or use a secure method of retaining it for reuse in the future: you use this passphrase when you decrypt your data packages.
6. To send only your public key to your edX partner manager, click the key and then click **Export**. A dialog box opens.
    1. Specify a file name and location to save the file.
    2. Make sure that **Format** is set to ASCII.
    3. Make sure that **Allow secret key export** is cleared.

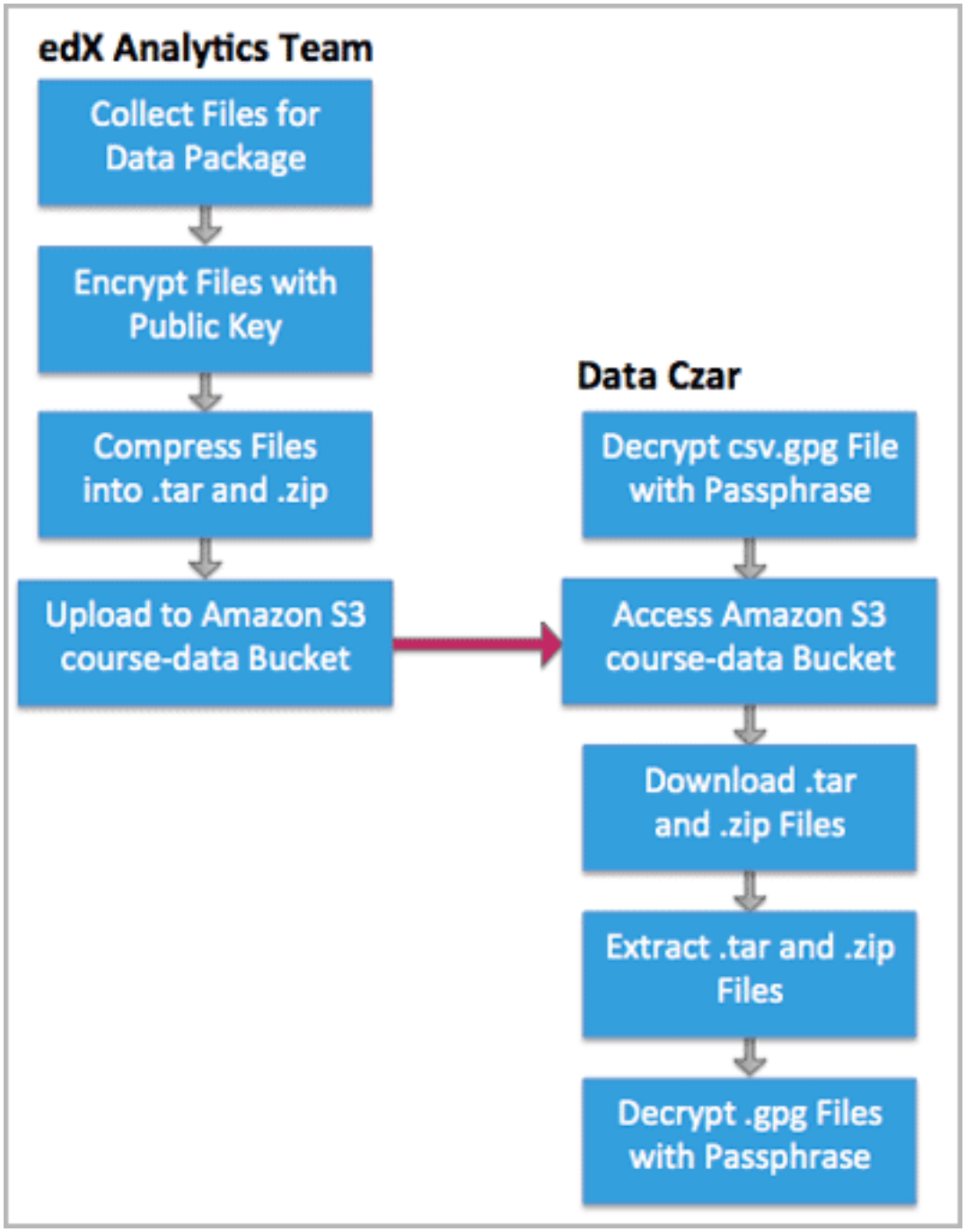
    When you click **Save**, only the public key is saved in the resulting .asc file. Do not share your private key with edX or any third party.
7. Compose an e-mail message to your edX partner manager. Attach the .asc file that you saved in the previous step to the message, then send the message.

3.2 EdX: Deliver Credentials for Accessing Data Storage

The data packages that edX prepares for each partner organization are uploaded to the Amazon Web Service (AWS) Simple Storage Service (Amazon S3). The edX Analytics team creates an individual account to access this storage service for each data czar. The credentials for accessing this account are called an Access Key and a Secret Key.

After the edX Analytics team creates these access credentials for you, they use the public encryption key that you sent your partner manager to encrypt the credentials into a **credentials.csv.gpg** file. The edX Analytics team then sends the file to you as an email attachment.

The **credentials.csv.gpg** file is likely to be the first file that you decrypt with your private GPG key. You use the same process to decrypt the data package files that you retrieve from Amazon S3. See *Decrypt an Encrypted File*.



3.3 Decrypt an Encrypted File

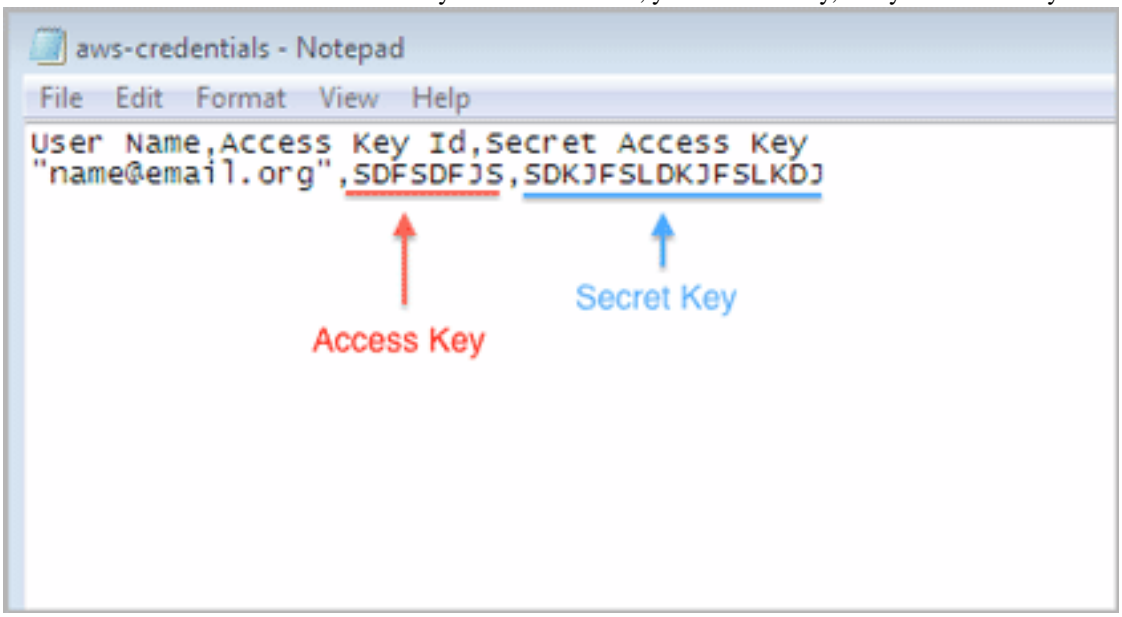
To work with an encrypted .gpg file, you use the same GNU Privacy Guard program that you used to create your public/private key pair. You use your private key to decrypt the Amazon S3 credentials file and the files in your data packages.

1. Save the encrypted file in an accessible location.
2. On a Windows computer, open Windows Explorer. On a Macintosh, open Finder.

3. Navigate to the file and right-click it.
4. On a Windows computer, select **Decrypt and verify**, and then click **Decrypt/Verify**. Do not change any other setting.

   On a Macintosh, select **Services**, and then click **OpenPGP: Decrypt File**.
5. Enter your passphrase. The GNU Privacy Guard program decrypts the file.

For example, when you decrypt the credentials.csv.gpg file the result is a credentials.csv file. Open the decrypted credentials.csv file to see that it contains your email address, your Access Key, and your Secret Key.



3.4 Access Amazon S3

To connect to Amazon S3, you must have your decrypted credentials. You may want to have a third-party tool that gives you a user interface for managing files and transferring them from Amazon S3 to your network. Some data czars use applications like CloudBerry Explorer for Amazon S3, Bucket Explorer, or S3 Browser. Alternatively, you can use the AWS Command Line Interface.

1. Select and install a third-party tool or interface to manage your S3 account.
2. Open your decrypted `credentials.csv` file. This file contains your AWS Access Key and your AWS Secret Key.
3. Open the third-party tool.
4. Enter information to connect to the S3 account.

   For example, you might need to select an option such as **Open Connection**, and then supply the service you want to connect to (Amazon S3), your Access Key, and your Secret Key. For more information, see the documentation provided for the tool that you selected.
5. To access the database data files, specify or select `s3://course-data`.

   To access the event data files, specify or select `s3://edx-course- data/{org}/`. You must include the identifier for your organization after the name of the bucket.

Note: If you are using a third-party tool to connect to Amazon S3, you might not be able to navigate directly between `s3://course-data` and `s3://edx-course-data/{org}/`. You might need to disconnect from Amazon S3 and then reconnect to specify the other destination.

For information about the files found at each of these Amazon S3 destinations, see *Data Delivered in Data Packages*.

CHAPTER 4

Data Delivered in Data Packages

For partners who are running courses on edx.org and edge.edx.org, edX regularly makes research data available for download from the Amazon S3 storage service. The *data package* that data czars download from Amazon S3 consists of a set of compressed and encrypted files that contain event logs and database snapshots for all of their organizations' edx.org and edge.edx.org courses.

Course-specific data is also available to the members of individual course teams. Users who are assigned the Admin or Staff role for the course can view and download data from the Instructor Dashboard in their live courses and from edX Insights. The data available to course teams from these applications is a subset of the data available in the data packages. For more information, see Building and Running an edX Course and Overview.

4.1 Data Package Files

A data package consists of different files that contain event data and database data.

Note: In all file names, the date is in {YYYY}-{MM}-{DD} format.

You download these files from different Amazon S3 "buckets" and folders. See *Amazon S3 Buckets and Folders*.

4.1.1 Event Data

The `{org}-{site}-events-{date}.log.gz.gpg` file contains a daily log of course events. A separate file is available for courses running on edge.edx.org (with "edge" for {site} in the file name) and on edx.org (with "prod" for {site}).

For a partner organization named UniversityX, these daily files are identified by the organization name, the edX site name, and the date. For example, `universityx-edge-events-2014-07-25.log.gz.gpg`.

Each of these compressed files can range in size from hundreds of kilobytes to tens of megabytes. When you extract a compressed file, it is approximately 20 times larger. As a result, multiple gigabytes of space might be needed to store the tracking logs for a year.

For information about the contents of these files, see *Data Package Contents*.

4.1.2 Database Data

The `{org}-{date}.zip` file contains views on database tables. This file includes data as of the time of the export, for all of an organization's courses on both the edx.org and edge.edx.org. sites. A new file is available every week, representing the database at that point in time.

For a partner organization named UniversityX, each weekly file is identified by the organization name and its extraction date: for example, `universityx-2013-10-27.zip`.

Compressed, these files can range in size from hundreds of megabytes to tens of gigabytes in size. When you extract a compressed file, it is approximately 20 times larger. As a result, institutions that receive data for several courses for several years might require from tens to hundreds of gigabytes of space for data storage.

For information about the contents of this file, see *Data Package Contents*.

4.2 Amazon S3 Buckets and Folders

Data package files are located at the following Amazon S3 destinations:

- The **s3://edx-course-data/{org}** folder contains the daily `{org}-{site}-events-{date}.log.gz.gpg` files of course event data.
- The **s3://course-data** bucket contains the weekly `{org}-{date}.zip` database snapshot.

For information about accessing Amazon S3, see *Access Amazon S3*.

4.3 Download Data Packages from Amazon S3

You download the files in your data package from the Amazon S3 storage service.

4.3.1 Download Daily Event Files

1. To download daily event files, use the AWS Command Line Interface or a third-party tool to connect to the **s3://edx-course-data/{org}** folder on Amazon S3.

   For information about providing your credentials to connect to Amazon S3, see *Access Amazon S3*.

2. Navigate within **s3://edx-course-data/{org}** to locate the files that you want:

   `{org}/{site}/events/{year}`

   The event logs in the `{year}` folder are in compressed, encrypted files named `{org}-{site}-events-{date}.log.gz.gpg`.

3. Download the `{org}-{site}-events-{date}.log.gz.gpg` file.

   If your organization has courses running on both edx.org and edge.edx.org, separate log files are available for the "prod" site and the "edge" site. Repeat this step to download the file for the other site.

4.3.2 Download Weekly Database Files

Note: If you are using a third-party tool to connect to Amazon S3, you might not be able to navigate directly between the **s3://course-data** bucket and the **s3://edx-course-data/{org}** folder. You might need to disconnect from Amazon S3 and then reconnect to the other destination.

1. To download a weekly database data file, connect to the edX **s3://course-data** bucket on Amazon S3 using the AWS Command Line Interface or a third-party tool.

   For information about providing your credentials to connect to Amazon S3, see *Access Amazon S3*.

2. Download the `{org}-{date}.zip` database data file from the **s3://course-data** bucket.

4.4 Data Package Contents

Each of the files you download contains one or more files of research data.

4.4.1 Extracted Contents of `{org}-{site}-events-{date}.log.gz.gpg`

The `{org}-{site}-events-{date}.log.gz.gpg` file contains all event data for courses on a single edX site for one 24-hour period. After you download a `{org}-{site}-events-{date}.log.gz.gpg` file for your institution, you:

1. Use your private key to decrypt the file. See *Decrypt an Encrypted File*.
2. Extract the log file from the compressed .gz file. The result is a single file named `{org}-{site}-events-{date}.log`. (Alternatively, the data can be decompressed in stream using a tool such as gzip.)

For more information about the events in this file, see *Events in the Tracking Logs*.

4.4.2 Extracted Contents of `{org}-{date}.zip`

After you download the `{org}-{date}.zip` file for your institution, you:

1. Extract the contents of the file. When you extract (or unzip) this file, all of the files that it contains are placed in the same directory. All of the extracted files end in `.gpg`, which indicates that they are encrypted.
2. Use your private key to decrypt the extracted files. See *Decrypt an Encrypted File*.

The result of extracting and decrypting the `{org}-{date}.zip` file is the following set of .sql, .csv, and .mongo files. Note that the .sql files are tab separated.

`{org}-{course}-{date}-auth_user-{site}-analytics.sql`

Information about the users who are authorized to access the course. See *Columns in the auth_user Table*.

`{org}-{course}-{date}-auth_userprofile-{site}-analytics.sql`

Demographic data provided by users during site registration. See *Columns in the auth_userprofile Table*.

`{org}-{course}-{date}-certificates_generatedcertificate-{site}-analytics.sql`

The final grade and certificate status for students (populated after course completion). See *Columns in the certificates_generatedcertificate Table*.

`{org}-{course}-{date}-course_structure-{site}-analytics.json`

This file documents the structure of a course at a point in time. The file includes data for the course, including important dates, pages, and course- wide discussion topics. It also identifies each item of course content defined in the course outline. A separate file is included for each course on the site. For more information, see *Course Content Data*.

`{org}-{course}-{date}-courseware_studentmodule-{site}-analytics.sql`

The courseware state for each student, with a separate row for each item in the course content that the student accesses. No file is produced for courses that do not have any records in this table (for example, recently created courses). See *Columns in the courseware_studentmodule Table*.

`{org}-email_opt_in-{site}-analytics.csv`

This file reports the email preference selected by students who are enrolled in any of your institution's courses. See *Institution-wide Data*.

`{org}-{course}-{date}-student_courseenrollment-{site}-analytics.sql`

The enrollment status and type of enrollment selected by each student in the course. See *Columns in the student_courseenrollment Table*.

`{org}-{course}-{date}-user_api_usercoursetag-{site}-analytics.sql`

Metadata that describes different types of student participation in the course. See *Columns in the user_api_usercoursetag Table*.

`{org}-{course}-{date}-user_id_map-{site}-analytics.sql`

A mapping of user IDs to site-wide obfuscated IDs. See *Columns in the user_id_map Table*.

`{org}-{course}-{date}-{site}.mongo`

The content and characteristics of course discussion interactions. See *Discussion Forums Data*.

`{org}-{course}-{date}-wiki_article-{site}-analytics.sql`

Information about the articles added to the course wiki. See *Fields in the wiki_article file*.

`{org}-{course}-{date}-wiki_articlerevision-{site}-analytics.sql`

Changes and deletions affecting course wiki articles. See *Fields in the wiki_articlerevision file*.

CHAPTER 5

Student Info and Progress Data

The following sections detail how edX stores stateful data for students internally, and is useful for developers and researchers who are examining database exports. Data for students is presented in these categories.

- *User Data*
- *Courseware Progress Data*
- *Certificate Data*

5.1 Conventions

- EdX uses MySQL 5.1 relational database system with InnoDb storage engine.
- All strings are stored as UTF-8.
- All datetimes are stored as UTC (Coordinated Universal Time).
- The .sql files in edX data packages are tab separated.

Note: EdX also uses the Django Python Web framework. Tables that are built into the Django Web framework are documented here only if they are used in unconventional ways.

Descriptions of the tables and columns that store student data follow, first in summary form with field types and constraints, and then with a detailed explanation of each column.

5.2 MySQL Terminology

The summary information provided about the SQL table columns uses the following MySQL schema terminology.

5.2.1 Type

The kind of data and the size of the field. When a numeric field has a length specified, the length indicates how many digits display but does not affect the number of bytes used.

Value	Description
int	4 byte integer.
small-int	2 byte integer, sometimes used for enumerated values.
tinyint	1 byte integer, usually used to indicate a Boolean with 0 = False and 1 = True.
var-char	String, typically short and indexable. The length is the number of chars, not bytes, to support multi-byte character sets.
long-text	A long block of text, usually not indexed.
date	Date
date-time	Datetime in UTC, precision in seconds.

5.2.2 Null

Value	Description
YES	NULL values are allowed.
NO	NULL values are not allowed.

Note: Django often just places blank strings instead of NULL when it wants to indicate a text value is optional. This is more meaningful for numeric and date fields.

5.2.3 Key

Value	Description
PRI	Primary key for the table, usually named `id`, unique.
UNI	Unique
MUL	Indexed for fast lookup, but the same value can appear multiple times. A unique index that allows NULL can also show up as MUL.

5.3 User Data

The following tables store data gathered during site registration and course enrollment.

- *Columns in the auth_user Table*
- *Columns in the auth_userprofile Table*
- *Columns in the student_courseenrollment Table*
- *Columns in the user_api_usercoursetag Table*
- *Columns in the user_id_map Table*
- *Columns in the student_languageproficiency Table*

The following tables store data gathered about the teams in a course.

- *Columns in the teams_courseteam Table*
- *Columns in the teams_courseteammembership Table*

Note: The Teams feature is in limited release. For more information, contact your edX partner manager. For Open edX sites, contact your system administrator.

5.3.1 Columns in the auth_user Table

The `auth_user` table is built into the edX Django Web framework. It holds generic information necessary for user login and permissions.

A sample of the heading row and a data row in the `auth_user` table follow.

```
id  username  first_name  last_name  email  password  is_staff  is_active
is_superuser  last_login  date_joined status  email_key  avatar_typ
country  show_country  date_of_birth  interesting_tags  ignored_tags
email_tag_filter_strategy display_tag_filter_strategy
consecutive_days_visit_count

9999999    AAAAAAAAA    AAAAAA  AAAAAA 1 1 0 2014-01-01 17:28:27 2012-03-04
00:57:49   NULL      0 NULL      0 0
```

The `auth_user` table has the following columns.

Column	Type	Null	Key	Comment
id	int(11)	NO	PRI	
username	varchar(30)	NO	UNI	
first_name	varchar(30)	NO		# Never used
last_name	varchar(30)	NO		# Never used
email	varchar(75)	NO	UNI	
password	varchar(128)	NO		
is_staff	tinyint(1)	NO		
is_active	tinyint(1)	NO		
is_superuser	tinyint(1)	NO		
last_login	datetime	NO		
date_joined	datetime	NO		
status	varchar(2)	NO		# Obsolete
email_key	varchar(32)	YES		# Obsolete
avatar_typ	varchar(1)	NO		# Obsolete
country	varchar(2)	NO		# Obsolete
show_country	tinyint(1)	NO		# Obsolete
date_of_birth	date	YES		# Obsolete
interesting_tags	longtext	NO		# Obsolete
ignored_tags	longtext	NO		# Obsolete
email_tag_filter_strategy	smallint(6)	NO		# Obsolete
display_tag_filter_strategy	smallint(6)	NO		# Obsolete
consecutive_days_visit_count	int(11)	NO		# Obsolete

id

Primary key, and the value typically used in URLs that reference the user. A user has the same value for `id` here as they do in the MongoDB database's users collection. Foreign keys referencing `auth_user.id` will often be named `user_id`, but are sometimes named `student_id`.

username

The unique username for a user in the edX system. It can contain alphanumerics and the special characters shown within the brackets: [_ @ + - .]. The username is the only user-provided information that other users can currently see. EdX has never allowed users to change usernames, but may do so in the future.

first_name

Not used; a user's full name is stored in `auth_userprofile.name` instead.

last_name

Not used; a user's full name is stored in `auth_userprofile.name` instead.

email

The user's email address, which is the primary mechanism users use to log in. This value is optional by default in Django, but is required by edX. This value must be unique to each user and is never shown to other users.

password

A hashed version of the user's password. Depending on when the password was last set, this will either be a SHA1 hash or PBKDF2 with SHA256 (Django 1.3 uses the former and 1.4 the latter).

is_staff

Most users have a 0 for this field. Set to 1 if the user is a staff member of **edX**, with corresponding elevated privileges that cut across courses. It does not indicate that the person is a member of the course team for any given course.

Generally, users with this flag set to 1 are either edX partner managers responsible for course delivery, or edX developers who need access for testing and debugging purposes. Users who have `is_staff` = 1 have Admin privileges on all courses and can see additional debug information on the Instructor Dashboard.

Note: This designation has no bearing on a user's role in the discussion forums, and confers no elevated privileges there.

---

is_active

This value is 1 if the user has clicked on the activation link that was sent to them when they created their account, and 0 otherwise.

Users who have `is_active` = 0 generally cannot log into the system. However, when users first create an account, they are automatically logged in even though they have not yet activated the account. This is to let them experience the site immediately without having to check their email. A message displays on the dashboard to remind users to check their email and activate their accounts when they have time. When they log out, they cannot log back in again until activation is complete. However, because edX sessions last a long time, it is possible for someone to use the site as a student for days without being "active".

Once `is_active` is set to 1, it is *only* set back to 0 if the user is banned (which is a very rare, manual operation).

is_superuser

Controls access to django_admin views. Set to 1 (true) only for site admins. 0 for almost everybody.

History: Only the earliest developers of the system have this set to 1, and it is no longer really used in the codebase.

last_login

A datetime of the user's last login. Should not be used as a proxy for activity, since people can use the site all the time and go days between logging in and out.

date_joined

Date that the account was created.

Note: This is not the date that the user activated the account.

Obsolete columns

All of the following columns were added by an application called Askbot, a discussion forum package that is no longer part of the system.

- status
- email_key
- avatar_typ
- country
- show_country
- date_of_birth
- interesting_tags
- ignored_tags
- email_tag_filter_strategy
- display_tag_filter_strategy
- consecutive_days_visit_count

Only users who were part of the prototype 6.002x course run in the Spring of 2012 have any information in these columns. Even for those users, most of this information was never collected. Only the columns with values that are automatically generated have any values in them, such as the tag-related columns.

These columns are unrelated to the discussion forums that edX currently uses, and will eventually be dropped from this table.

5.3.2 Columns in the auth_userprofile Table

The `auth_userprofile` table stores user demographic data collected when students register for a user account. Every row in this table corresponds to one row in `auth_user`.

A sample of the heading row and a data row in the `auth_userprofile` table follow.

```
id  user_id name  language  location  meta  courseware  gender
mailing_address year_of_birth level_of_education  goals allow_certificate
country  city  bio   profile_image_uploaded_at

9999999  AAAAAAAA  AAAAAAAAA English MIT {"old_emails":
[["aaaaa@xxxxx.xxx", "2012-11-16T10:28:10.096489"]], "old_names":
[["BBBBBBBBBBBBB", "I wanted to test out the name-change functionality",
"2012-10-22T12:23:10.598444"]]} course.xml  NULL  NULL  NULL  NULL  NULL
1      NULL   Hi! I'm from the US and I've taken 4 edX courses so far. I
want to learn how to confront problems of wealth inequality. 2015-04-19 16:41:27
```

The `auth_userprofile` table has the following columns.

Column	Type	Null	Key	Comment
id	int(11)	NO	PRI	
user_id	int(11)	NO	UNI	
name	varchar(255)	NO	MUL	
language	varchar(255)	NO	MUL	# Obsolete
location	varchar(255)	NO	MUL	# Obsolete
meta	longtext	NO		
courseware	varchar(255)	NO		# Obsolete
gender	varchar(6)	YES	MUL	# Only users signed up after prototype
mailing_address	longtext	YES		# Only users signed up after prototype
year_of_birth	int(11)	YES	MUL	# Only users signed up after prototype
level_of_education	varchar(6)	YES	MUL	# Only users signed up after prototype
goals	longtext	YES		# Only users signed up after prototype
allow_certificate	tinyint(1)	NO		
country	varchar(2)	YES		
city	longtext	YES		
bio	varchar(3000)	YES		
profile_image_uploaded_at	datetime	YES		

History: `bio` and `profile_image_uploaded_at` added 22 April 2015. `country` and `city` added January 2014. The organization of this table was different for the students who signed up for the MITx prototype phase in the spring of 2012, than for those who signed up afterwards. The column descriptions that follow detail the differences in the demographic data gathered.

id

Primary key, not referenced anywhere else.

user_id

A foreign key that maps to `auth_user.id`.

name

String for a user's full name. EdX makes no constraints on language or breakdown into first/last name. The names are never shown to other students. International students usually enter a romanized version of their names, but not always. Name changes are permitted, and the previous name is logged in the `meta` field.

History: A former edX policy required manual approval of name changes to guard the integrity of the certificates. Students would submit a name change request, and an edX team member would approve or reject the request.

language

No longer used.

History: User's preferred language, asked during the sign up process for the 6.002x prototype course given in the Spring of 2012. Sometimes written in those languages. EdX stopped collecting this data after MITx transitioned to edX, but never removed the values for the first group of students.

location

No longer used.

History: User's location, asked during the sign up process for the 6.002x prototype course given in the Spring of 2012. The request was not specific, so people tended to put the city they were in, though some just supplied a country and some got as specific as their street address. Again, sometimes romanized and sometimes written in their native language. Like `language`, edX stopped collecting this column after MITx transitioned to edX, so it is only available for the first batch of students.

meta

An optional, freeform text field that stores JSON data. This field allows us to associate arbitrary metadata with a user. An example of the JSON that can be stored in this field follows, using pretty print for an easier-to-read display format.

```
{
 "old_names": [
   [
     "Mike Smith",
     "Mike's too informal for a certificate.",
     "2012-11-15T17:28:12.658126"
   ],
   [
     "Michael Smith",
     "I want to add a middle name as well.",
     "2013-02-07T11:15:46.524331"
   ]
 ],
 "old_emails": [
   [
     "mr_mike@email.com",
     "2012-10-18T15:21:41.916389"
   ]
 ],
 "6002x_exit_response": {
```

```
    "rating": [
      "6"
    ],
    "teach_ee": [
      "I do not teach EE."
    ],
    "improvement_textbook": [
      "I'd like to get the full PDF."
    ],
    "future_offerings": [
      "true"
    ],
    "university_comparison": [
      "This course was <strong>on the same level<\/strong> as the university class."
    ],
    "improvement_lectures": [
      "More PowerPoint!"
    ],
    "highest_degree": [
      "Bachelor's degree."
    ],
    "future_classes": [
      "true"
    ],
    "future_updates": [
      "true"
    ],
    "favorite_parts": [
      "Releases, bug fixes, and askbot."
    ]
  }
}
```


Details about this metadata follow. Please note that the "fields" described here are found as JSON attributes *inside* a given `meta` field, and are *not* separate database columns of their own.

`old_names`

A list of the previous names this user had, and the timestamps at which they submitted a request to change those names. These name change request submissions used to require a staff member to approve it before the name change took effect. This is no longer the case, though their previous names are still recorded.

Note that the value stored for each entry is the name they had, not the name they requested to get changed to. People often changed their names as the time for certificate generation approached, to replace nicknames with their actual names or correct spelling/punctuation errors.

The timestamps are UTC, like all datetimes stored in the edX database.

`old_emails`

A list of previous emails this user had, with timestamps of when they changed them, in a format similar to *old_names*. There was never an approval process for this.

The timestamps are UTC, like all datetimes stored in the edX database.

`6002x_exit_response`

Answers to a survey that was sent to students after the prototype 6.002x course in the Spring of 2012. The questions and number of questions were randomly selected to measure how much survey length affected response rate. Only students from this course have this field.

courseware

No longer used.

History: At one point, it was part of a way to do A/B tests, but it has not been used for anything meaningful since the conclusion of the prototype course in the spring of 2012.

gender

Collected during student signup from a drop-down list control.

Value	Description
f	Female
m	Male
o	Other
(blank)	User did not specify a gender.
NULL	This student signed up before this information was collected.

History: This information began to be collected after the transition from MITx to edX; prototype course students have NULL for this field.

mailing_address

Collected during student registration from a text field control. A blank string for students who elect not to enter anything.

This column can contain multiple lines, which are separated by '`\r\n`'.

History: This information began to be collected after the transition from MITx to edX; prototype course students have NULL for this field.

year_of_birth

Collected during student registration from a drop-down list control. NULL for students who decide not to fill this in.

History: This information began to be collected after the transition from MITx to edX; prototype course students have NULL for this field.

level_of_education

Collected during student registration from a drop-down list control.

Value	Description
p	Doctorate.
m	Master's or professional degree.
b	Bachelor's degree.
a	Associate degree.
hs	Secondary/high school.
jhs	Junior secondary/junior high/middle school.
el	Elementary/primary school.
none	None.
other	Other.
(blank)	User did not specify level of education.
p_se	Doctorate in science or engineering (no longer used).
p_oth	Doctorate in another field (no longer used).
NULL	This student signed up before this information was collected.

History: Data began to be collected in this column after the transition from MITx to edX; prototype course students have NULL for this field.

goals

Collected during student registration from a text field control with the label "Goals in signing up for edX". A blank string for students who elect not to enter anything.

This column can contain multiple lines, which are separated by '`\r\n`'.

History: This information began to be collected after the transition from MITx to edX; prototype course students have NULL for this field.

allow_certificate

Set to 1 (true).

History: Prior to 10 Feb 2014, this field was set to 0 (false) if log analysis revealed that the student was accessing the edX site from a country that the U.S. had embargoed. This restriction is no longer in effect, and on 10 Feb 2014 this value was changed to 1 for all users.

country

Stores a two-digit country code based on the selection made by the student during registration. Set to an empty string for students who do not select a country.

History: Added in Jan 2014, but not implemented until 18 Sep 2014. Null for all user profiles created before 18 Sep 2014.

city

Not currently used. Set to null for all user profiles.

History: Added in Jan 2014, not yet implemented.

bio

Stores one or more paragraphs of biographical information that the learner enters. The maximum number of characters is 3000.

History: Added 22 April 2015.

profile_image_uploaded_at

Stores the date and time when a learner uploaded a profile image.

History: Added 22 April 2015.

5.3.3 Columns in the student_courseenrollment Table

A row in this table represents a student's enrollment for a particular course run.

Note: A row is created for every student who starts the enrollment process, even if they never complete site registration by activating the user account.

History: As of 20 Aug 2013, this table retains the records of students who unenroll. Records are no longer deleted from this table.

A sample of the heading row and a data row in the `student_courseenrollment` table follow.

```
id  user_id course_id created is_active mode

1135683 9999999 edX/DemoX/Demo_course 2013-03-19 17:20:58 1 honor
```

The `student_courseenrollment` table has the following columns.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	NO	MUL	NULL	
course_id	varchar(255)	NO	MUL	NULL	
created	datetime	YES	MUL	NULL	
is_active	tinyint(1)	NO		NULL	
mode	varchar(100)	NO		NULL	

id

Primary key.

user_id

Student's ID in `auth_user.id`.

course_id

The ID of the course run that the user is enrolling in, in the format `{key type}:{org}+{course}+{run}`. For example, `course-v1:edX+DemoX+Demo_2014`. When you view the course content in your browser, the `course_id` appears as part of the URL. For example, `http://www.edx.org/courses/course-v1:edX+DemoX+Demo_2014/info`.

History: In October 2014, identifiers for some new courses began to use the format shown above. Other new courses, and all courses created prior to October 2014, use the format `{org}/{course}/{run}`, for example, `MITx/6.002x/2012_Fall`. The URL format for a course with a `course_id` in this format was `https://www.edx.org/courses/MITx/6.002x/2012_Fall/info`.

created

Stores the date and time that this row was created, in UTC format.

is_active

Boolean indicating whether this enrollment is active. If an enrollment is not active, a student is not enrolled in that course. For example, if a student decides to unenroll from the course, `is_active` is set to 0 (false). The student's state in `courseware_studentmodule` is untouched, so courseware state is not lost if a student unenrolls and then re-enrolls.

`is_active` can also be set to 0 if a student begins the process of enrolling in a course by purchasing a verified certificate, but then abandons the shopping cart before completing the purchase (and the enrollment).

History: This column was introduced in the 20 Aug 2013 release. Before this release, unenrolling a student simply deleted the row in `student_courseenrollment`.

mode

String indicating what kind of enrollment this is: audit, honor, professional, verified, or blank.

History:

- All enrollments prior to 20 Aug 2013 are "honor", when the "audit" and "verified" values were added.
- The "professional" value was added for courses on edx.org on 29 Sep 2014.
- The "audit" value was deprecated on 23 Oct 2014.

5.3.4 Columns in the user_api_usercoursetag Table

This table uses key-value pairs to store metadata about a specific student's involvement in a specific course. For example, for a course that assigns students to groups randomly for content experiments, a row in this table identifies the student's assignment to a partition and group.

History: Added 7 Mar 2014.

The `user_api_usercoursetag` table has the following columns.

Column	Type	Null	Key
user_id	int(11)	NO	PRI
course_id	varchar(255)	NO	
key	varchar(255)	NO	
value	textfield	NO	

user_id

The student's ID in `auth_user.id`.

course_id

The course identifier, in the format `{key type}:{org}+{course}+{run}`. For example, `course-v1:edX+DemoX+Demo_2014`.

History: In October 2014, identifiers for some new courses began to use the format shown above. Other new courses, and all courses created prior to October 2014, use the format `{org}/{course}/{run}`, for example, `MITx/6.002x/2012_Fall`.

key

Identifies an attribute of the course.

For example, for a course that includes modules that are set up to perform content experiments, the value in this column identifies a partition, or type of experiment. The key for the partition is in the format `xblock.partition_service.partition_ID`, where ID is an integer.

value

The content for the key that is set for a student.

For example, for a course that includes modules that are set up to perform content experiments, this column stores the group ID of the particular group the student is assigned to within the partition.

5.3.5 Columns in the user_id_map Table

A row in this table maps a student's real user ID to an anonymous ID generated to obfuscate the student's identity.

A sample of the heading row and a data row in the `user_id_map` table follow.

```
hash_id id  username

e9989f2cca1d699d88e14fd43ccb5b5f  9999999 AAAAAAAA
```

The `student_courseenrollment` table has the following columns.

Column	Type	Null	Key
hashid	int(11)	NO	PRI
id	int(11)	NO	
username	varchar(30)	NO	

hash_id

The user ID generated to obfuscate the student's identity.

id

The student's ID in `auth_user.id`.

username

The student's username in `auth_user.username`.

5.3.6 Columns in the student_languageproficiency Table

The `student_languageproficiency` table stores information about students' self-reported language preferences. Students can select only one value.

History: Added 22 April 2015.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
user_profile_id	int(11)	NO	MUL	NULL	
code	varchar(16)	NO	MUL	NULL	

id

A database auto-increment field that uniquely identifies the language. This field is not exposed through the API.

user_profile_id

Specifies the ID in the `authuser_profile` table that is associated with a particular language proficiency.

code

The language code. Most codes are ISO 639-1 codes, with the addition of codes for simplified and traditional Chinese.

5.3.7 Columns in the teams_courseteam Table

This table stores information about the teams in a course.

Note: The Teams feature is in limited release. For more information, contact your edX partner manager. For Open edX sites, contact your system administrator.

History: Added September 15 2015

The `teams_courseteam` table has the following columns.

Column	Type	Null	Key
id	int(11)	NO	PRI
team_id	varchar(255)	NO	UNI
name	varchar(255)	NO	UNI
course_id	textfield	NO	MUL
topic_id	varchar(255)	YES	MUL
date_created	datetime	NO	MUL
description	varchar(300)	NO	MUL
country	varchar(2)	YES	MUL
language	varchar(16)	YES	MUL
discussion_topic_id	varchar(255)	NO	MUL
last_activity_at	datetime	NO	MUL
team_size	int(11)	NO	MUL

id

The primary key, a database auto-increment field that uniquely identifies the team.

team_id

The unique identifier for this team.

name

The display name for this team. A name is required when a team is created.

course_id

The course identifier, in the format `{key type}:{org}+{course}+{run}`. For example, `course-v1:edX+DemoX+Demo_2014`.

History: In October 2014, identifiers for some new courses began to use the format shown above. Other new courses, and all courses created prior to October 2014, use the format `{org}/{course}/{run}`, for example, `MITx/6.002x/2012_Fall`.

topic_id

The unique identifier for the teams topic associated with the team. Topics, including an ID for each topic, are defined by course team members in **Advanced Settings** in Studio.

date_created

The date and time that this team was created, in the format `YYYY-MM-DD HH:MM:SS`.

description

The description for the team. A team description is required when a team is created.

country

An optional field in a team's details. The person who creates a team can specify a country that the team's members primarily identify with. Country codes are ISO 3166-1 codes.

language

An optional field in a team's details. A team can specify a language that the team's members primarily communicate using. Most language codes are ISO 639-1 codes, with the addition of codes for simplified and traditional Chinese.

discussion_topic_id

The identifier for all discussion topics within this team's discussions.

last_activity_at

The date and time that the most recent activity on the team was recorded, in the format `YYYY-MM-DD HH:MM:SS`. The current definition of activity for this field includes team creation, and the creation of posts, comments, and responses in the team's discussions.

team_size

The current count of the number of members in the team.

5.3.8 Columns in the teams_courseteammembership Table

This table stores information about learners who are members of a team.

Note: The Teams feature is in limited release. For more information, contact your edX partner manager. For Open edX sites, contact your system administrator.

History: Added September 15 2015.

The `teams_courseteammembership` table has the following columns.

Column	Type	Null	Key
id	int (11)	NO	PRI
user_id	int (11)	NO	UNI
team_id	int (11)	NO	MUL
date_joined	datetime	NO	MUL
last_activity_at	datetime	NO	MUL

id

The primary key, a database auto-increment field that uniquely identifies the membership of a user on a team.

user_id

The ID of a user who is currently a member of the team, from `auth_user.id`.

team_id

The ID of the team, from `teams_courseteam.id`.

date_joined

The timestamp of the time that the user joined the team, in the format `YYYY-MM-DD HH:MM:SS`.

last_activity_at

The date/time of the most recent activity performed by this user on this team, in the format `YYYY-MM-DD HH:MM:SS`. The current definition of activity for this field is limited to discussions-related actions by this user: adding or deleting posts, adding comments or responses, and voting on posts. If the user has not yet participated in the team's discussion, the `last_activity_at` date/time reflects the timestamp when the user joined the team.

5.4 Courseware Progress Data

Any piece of content in the courseware can store state and score in the `courseware_studentmodule` table. Grades and the user Progress page are generated by doing a walk of the course contents, searching for graded items, looking up a student's entries for those items in `courseware_studentmodule` via *(course_id, student_id, module_id)*, and then applying the grade weighting found in the course policy and grading policy files. Course policy files determine how much weight one problem has relative to another, and grading policy files determine how much categories of problems are weighted (for example, HW=50%, Final=25%, etc.).

5.4.1 About Modules

Modules can store state, but whether and how they do so varies based on the implementation for that particular kind of module. When a user loads a page, the system looks up all the modules that need to be rendered in order to display it, and then asks the database to look up state for those modules for that user. If there is no corresponding entry for that user for a given module, a new row is created and the state is set to an empty JSON object.

5.4.2 Columns in the courseware_studentmodule Table

The `courseware_studentmodule` table holds all courseware state for a given user.

A sample of the heading row and a data row in the `courseware_studentmodule` table follow.

```
id  module_type module_id student_id  state grade created modified  max_grade done
course_id

33973858  course  i4x://edX/DemoX/course/Demo_course  96452 {"position": 3} NULL
2013-03-19 17:21:07 2014-01-07 20:18:54 NULL  na  edX/DemoX/Demo_course
```

Students have a separate row for every piece of content that they access or that is created to hold state data, making this the largest table in the data package.

The `courseware_studentmodule` table has the following columns.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
module_type	varchar(32)	NO	MUL	problem	
module_id	varchar(255)	NO	MUL	NULL	
student_id	int(11)	NO	MUL	NULL	
state	longtext	YES		NULL	
grade	double	YES	MUL	NULL	
created	datetime	NO	MUL	NULL	
modified	datetime	NO	MUL	NULL	
max_grade	double	YES		NULL	
done	varchar(8)	NO	MUL	NULL	
course_id	varchar(255)	NO	MUL	NULL	

id

Primary key. Rarely used though, since most lookups on this table are searches on the three tuple of *(course_id, student_id, module_id)*.

module_type

Type	Description
chapter	The top level categories for a course. Each of these is usually labeled as a Week in the courseware, but this is just convention.
combinedopenended	A module type developed for grading open ended questions via self assessment, peer assessment, and machine learning.
conditional	Allows you to prevent access to certain parts of the courseware if other parts have not been completed first.
course	The top level course module of which all course content is descended.
crowdsource_hinter	Not currently used. **History**: This `module_type` was included in a single course on a test basis and then deprecated.
lti	Learning Tools Interoperability component that adds an external learning application to display content, or to display content and also require a student response.
peergrading	Indicates a problem that is graded by other students. An option for grading open ended questions.
poll_question	Not currently used. **History**: This `module_type` was included in a single course on a test basis and then deprecated.
problem	A problem that the user can submit solutions for. EdX offers many different varieties.
problemset	A collection of problems and supplementary materials, typically used for homeworks and rendered as a horizontal icon bar in the courseware. Use is inconsistent, and some courses use a `sequential` instead.
randomize	Identifies a module in which one of several possible defined alternatives is randomly selected for display to each student.
selfassessment	Self assessment problems. Used in a single course in Fall 2012 as an early test of the open ended grading system. Deprecated in favor of `combinedopenended`.
sequential	A collection of videos, problems, and other materials, rendered as a horizontal icon bar in the courseware.
timelimit	Not currently used. **History**: This `module_type` was included in a single course on a test basis and then deprecated.
video	A component that makes a video file available for students to play.
videoalpha	Not currently used. **History**: During the implementation of a change to the `video` `module_type`, both `video` and `videoalpha` were stored. The `videoalpha` type was then deprecated.
videosequence	A collection of videos, exercise problems, and other materials, rendered as a horizontal icon bar in the courseware. **History**: This `module_type` is no longer in use, courses now use `sequential` instead.
word_cloud	A specialized problem that produces a graphic from the words that students enter.

module_id

Unique ID for a distinct piece of content in a course. Each `module_id` is recorded as a URL with the format `{key type}:{org}+{course}+{run}@{module type}+block@{module name or hash code}`. Having URLs of this form gives content a canonical representation even during a transition between back-end data stores.

As an example, this example `module_id` contains the following parts.

```
block-v1:edX+DemoX+Demo_2014+type@problem+block@303034da25524878a2e66fb57c91cf85
```

Part	Example Value	Definition
`{key type}`	`block-v1`	The type of namespace identifier, including the implementation version.
`{org}`	`edX`	The organization part of the ID, indicating what organization created this piece of content.
`{course}`	`DemoX`	The course that this content was created for.
`{run}`	`Demo_2014`	The term or specific iteration of the course.
`type@{module type}`	`type@problem`	The module type. The same value is stored in the `courseware_studentmodule.module_type` column.
`block@{module name or hash code}`	`block@303034da25524878a2e66fb57c91cf85`	The name that the content creators supplied for this module. If the module does not have a name, the system generates a hash code as its identifier.

History: In October 2014, identifiers for modules in some new courses began to use the format shown above. Other new courses, and all courses created prior to October 2014, use the format `i4x://{org}/{course}/{module type}/{module name or hash code}`. For example, `i4x://MITx/3.091x/problemset/Sample_Problems`. Note that this format does not include course run information, so the `courseware_studentmodule.course_id` column may need to be used as well.

student_id

A reference to `auth_user.id`, this is the student that this module state row belongs to.

state

This is a JSON text field where different module types are free to store their state however they wish.

`course`, `chapter`, `problemset`, `sequential`, `videosequence`

The state for all of these container modules is a JSON object indicating the user's last known position within this container. This is 1-indexed, not 0-indexed, mostly because it was released that way and a later change would have broken saved navigation state for users.

Example: `{"position" : 3}`

When this user last interacted with this course/chapter/etc., she clicked on the third child element. Note that the position is a simple index and not a `module_id`, so if you rearranged the order of the contents, it would not be smart enough to accommodate the changes and would point users to the wrong place.

The hierarchy of these containers is `course > chapter > (problemset | sequential | videosequence)`.

`combinedopenended`

The JSON document includes attributes that identify the student's `answer`, a `rubric_xml` that includes the complete XML syntax for the rubric, the `score` earned and the `max_score`, and the `grader_id` (the `auth_user.id`) of each student who assessed the answer.

grade

Floating point value indicating the total unweighted grade for this problem that the student has scored. Basically how many responses they got right within the problem.

Only `problem` and `selfassessment` types use this column. All other modules set this to NULL. Due to a quirk in how rendering is done, `grade` can also be NULL for a tenth of a second or so the first time that a user loads a problem. The initial load triggers two writes, the first of which sets the `grade` to NULL, and the second of which sets it to 0.

created

Datetime when this row was created, which is typically when the student first accesses this piece of content.

Note: For a module that contains multiple child modules, a row is created for each of them when the student first accesses one of them.

---

modified

Datetime when this row was last updated. Set to be equal to `created` at first. A change in `modified` implies that there was a state change, usually in response to a user action like saving or submitting a problem, or clicking on a navigational element that records its state. However it can also be triggered if the module writes multiple times on its first load, like problems do (see note in `grade`).

max_grade

Floating point value indicating the total possible unweighted grade for this problem, or basically the number of responses that are in this problem. Though in practice it's the same for every entry with the same `module_id`, it is technically possible for it to be anything.

Another way in which `max_grade` can differ between entries with the same `module_id` is if the problem was modified after the `max_grade` was written and the user never went back to the problem after it was updated. This might happen if a member of the course team puts out a problem with five parts, realizes that the last part doesn't make sense, and decides to remove it. People who saw and answered it when it had five parts and never came back to it after the changes had been made will have a `max_grade` of 5, while people who saw it later will have a `max_grade` of 4.

Only graded module types use this column, with `problem` being the primary example. All other modules set this to NULL.

done

Not used. The value `na` appears in every row.

course_id

The course that this row applies to, in the format `{key type}:{org}+{course}+{run}`. For example, `course-v1:edX+DemoX+Demo_2014`.

Because the same course content (content with the same `module_id`) can be used in different courses, student state is tracked separately for each course.

History: In October 2014, identifiers for some new courses began to use the format shown above. Other new courses, and all courses created prior to October 2014, use the format `{org}/{course}/{run}`, for example, `MITx/6.002x/2012_Fall`.

5.5 Certificate Data

5.5.1 Columns in the certificates_generatedcertificate Table

The `certificates_generatedcertificate` table tracks the state of certificates and final grades for a course. The table is populated when a script is run to grade all of the students who are enrolled in the course at that time and issue certificates. The certificate process can be rerun and this table is updated appropriately.

A sample of the heading row and two data rows in the `certificates_generatedcertificate` table follow.

```
id  user_id  download_url  grade  course_id  key  distinction  status  verify_uuid
download_uuid  name  created_date  modified_date error_reason  mode

26  9999999
https://s3.amazonaws.com/verify.edx.org/downloads/9_hash_1/Certificate.pdf
0.84  BerkeleyX/CS169.1x/2012_Fall  f_hash_a  0  downloadable  2_hash_f
9_hash_1  AAAAAA  2012-11-10  00:12:11  2012-11-10  00:12:13  honor

27  9999999  0.0  BerkeleyX/CS169.1x/2012_Fall  0  notpassing  AAAAAA
2012-11-10  00:12:11  2012-11-26  19:06:19  honor
```

The `certificates_generatedcertificate` table has the following columns.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	NO	MUL	NULL	
download_url	varchar(128)	NO		NULL	
grade	varchar(5)	NO		NULL	
course_id	varchar(255)	NO	MUL	NULL	
key	varchar(32)	NO		NULL	
distinction	tinyint(1)	NO		NULL	
status	varchar(32)	NO		NULL	
verify_uuid	varchar(32)	NO		NULL	
download_uuid	varchar(32)	NO		NULL	
name	varchar(255)	NO		NULL	
created_date	datetime	NO		NULL	
modified_date	datetime	NO		NULL	
error_reason	varchar(512)	NO		NULL	
mode	varchar(32)	NO		NULL	

id

The primary key.

user_id, course_id

The table is indexed by user and course.

download_url

The `download_url` contains the full URL to the certificate.

grade

The grade computed the last time certificate generation ran. If the courseware, student state, or grading policy change, the value in this column can be different than the grade shown on a student's Progress page.

key

Used internally only. A random string that is used to match server requests to responses sent to the LMS.

distinction

Not used.

History: This was used for letters of distinction for 188.1x, but is not being used for any current courses.

status

The status can be one of these states.

Value	Description
deleted	The certificate has been deleted.
deleting	A request has been made to delete a certificate.
downloadable	The student passed the course and a certificate is available for download.
error	An error ocurred during certificate generation.
generating	A request has been made to generate a certificate but it has not yet been generated.
notpassing	The student's grade is not a passing grade.
regenerating	A request has been made to regenerate a certificate but it has not yet been generated.
restricted	No longer used. **History**: Specified when `userprofile.allow_certificate` was set to false: to indicate that the student was on the restricted embargo list.
unavailable	No entry, typically because the student has not yet been graded for certificate generation.

After a course has been graded and certificates have been issued, status is one of these values.

- downloadable
- notpassing

verify_uuid

A hash code that verifies the validity of a certificate. Included on the certificate itself as part of a URL.

download_uuid

A hash code that identifies this student's certificate. Included as part of the `download_url`.

name

This column records the name of the student that was set at the time the student was graded and the certificate was generated.

created_date

Date this row in the database was created.

modified_date

Date this row in the database was modified.

error_reason

Used internally only. Logs messages that are used for debugging if the certificate generation process fails.

mode

Contains the value found in the `enrollment.mode` field for a student and course at the time the certificate was generated: blank, audit, honor, or verified. This value is not updated if the student's `enrollment.mode` changes after certificates are generated.

CHAPTER 6

Course Content Data

For each course, the database files include a `{org}-{course}-{date}-course_structure-{site}-analytics.json` JSON file. Researchers can use this file to gain an overview of a course's content and investigate course state at a point in time.

This section describes the contents of the `course_structure` file.

6.1 Shared Fields

The following fields are present for all of the objects in the `course_structure` file.

- `category`
- `children`
- `metadata`

Descriptions of these fields follow.

6.1.1 Course Structure `category` Field

In the `course_structure` JSON file, the root level `category` field identifies core structural elements of a course. Each `course_structure` file contains a single `"category": "course"` object, and one or more of the objects for each of the other categories. For each object in the file, the `category` field contains one of the following strings.

Category Value	Description
chapter	The sections defined in the course outline. For more information, see *Course Building Block Data*.
course	The dates, settings, and other metadata defined for the course as a whole. For more information, see *Course Data*.
discussion	The content-specific discussion components defined for the course. For more information, see *Course Component Data*.
html	The HTML components defined for the course. For more information, see *Course Component Data*.
problem	The problem components defined for the course. For more information, see *Course Component Data*.
sequential	The subsections defined in the course outline. For more information, see *Course Building Block Data*.
vertical	The units defined in the course outline. For more information, see *Course Building Block Data*.
video	The video components defined for the course. For more information, see *Course Component Data*.

The `category` file can also contain additional values. These values, such as `poll_question`, identify optional components. These components add different types of exercises or tools to the course.

6.1.2 Course Structure `children` Field

The `children` field is an array. It identifies the modules that a specific structural element in the course contains. The `children` field for the `"category": "course"` object lists all of the sections in the course. For one of the `"category": "vertical"` objects, this field lists all of the HTML, discussion, problem, and video components in that unit.

6.1.3 Course Structure `metadata` Field

The `metadata` field is an object. This field contains key-value pairs that describe the settings defined for the course and for each of the modules that it contains.

6.2 Course Data

In the `"category": "course"` object, the `children` field lists all of the sections defined for the course. In the sample that follows, the edX DemoX course has five sections, or chapters, defined.

The `metadata` field provides information about parameters set for the course, including dates, pages, textbooks, and advanced setting values. In the sample that follows, note that the edX DemoX course includes a course-specific page, or tab, named "edX Community".

A partial list of the `metadata` member fields for a course follows. For information about the settings that course teams define for a course in Studio, see Building and Running an edX Course.

metadata Member Field	Description
advanced_modules	This array stores values entered for **Advanced Module List** on the Studio **Advanced Settings** page.
days_early_for_beta	This field stores the number entered for **Days Early for Beta Users** on the Studio **Advanced Settings** page.
discussion_topics	This object lists the course-wide discussion topics entered for **Discussion Topic Mapping** on the Studio **Advanced Settings** page.
showanswer	This field stores the string entered for **Show Answer** on the Studio **Advanced Settings** page. Valid values are 'always', 'answered', 'attempted', 'closed', 'finished', 'past_due', or 'never'.
start	This field stores the value entered for **Course Start Date** on the Studio **Settings & Details** page.
tabs	This array contains member objects that describe the tabs, or pages, that appear for the course in the learning management system (LMS). Course teams can rename most of the default pages, and add more pages, on the Studio **Pages** page. The default **Courseware** page cannot be renamed. This page uses the structure defined by the course building blocks to deliver the content defined by the course components.

6.2.1 Course Data Sample


```
"i4x://edX/DemoX/course/1T2015": {
   "category": "course",
   "children": [
     "i4x://edX/DemoX/chapter/1ff96c6155eb40c39140c656cdc2708b",
     "i4x://edX/DemoX/chapter/00d4374f346b4744aa6f4708cdf46d53",
     "i4x://edX/DemoX/chapter/abc5cf5203ee494faf73fa3f55b4485b",
     "i4x://edX/DemoX/chapter/a783b6e59fe24917985a8aa29eeec150",
     "i4x://edX/DemoX/chapter/0cdd0de7b1f740468381c265796f6f63"
   ],
   "metadata": {
     "advertised_start": "4/15/2015",
     "days_early_for_beta": 90.0,
     "discussion_topics": {
       "General": {
         "id": "i4x-edX-DemoX-course-1T2015"
       }
     },
     "display_name": "edX Demonstration Course",
     "end": null,
     "graceperiod": "18000 seconds",
     "start": "2014-08-10T07:00:00Z",
     "tabs": [
       {
         "name": "Courseware",
         "type": "courseware"
       },
       {
         "name": "Course Info",
         "type": "course_info"
       },
       {
         "name": "Discussion",
         "type": "discussion"
       },
       {
         "name": "edX Community",
```

```
          "type": "static_tab",
          "url_slug": "67e8a9e44dde4e97b2bd33a928b9099e"
        }
        {
          "name": "Progress",
          "type": "progress"
        },
        {
          "name": "Wiki",
          "type": "wiki"
        }
      ]
    }
  },
```

6.3 Course Building Block Data

In Studio, a course team organizes course content by defining hierarchical sections, subsections, and units. Internally, the edX code identifies these building blocks with a `category` value of 'chapter', 'sequential', and 'vertical'.

The sample that follows extracts the objects that represent one of the sections in a course, a subsection that the section contains, and a unit that the subsection contains from a JSON `course_structure` document.

The `children` array for each of these types of objects lists identifiers for objects that it contains.

- Objects with a category of `chapter` list all of the sequentials (subsections) that they contain.
- Objects with a category of `sequential` list all of the verticals (units) that they contain.
- Objects with a category of `vertical` list all of the components that they contain.

The `metadata` field provides information about parameters set for the section, subsection, or unit. A partial list of the `metadata` member fields for a section, subsection, or unit follows. For information about the structure that course teams can define for a course, see Developing Your Course in the *Building and Running an edX Course* guide.

`metadata` Member Field	Description
`display_name`	This field stores the string in the name field for the section, subsection, or unit on the Studio **Course Outline** page. Course teams can edit the default name that Studio supplies. This name identifies this structural element to learners in the LMS and in edX Insights.
`start`	This field stores the value entered for the section, subsection, or unit on the Studio **Course Outline** page. Course teams provide these optional start dates so that course content is released incrementally after the course start date.
`visible_to_staff_only`	This Boolean indicates the setting selected for the **Hide from Students** option for the section, subsection, or unit on the Studio **Course Outline** page.

6.3.1 Course Building Block Data Sample

```
"i4x://edX/DemoX/chapter/00d4374f346b4744aa6f4708cdf46d53": {
  "category": "chapter",
  "children": [
    "i4x://edX/DemoX/sequential/9681154b9c0a4baaafb5f4e26bc71550"
  ],
  "metadata": {
    "display_name": "Introduction to edX Studio",
```

```
    "start": "2020-08-09T16:00:00Z",
    "visible_to_staff_only": true
  }
},
.
.
.
"i4x://edX/DemoX/sequential/547f430ffd414a5fbb4a080fd5eb7566": {
  "category": "sequential",
  "children": [
    "i4x://edX/DemoX/vertical/2ea89cbec5bd4034981a70abff7a82e1",
    "i4x://edX/DemoX/vertical/7405431e9fe14354a39ac52a2973bc1c"
  ],
  "metadata": {
    "display_name": "What Does an edX Course Look Like?"
  }
},
.
.
.
 "i4x://edX/DemoX/vertical/7405431e9fe14354a39ac52a2973bc1c": {
  "category": "vertical",
  "children": [
    "i4x://edX/DemoX/html/d3bd5215cf044056beb8e6f7f3e3afc4",
    "i4x://edX/DemoX/video/ddf62dd7bff249efa1add6776f1e2ab8"
  ],
  "metadata": {
    "display_name": "Your Course Info"
  }
},
.
.
.
"i4x://edX/DemoX/vertical/778671e308e446409c0c797d9d424eae": {
  "category": "vertical",
  "children": [
    "i4x://edX/DemoX/problem/db71da27320a44bdb45df31d0d801e20",
    "i4x://edX/DemoX/discussion/05d808aad49543de997964be3bfac528"
  ],
  "metadata": {
    "display_name": "Exercise Gallery"
  }
```


6.4 Course Component Data

In Studio, a course team specifies course content by adding components to units. The core, or basic, types of components that course teams can add have a `category` value of 'discussion', 'html', 'problem', and 'video'. The sample that follows extracts objects for each of these component types from a `course_structure` file.

The `children` array is not used for these types of objects.

The `metadata` field provides information about parameters set for a component.

- The `metadata.display_name` member field applies to all of the basic component types. This name identifies the component to learners in the LMS and in edX Insights. Course teams can edit the default name in the Studio **Settings** dialog box.

- The other `metadata` member fields reflect settings specific to each component type. For information about the settings that course teams can define for components, see Adding Course Components in the *Building and Running an edX Course* guide.

6.4.1 Course Component Data Sample


```
"i4x://edX/DemoX/html/d3bd5215cf044056beb8e6f7f3e3afc4": {
  "category": "html",
  "children": [],
  "metadata": {
    "display_name": "Intro to Video"
  }
},
.
.
.
"i4x://edX/DemoX/video/ddf62dd7bff249efa1add6776f1e2ab8": {
  "category": "video",
  "children": [],
  "metadata": {
    "display_name": "Your Course About Page",
    "download_track": true,
    "download_video": true,
    "end_time": "00:07:24",
    "html5_sources": [
      "https://d2f1egay8yehza.cloudfront.net/BERGG101/BERGG101T314-V001800_100.mp4"
    ],
    "sub": "BERGG101T314-V001800_100",
    "youtube_id_1_0": "uxypPaUu8ng"
  }
},
.
.
.
  "i4x://edX/DemoX/problem/db71da27320a44bdb45df31d0d801e20": {
  "category": "problem",
  "children": [],
  "metadata": {
    "display_name": "Multiple Choice Questions",
    "markdown": "Many edX courses have homework or exercises you need to complete.  Notice the clock
    "max_attempts": null,
    "rerandomize": "never",
    "showanswer": "never",
    "weight": null
  }
},
.
.
.
"i4x://edX/DemoX/discussion/05d808aad49543de997964be3bfac528": {
  "category": "discussion",
  "children": [],
  "metadata": {
    "discussion_category": "Week 2",
    "discussion_id": "7be676c36bba4486aeeabe3ecb5b06e8",
    "discussion_target": "Improve the Question",
    "display_name": "Discussion Space: Improve the Question"
```

```
    }
},
```

CHAPTER 7

Discussion Forums Data

EdX discussion data is stored as collections of JSON documents in a MongoDB database. MongoDB is a document-oriented, NoSQL database system. Documentation can be found at the mongodb web site.

In the data package, discussion data is delivered in a .mongo file, identified by organization and course, in this format: edX-*organization-course-source*.mongo.

The primary collection that holds all of the discussion posts written by users is "contents". Two different types of objects are stored, representing the three levels of interactions that users can have in a discussion.

- A `CommentThread` represents the first level of interaction: a post that opens a new thread, often a student question of some sort.
- A `Comment` represents both the second and third levels of interaction: a response made directly to the conversation started by a `CommentThread` is a `Comment`. Any further contributions made to a specific response are also in `Comment` objects.

A sample of the field/value pairs that are in the mongo file, and descriptions of the attributes that these two types of objects share and that are specific to each type, follow.

In addition to these collections, events are also emitted to track specific user activities. For more information, see *Discussion Forum Events*.

7.1 Samples

Two sample rows, or JSON documents, from a `.mongo` file of discussion data follow.

7.1.1 CommentThread Document Example

The JSON documents that include discussion data are delivered in a compact, machine-readable format that can be difficult to read at a glance.

```
{ "_id" : { "$oid" : "50f1dd4ae05f6d2600000001" }, "_type" : "CommentThread",
"anonymous" :false, "anonymous_to_peers" : false, "at_position_list" : [],
"author_id" : "NNNNNNN","author_username" : "AAAAAAAAAA", "body" : "Welcome to
the edX101 forum!\n\nThis forum willbe regularly monitored by edX. Please post
your questions and comments here. When asking aquestion, don't forget to
search the forum to check whether your question has already
beenanswered.\n\n", "closed" : false, "comment_count" : 0, "commentable_id" :
"i4x-edX-edX101-course-How_to_Create_an_edX_Course", "course_id" :
"edX/edX101/How_to_Create_an_edX_Course","created_at" : { "$date" :
1358028106904 }, "last_activity_at" : { "$date" : 1358134464424 },"tags_array"
```

```
: [], "thread_type": "discussion", "title" : "Welcome to the edX101 forum!",
"updated_at" : { "$date" :1358134453862 }, "votes" : { "count" : 1, "down" :
[], "down_count" : 0, "point" : 1, "up" :[ "48" ], "up_count" : 1 } }
```

If you use a JSON formatter to "pretty print" this document, a version that is more readable is produced.


```
{
 "_id": {
   "$oid": "50f1dd4ae05f6d2600000001"
 },
 "_type": "CommentThread",
 "anonymous": false,
 "anonymous_to_peers": false,
 "at_position_list": [

 ],
 "author_id": "NNNNNNN",
 "author_username": "AAAAAAAAAA",
 "body": "Welcome to the edX101 forum!\n\nThis forum will be regularly
 monitored by edX. Please post your questions and comments here. When
 asking a question, don't forget to search the forum to check whether
 your question has already been answered.\n\n",
 "closed": false,
 "comment_count": 0,
 "commentable_id": "i4x-edX-edX101-course-How_to_Create_an_edX_Course",
 "course_id": "edX\/edX101\/How_to_Create_an_edX_Course",
 "created_at": {
   "$date": 1358028106904
 },
 "last_activity_at": {
   "$date": 1358134464424
 },
 "tags_array": [

 ],
 "thread_type": "discussion",
 "title": "Welcome to the edX101 forum!",
 "updated_at": {
   "$date": 1358134453862
 },
 "votes": {
   "count": 1,
   "down": [

   ],
   "down_count": 0,
   "point": 1,
   "up": [
     "48"
   ],
   "up_count": 1
 }
}
```

7.1.2 Comment Document Example


```
{ "_id" : { "$oid" : "52e54fdd801eb74c33000070" }, "votes" : { "up" : [],
"down" : [], "up_count" : 0, "down_count" : 0, "count" : 0, "point" : 0 },
"visible" : true, "abuse_flaggers" : [], "historical_abuse_flaggers" : [],
"parent_ids" : [], "at_position_list" : [], "body" : "I'm hoping this
Demonstration course will help me figure out how to take the course I enrolled
in. I am just auditing the course, but I want to benefit from it as much as
possible, as I am extremely interested in it.\n", "course_id" :
"edX/DemoX/Demo_Course", "_type" : "Comment", "endorsed" : true, "endorsement"
: { "user_id" : "9", "time" : ISODate("2014-08-29T15:11:49.442Z") },
"anonymous" : false, "anonymous_to_peers" : false, "author_id" : "NNNNNNN",
"comment_thread_id" : { "$oid" : "52e4e880c0df1fa59600004d" },
"author_username" : "AAAAAAAAAA", "sk" : "52e54fdd801eb74c33000070",
updated_at" : { "$date" : 1390759901966 }, "created_at" : { "$date" :
1390759901966 } }
```


When pretty printed, this comment looks like this:


```
{
 "_id": {
   "$oid": "52e54fdd801eb74c33000070"
 },
 "votes": {
   "up": [

   ],
   "down": [

   ],
   "up_count": 0,
   "down_count": 0,
   "count": 0,
   "point": 0
 },
 "visible": true,
 "abuse_flaggers": [

 ],
 "historical_abuse_flaggers": [

 ],
 "parent_ids": [

 ],
 "at_position_list": [

 ],
 "body": "I'm hoping this Demonstration course will help me figure out how
 to take the course I enrolled in. I am just auditing the course, but I
 want to benefit from it as much as possible, as I am extremely interested
 in it.\n",
 "course_id": "edX\/DemoX\/Demo_Course",
 "_type": "Comment",
 "endorsed": true,
 "endorsement": {
   "user_id": "9",
   "time": {
     "$date": 1390759911966
```

```
    }
  }
  "anonymous": false,
  "anonymous_to_peers": false,
  "author_id": "NNNNNNN",
  "comment_thread_id": {
    "$oid": "52e4e880c0df1fa59600004d"
  },
  "author_username": "AAAAAAAAAA",
  "sk": "52e54fdd801eb74c33000070",
  "updated_at": {
    "$date": 1390759901966
  },
  "created_at": {
    "$date": 1390759901966
  }
}
```

7.2 Shared Fields

Descriptions of the fields that are present for both `CommentThread` and `Comment` objects follow.

7.2.1 _id

The 12-byte MongoDB unique ID for this collection. Like all MongoDB IDs, the IDs are monotonically increasing and the first four bytes are a timestamp.

7.2.2 _type

`CommentThread` or `Comment` depending on the type of object.

7.2.3 anonymous

If true, this `CommentThread` or `Comment` displays in the user interface as written by "anonymous", even to course team members and discussion team members.

7.2.4 anonymous_to_peers

If true, this `CommentThread` or `Comment` displays in the user interface as written by "anonymous" to students, but members of the course team and the discussion team can see the author's username.

7.2.5 at_position_list

No longer used. Child comments (replies) are sorted by their `created_at` timestamp only.

7.2.6 author_id

Identifies the user who wrote this. Corresponds to the user IDs stored in the MySQL database as `auth_user.id`.

7.2.7 author_username

The username of the person who wrote the discussion post or comment.

7.2.8 body

Text of the comment in Markdown. UTF-8 encoded.

7.2.9 course_id

The full course_id of the course that this comment was made in, including org and run. This value can be seen in the URL when browsing the courseware section. Example: `BerkeleyX/Stat2.1x/2013_Spring`.

7.2.10 created_at

Timestamp in UTC. Example: `ISODate("2013-02-21T03:03:04.587Z")`.

7.2.11 updated_at

Timestamp in UTC. Example: `ISODate("2013-02-21T03:03:04.587Z")`.

7.2.12 votes

Both `CommentThread` and `Comment` objects support voting. In the user interface, students can vote for posts (`CommentThread` objects) and for responses, but not for the third-level comments made on responses. All `Comment` objects still have this attribute, even though there is no way to actually vote on the comment-level items in the UI. This attribute is an object that has the following items inside.

- up = list of User IDs that up-voted this comment or thread.
- down = (no longer used) list of User IDs that down-voted this comment or thread.
- up_count = total upvotes received.
- down_count = No longer used. Total downvotes received.
- count = total votes cast.
- point = net vote, now always equal to up_count.

A user only has one vote per `Comment` or `CommentThread`. Though it's still written to the database, the UI no longer displays an option to downvote anything.

7.3 CommentThread Fields

The following fields are specific to `CommentThread` objects. Each thread in the discussion forums is represented by one `CommentThread`.

7.3.1 closed

If true, this thread was closed by a discussion forum moderator or admin.

7.3.2 comment_count

The number of comment replies in this thread. This includes all responses and replies, but does not include the original post that started the thread. So for this exchange:

```
CommentThread: "What's a good breakfast?"
  * Comment: "Just eat cereal!"
  * Comment: "Try a Loco Moco, it's amazing!"
    * Comment: "A Loco Moco? Only if you want a heart attack!"
    * Comment: "But it's worth it! Just get a spam musubi on the side."
```

The `comment_count` for this `CommentThread` is **4**.

7.3.3 commentable_id

A course team can attach a discussion to any piece of content in the course, or to top level categories like "General" and "Troubleshooting". When the discussion is a top level category it is specified in the course's policy file, and the `commentable_id` is formatted like this: "i4x-edX-edX101-course-How_to_Create_an_edX_Course". When the discussion is a specific component in the course, the `commentable_id` identifies that component: "d9f970a42067413cbb633f81cfb12604".

7.3.4 last_activity_at

Timestamp in UTC indicating the last time there was activity in the thread (new posts, edits, etc). Closing the thread does not affect the value in this field.

7.3.5 tags_array

No longer used.

History: Intended to be a list of user definable tags.

7.3.6 title

Title of the thread. UTF-8 string.

7.3.7 thread_type

Identifies the type of post as a "question" or "discussion".

History: Added 4 Sep 2014.

7.4 Comment Fields

The following fields are specific to `Comment` objects. A `Comment` is either a response to a `CommentThread` (such as an answer to the question), or a reply to another `Comment` (a comment about somebody's answer).

History: It used to be the case that `Comment` replies could nest much more deeply, but this was later capped at just these three levels (post, response, comment) much in the way that StackOverflow does.

7.4.1 visible

Not used.

7.4.2 abuse_flaggers

Records the user id of each user who selects the **Report Misuse** flag for a `Comment` in the user interface. Stores an array of user ids if more than one user flags the `Comment`. This is empty if no users flag the `Comment`.

7.4.3 historical_abuse_flaggers

If a discussion moderator removes the **Report Misuse** flag from a `Comment`, all user IDs are removed from the `abuse_flaggers` field and then written to this field.

7.4.4 endorsed

Boolean value. True if a forum moderator has marked this response to a `CommentThread` with a `thread_type` of "discussion" as a valuable contribution, or if a forum moderator or the originator of a `CommentThread` with a `thread_type` of "question" has marked this response as the correct answer.

The `endorsed` field is present for comments that are made as replies to responses, but in these cases the value is always false: the user interface does not offer a way to endorse comments.

7.4.5 endorsement

Contains `time` and `user_id` fields for the date and time that this response to a post was endorsed and the numeric user ID (from `auth_user.id`) of the person who endorsed it.

History: Added 4 Sep 2014.

7.4.6 comment_thread_id

Identifies the `CommentThread` that the `Comment` is a part of.

7.4.7 parent_id

Applies only to comments made to a response. In the example given for `comment_count` above, "A Loco Moco? Only if you want a heart attack!" is a comment that was made to the response, "Try a Loco Moco, it's amazing!"

The `parent_id` is the `_id` of the response-level `Comment` that this `Comment` is a reply to. Note that this field is only present in a `Comment` that is a reply to another `Comment`; it does not appear in a `Comment` that is a reply to a `CommentThread`.

7.4.8 parent_ids

The `parent_ids` field appears in all `Comment` objects, and contains the `_id` of all ancestor comments. Since the UI now prevents comments from being nested more than one layer deep, it will only ever have at most one element in it. If a `Comment` has no parent, it is an empty list.

7.4.9 sk

A randomly generated number that drives a sorted index to improve online performance.

CHAPTER 8

Wiki Data

The following sections detail how edX stores wiki data internally, and is useful for developers and researchers who are examining database exports.

EdX currently uses an external application called django-wiki for wiki functionality within courses.

In the data package, wiki data is delivered in two SQL files:

- The wiki_article file is a container for each article that is added to the wiki. The full name of this file also includes the organization and course, and indicates a source of either prod (edX) or edge, in this format: edX-*organization-course*-wiki_article-*source*-analytics.sql.
- The wiki_articlerevision file stores data about the articles, including data about changes and deletions. The full name of this file is in this format: edX-*organization-course*-wiki_articlerevision-*source*-analytics.sql.

8.1 Fields in the wiki_article file

The header row of the wiki_article SQL file, and a row of sample data, follow.

```
id  current_revision_id created modified  owner_id  group_id  group_read  group_write
other_read  other_write

1437  29819 2013-07-17 21:53:57 2014-01-26 14:48:02 NULL  NULL  1 1 1 1
```

The table that follows provides a reference to each field in this file. A description of each field follows the table.

Field	Type	Null	Key
id	int(11)	NO	PRI
current_revision_id	int(11)	NO	UNI
created	datetime	NO	
modified	datetime	NO	
owner_id	int(11)	YES	MUL
group_id	int(11)	YES	MUL
group_read	tinyint(1)	NO	
group_write	tinyint(1)	NO	
other_read	tinyint(1)	NO	
other_write	tinyint(1)	NO	

8.1.1 id

The primary key.

8.1.2 current_revision_id

The ID of the revision that displays for this article.

8.1.3 created

The date the article was created.

8.1.4 modified

The date the article properties were last modified.

8.1.5 owner_id

The owner of the article, usually the creator. The owner always has both read and write access.

8.1.6 group_id

As in a UNIX file system, permissions can be given to a user according to group membership. Groups are handled through the Django authentication system.

8.1.7 group_read

Defines whether the group has read access to the article. 1 if so, 0 if not.

8.1.8 group_write

Defines whether the group has write access to the article. 1 if so, 0 if not.

8.1.9 other_read

Defines whether others have read access to the article. 1 if so, 0 if not.

8.1.10 other_write

Defines whether others have write access to the article. 1 if so, 0 if not.

8.2 Fields in the wiki_articlerevision file

The header row of the wiki_articlerevision SQL file, and a row of sample data, follow.

```
id  revision_number user_message  automatic_log ip_address  user_id modified  created
previous_revision_id  deleted locked  article_id  content title

17553 1 Course page automatically created.    NULL  NULL  2013-07-17 21:53:57 2013-07-17
21:53:57 NULL  0 0 1437  This is the wiki for edX's edX Demonstration Course.  DemoX
```

The table that follows provides a reference to the characteristics of each field in this file. Descriptions of the fields follow the table.

Field	Type	Null	Key
id	int(11)	NO	PRI
revision_number	int(11)	NO	
user_message	longtext	NO	
automatic_log	longtext	NO	
ip_address	char(15)	YES	
user_id	int(11)	YES	MUL
modified	datetime	NO	
created	datetime	NO	
previous_revision_id	int(11)	YES	MUL
deleted	tinyint(1)	NO	
locked	tinyint(1)	NO	
article_id	int(11)	NO	MUL
content	longtext	NO	
title	varchar(512)	NO	

8.2.1 id

The primary key.

8.2.2 revision_number

The ID of the revision.

8.2.3 user_message

The message the user added when saving the revision.

8.2.4 automatic_log

Some changes to wiki pages are logged to make the revision history for an article available in the user interface.

8.2.5 ip_address

The IP address of the device where the revision was made.

8.2.6 user_id

The ID of the user who made the revision.

8.2.7 modified

The date the article was last modified.

8.2.8 created

The date the article was created.

8.2.9 previous_revision_id

The ID of the revision previous to this one.

8.2.10 deleted

Defines whether the revision was deleted.

8.2.11 locked

Defines whether the revision is locked.

8.2.12 article_id

The ID of the revision that displays data for this article.

8.2.13 content

The content of the article revision.

8.2.14 title

The title of the article revision.

CHAPTER 9

Institution-wide Data

The data package includes a report of data collected across all of an institution's edx.org or edge.edx.org courses. This report is typically more useful to administrative or marketing teams, rather than research teams.

9.1 Email Opt In Report

When students enroll in a course on edx.org, they can specify whether they want to receive email from the organization that presents the course.

The `{org}-email_opt_in-{site}-analytics.csv` file reports the email preference selected by each student enrolled in your institution's courses. You can use this information to develop a distribution list for campaigns that introduce new or related courses to students.

Note: Your data package includes a .csv file for the edx.org site only. At this time, students can specify an email preference only on edx.org.

The file contains data in these columns.

```
email,full_name,course_id,is_opted_in_for_email,preference_set_datetime
```

9.1.1 email

The email address that the student used to register a user account on the site. For more information, see the `auth_user.email` *column*.

9.1.2 full_name

The name that the student supplied. For more information, see the `auth_userprofile.name` *column*.

9.1.3 course_id

The ID of the course run in which the student is enrolled. For more information, see the `student_courseenrollment.course_id` *column*.

9.1.4 is_opted_in_for_email

True or False. By default, this preference is set to True. If a student is enrolled in more than one course, the option that the student selected most recently applies to all of the courses.

9.1.5 preference_set_datetime

Indicates when the student selected this preference. If a student is enrolled in more than one of your institution's courses, the date and time when the student most recently selected an email preference applies to all of the courses.

CHAPTER 10

Alphabetical Event List

A, B, C - D, E, F - G, H, I - J, K, L - M, N, O - P, Q, R - S, T - U, V, W, X, Y, Z

10.1 A, B, C

Event	Description
add-forum-admin	*Course Team Events*
add-forum-community-TA	*Course Team Events*
add-forum-mod	*Course Team Events*
add-instructor	*Course Team Events*
add-or-remove-user-group	*Course Team Events*
book	*Textbook Interaction Events*

10.2 D, E, F

Event	Description
delete-student-module-state	*Course Team Events*
dump-answer-dist-csv	*Course Team Events*
dump-graded-assignments-config	*Course Team Events*
dump-grades	*Course Team Events*
dump-grades-csv	*Course Team Events*
dump-grades-csv-raw	*Course Team Events*
dump-grades-raw	*Course Team Events*
edx.certificate.created	*Certificate Events*
edx.certificate.shared	*Certificate Events*
edx.certificate.evidence_visited	*Certificate Events*
edx.cohort.created	*Student Cohort Events*
edx.cohort.creation_requested	*Course Team Cohort Events*
edx.cohort.user_add_requested	*Course Team Cohort Events*
edx.cohort.user_added	*Student Cohort Events*
edx.cohort.user_removed	*Student Cohort Events*
edx.course.enrollment.activated	*Enrollment Events* and *Instructor Enrollment Events*
edx.course.enrollment.deactivated	*Enrollment Events* and *Instructor Enrollment Events*
edx.course.enrollment.mode_changed	*Enrollment Events*

Continued on next page

Table 10.1 – continued from previous page

Event	Description
edx.course.enrollment.upgrade.clicked	*Enrollment Events*
edx.course.enrollment.upgrade.succeeded	*Enrollment Events*
edx.forum.comment.created	*Discussion Forum Events*
edx.forum.response.created	*Discussion Forum Events*
edx.forum.response.voted	*Discussion Forum Events*
edx.forum.searched	*Discussion Forum Events*
edx.forum.thread.created	*Discussion Forum Events*
edx.forum.thread.voted	*Discussion Forum Events*
edx.googlecomponent.calendar.displayed	*Third-Party Content Events*
edx.googlecomponent.document.displayed	*Third-Party Content Events*
edx.instructor.report.downloaded	*Course Team Events*
edx.instructor.report.requested	*Course Team Events*
edx.librarycontentblock.content.assigned	*Library Interaction Events*
edx.librarycontentblock.content.removed	*Library Interaction Events*
edx.problem.hint.demandhint_displayed	*Problem Interaction Events*
edx.problem.hint.feedback_displayed	*Problem Interaction Events*
edx.team.activity_updated	*Teams-Related Events*
edx.team.changed	*Teams-Related Events*
edx.team.created	*Teams-Related Events*
edx.team.deleted	*Teams-Related Events*
edx.team.learner_added	*Teams-Related Events*
edx.team.learner_removed	*Teams-Related Events*
edx.team.page_viewed	*Teams-Related Events*
edx.team.searched	*Teams-Related Events*
edx.video.bumper.dismissed	*Pre-Roll Video Interaction Events*
edx.video.bumper.loaded	*Pre-Roll Video Interaction Events*
edx.video.bumper.played	*Pre-Roll Video Interaction Events*
edx.video.bumper.skipped	*Pre-Roll Video Interaction Events*
edx.video.bumper.stopped	*Pre-Roll Video Interaction Events*
edx.video.bumper.transcript.hidden	*Pre-Roll Video Interaction Events*
edx.video.bumper.transcript.menu.hidden	*Pre-Roll Video Interaction Events*
edx.video.bumper.transcript.menu.shown	*Pre-Roll Video Interaction Events*
edx.video.bumper.transcript.shown	*Pre-Roll Video Interaction Events*
edx.video.loaded	*Video Interaction Events*, see load_video
edx.video.paused	*Video Interaction Events*, see pause_video
edx.video.played	*Video Interaction Events*, see play_video
edx.video.position.changed	*Video Interaction Events*, see seek_video
edx.video.stopped	*Video Interaction Events*, see stop_video
edx.video.transcript.hidden	*Video Interaction Events*, see hide_transcript
edx.video.transcript.shown	*Video Interaction Events*, see show_transcript

10.3 G, H, I

Event	Description
get-student-progress-page	*Course Team Events*
hide_transcript	*Video Interaction Events*

10.4 J, K, L

Event	Description
list-beta-testers	*Course Team Events*
list-forum-admins	*Course Team Events*
list-forum-community-TAs	*Course Team Events*
list-forum-mods	*Course Team Events*
list-instructors	*Course Team Events*
list-staff	*Course Team Events*
list-students	*Course Team Events*
load_video	*Video Interaction Events*

10.5 M, N, O

Event	Description
microsoft.office.mix.loaded	*Third-Party Content Events*
microsoft.office.mix.paused	*Third-Party Content Events*
microsoft.office.mix.played	*Third-Party Content Events*
microsoft.office.mix.slide.loaded	*Third-Party Content Events*
microsoft.office.mix.stopped	*Third-Party Content Events*
openassessmentblock.create_submission	*Open Response Assessment Events*
openassessmentblock.get_peer_submission	*Open Response Assessment Events*
openassessmentblock.peer_assess	*Open Response Assessment Events*
openassessmentblock.save_submission	*Open Response Assessment Events*
openassessmentblock.self_assess	*Open Response Assessment Events*
openassessmentblock.submit_feedback_on_assessments	*Open Response Assessment Events*
openassessment.student_training_assess_example	*Open Response Assessment Events*
openassessment.upload_file	*Open Response Assessment Events*
oppia.exploration.completed	*Third-Party Content Events*
oppia.exploration.loaded	*Third-Party Content Events*
oppia.exploration.state.changed	*Third-Party Content Events*

10.6 P, Q, R

Event	Description
page_close	*Navigational Events*
pause_video	*Video Interaction Events*
play_video	*Video Interaction Events*
problem_check	*Problem Interaction Events*
problem_check_fail	*Problem Interaction Events*
problem_graded	*Problem Interaction Events*
problem_rescore	*Problem Interaction Events*
problem_rescore_fail	*Problem Interaction Events*
problem_reset	*Problem Interaction Events*
problem_save	*Problem Interaction Events*
problem_show	*Problem Interaction Events*
remove-forum-admin	*Course Team Events*
remove-forum-community-TA	*Course Team Events*
remove-forum-mod	*Course Team Events*
remove-instructor	*Course Team Events*
rescore-all-submissions	*Course Team Events*
rescore-student-submission	*Course Team Events*
reset-all-attempts	*Course Team Events*
reset_problem	*Problem Interaction Events*
reset_problem_fail	*Problem Interaction Events*
reset-student-attempts	*Course Team Events*

10.7 S, T

Event	Description
save_problem_fail	*Problem Interaction Events*
save_problem_success	*Problem Interaction Events*
seek_video	*Video Interaction Events*
seq_goto	*Navigational Events*
seq_next	*Navigational Events*
seq_prev	*Navigational Events*
showanswer	*Problem Interaction Events*
show_transcript	*Video Interaction Events*
speed_change_video	*Video Interaction Events*
stop_video	*Video Interaction Events*
textbook.pdf.chapter.navigated	*Textbook Interaction Events*
textbook.pdf.display.scaled	*Textbook Interaction Events*
textbook.pdf.outline.toggled	*Textbook Interaction Events*
textbook.pdf.page.navigated	*Textbook Interaction Events*
textbook.pdf.page.scrolled	*Textbook Interaction Events*
textbook.pdf.searchcasesensitivity.toggled	*Textbook Interaction Events*
textbook.pdf.search.executed	*Textbook Interaction Events*
textbook.pdf.search.highlight.toggled	*Textbook Interaction Events*
textbook.pdf.search.navigatednext	*Textbook Interaction Events*
textbook.pdf.thumbnail.navigated	*Textbook Interaction Events*
textbook.pdf.thumbnails.toggled	*Textbook Interaction Events*
textbook.pdf.zoom.buttons.changed	*Textbook Interaction Events*
textbook.pdf.zoom.menu.changed	*Textbook Interaction Events*

10.8 U, V, W, X, Y, Z

Event	Description
video_hide_cc_menu	*Video Interaction Events*
video_show_cc_menu	*Video Interaction Events*
xblock.poll.submitted	*Poll and Survey Events*
xblock.poll.view_results	*Poll and Survey Events*
xblock.split_test.child_render	*Testing Events for Content Experiments*
xblock.survey.submitted	*Poll and Survey Events*
xblock.survey.view_results	*Poll and Survey Events*
xmodule.partitions.assigned_user_to_partition	*Testing Events for Content Experiments*

CHAPTER 11

Events in the Tracking Logs

This section provides reference information about the event data that is delivered in data packages. Events are emitted by the server, the browser, or the mobile device to capture information about interactions with the courseware and the Instructor Dashboard in the LMS, and are stored in JSON documents. In the data package, event data is delivered in a log file.

The student and course team events are grouped into categories in this section. For a complete, alphabetical list of events, see the *Alphabetical Event List*.

11.1 Reviewing a Sample Event

A sample event from an edX.log file follows. This sample was edited to remove personally identifiable information. Events are stored in JSON documents, which can be difficult to read before standard formatting is applied.


```
{"agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/30.0.1599.101 Safari/537.36", "context": {"course_id": "edx/AN101/2014_T1",
"module": {"display_name": "Multiple Choice Questions"}, "org_id": "edx", "user_id":
9999999}, "event": {"answers": {"i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1":
"yellow", "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_4_1": ["choice_0", "choice_2"]},
"attempts": 1, "correct_map": {"i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1":
{"correctness": "incorrect", "hint": "", "hintmode": null, "msg": "", "npoints": null,
"queuestate": null}, "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_4_1":
{"correctness": "correct", "hint": "", "hintmode": null, "msg": "", "npoints": null,
"queuestate": null}}, "grade": 2, "max_grade": 3, "problem_id": "i4x://edx/AN101/problem/
a0effb954cca4759994f1ac9e9434bf4", "state": {"correct_map": {}, "done": null, "input_state":
{"i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1": {}, "i4x-edx-AN101-problem-
a0effb954cca4759994f1ac9e9434bf4_4_1": {}}, "seed": 1, "student_answers": {}}, "submission":
{"i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1": {"answer": "yellow", "correct":
false, "input_type": "optioninput", "question": "What color is the open ocean on a sunny day?",
"response_type": "optionresponse", "variant": ""}, "i4x-edx-AN101-problem-
a0effb954cca4759994f1ac9e9434bf4_4_1": {"answer": ["a piano", "a guitar"], "correct": true,
"input_type": "checkboxgroup", "question": "Which of the following are musical instruments?",
"response_type": "choiceresponse", "variant": ""}}, "success": "incorrect"}, "event_source":
"server", "event_type": "problem_check", "host": "precise64", "referer": "http:\/\/localhost:8001\/
container\/i4x:\/\/edX\/DemoX\/vertical\/69dedd38233a46fc89e4d7b5e8da1bf4?action=new",
```

```
"accept_language": "en-US,en;q=0.8","ip": "NN.N.N.N", "page": "x_module",
"time": 2014-03-03T16:19:05.584523+00:00", "username": "AAAAAAAAAA"}
```

If you use a JSON formatter to "pretty print" this event, a version that is more readable is produced.


```
{
   "agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599
   "context": {
        "course_id": "edx/AN101/2014_T1",
        "module": {
            "display_name": "Multiple Choice Questions"
        },
        "org_id": "edx",
        "user_id": 9999999
   },
   "event": {
        "answers": {
            "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1": "yellow",
            "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_4_1": [
                "choice_0",
                "choice_2"
            ]
        },
        "attempts": 1,
        "correct_map": {
            "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1": {
                "correctness": "incorrect",
                "hint": "",
                "hintmode": null,
                "msg": "",
                "npoints": null,
                "queuestate": null
            },
            "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_4_1": {
                "correctness": "correct",
                "hint": "",
                "hintmode": null,
                "msg": "",
                "npoints": null,
                "queuestate": null
            }
        },
        "grade": 2,
        "max_grade": 3,
        "problem_id": "i4x://edx/AN101/problem/a0effb954cca4759994f1ac9e9434bf4",
        "state": {
            "correct_map": {},
            "done": null,
            "input_state": {
                "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1": {},
                "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_4_1": {}
            },
            "seed": 1,
            "student_answers": {}
        },
        "submission": {
            "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_2_1": {
                "answer": "yellow",
                "correct": false,
```

```
                "input_type": "optioninput",
                "question": "What color is the open ocean on a sunny day?",
                "response_type": "optionresponse",
                "variant": ""
            },
            "i4x-edx-AN101-problem-a0effb954cca4759994f1ac9e9434bf4_4_1": {
                "answer": [
                    "a piano",
                    "a guitar"
                ],
                "correct": true,
                "input_type": "checkboxgroup",
                "question": "Which of the following are musical instruments?",
                "response_type": "choiceresponse",
                "variant": ""
            }
        },
        "success": "incorrect"
    },
    "event_source": "server",
    "event_type": "problem_check",
    "host": "precise64",
    "referer": "http:\/\/localhost:8001\/container\/i4x:\/\/edX\/DemoX\/vertical\/69dedd38233a46fc89e4
    "accept_language": "en-US,en;q=0.8",
    "ip": "NN.N.N.N",
    "page": "x_module",
    "time": "2014-03-03T16:19:05.584523+00:00",
    "username": "AAAAAAAAAA"
}
```


For more information about fields that are included in every event, see *Common Fields*. For more information about this `problem_check` event and other types of events, see *Student Events* or *Course Team Events*.

11.2 Common Fields

This section describes the JSON fields that are common to the schema definitions of all events. These fields are at the root level of the event JSON documents.

This section presents the common fields in alphabetical order. Actual events in your data package can include these fields in different sequences.

11.2.1 `accept_language` Field

Type: string

Details: The value from the HTTP Accept-Language request-header field. For more information, see the HTTP/1.1 header field definition for Accept-Language.

History: Added 23 Feb 2015.

11.2.2 `agent` Field

Type: string

Details: Browser agent string of the user who triggered the event.

11.2.3 context Field

Type: object

Details:

The context field includes member fields that provide contextual information.

- This field contains a core set of member fields that are common to all events.
- For certain events with additional contextual requirements, this field contains a set of additional member fields that are common to those events only.
- For any event, this field can also include one or more additional member fields. For more information about the context member fields for an event, see the description of that event later in this section.

context Member Fields Common to All Events

The following member fields are present in the context field for all events.

context Member Field	Type	Details
course_id	string	Identifies the course that generated the event.
org_id	string	The organization that lists the course.
path	string	The URL that generated the event.
user_id	number	Identifies the individual who is performing the action.

context Member Fields for Applicable Events

When applicable for an event, the context field also includes these member fields to provide additional information.

context Member Field	Type	Details
course_user_tags	object	Contains the key(s) and value(s) from the user_api_usercoursetag table for the user. See *Columns in the user_api_usercoursetag Table*.
module	object	Provides identifying information for the components involved in a server event. For example, in a server problem_check event, the module field indicates the problem component that the server checked successfully. The member fields are display_name and usage_key. For modules that are used in a course to present content from a library, module also includes the original_usage_key and original_usage_version fields. These member fields provide a consistent way to identify components that are sourced from a library, and can be used to identify the source library.

The context member fields are blank if values cannot be determined.

History: usage_key added 28 Jan 2015. path added 07 May 2014. course_user_tags added 12 Mar 2014. user_id added 6 Nov 2013. Other event fields may duplicate this data. Added 23 Oct 2013.

11.2.4 event Field

Type: object

Details: This field includes member fields that identify specifics of each triggered event. Different member fields are supplied for different events. For more information about the event member fields for an event, see the description of that event later in this section.

11.2.5 `event_source` Field

Type: string

Details: Specifies the source of the interaction that triggered the event. The values in this field are:

- 'browser'
- 'mobile'
- 'server'
- 'task'

History: Updated 16 Oct 2014 to identify events emitted from mobile devices.

11.2.6 `event_type` Field

Type: string

Details: The type of event triggered. Values depend on `event_source`.

Student Events and *Course Team Events* later in this section provide descriptions of each type of event that is included in data packages. To locate information about a specific event type, see the *Alphabetical Event List*.

11.2.7 `host` Field

Type: string

Details: The site visited by the user, for example, `courses.edx.org`.

11.2.8 `ip` Field

Type: string

Details: IP address of the user who triggered the event. Empty for events that originate on mobile devices.

11.2.9 `name` Field

Type: string

Details: Identifies the type of event triggered.

History: Server and mobile events added beginning on 07 May 2014 include a `name` field. When this field is present for an event, it supersedes the `event_type` field.

11.2.10 `page` Field

Type: string

Details: The '$URL' of the page the user was visiting when the event was emitted.

For video events that originate on mobile devices, identifies the URL for the video component.

11.2.11 `referer` Field

Type: string

Details: The URI from the HTTP Referer request-header field. For more information, see the HTTP/1.1 header field definition for Referer.

History: Added 23 Feb 2015.

11.2.12 `session` Field

Type: string

Details: This 32-character value is a key that identifies the user's session. All browser events and the server *enrollment* events include a value for the session. Other server events and mobile events do not include a session value.

11.2.13 `time` Field

Type: string

Details: Gives the UTC time at which the event was emitted in 'YYYY-MM-DDThh:mm:ss.xxxxxx' format.

11.2.14 `username` Field

Type: string

Details: The username of the user who caused the event to be emitted. This string is empty for anonymous events, such as when the user is not logged in.

11.3 Student Events

This section lists the events that are typically initiated by learners. These events are generated by interactions with the learning management system (LMS) other than the Instructor Dashboard.

The descriptions that follow include what each event represents, the system component it originates from, the history of any changes made to the event over time, and any additional member fields that the common `context` or `event` fields contain. For more information about the common `context` or `event` fields, see *Common Fields*.

The value in the `event_source` field (see the *Common Fields* section above) distinguishes between events that originate in the browser (in JavaScript) and events that originate on the server (during the processing of a request).

11.3.1 Enrollment Events

This section includes descriptions of the following events.

edx.course.enrollment.activated and edx.course.enrollment.deactivated

The server emits these events in response to course enrollment activities completed by a student.

- When a student enrolls in a course, the server emits an `edx.course.enrollment.activated` event. For example, when a student clicks **Enroll** for a course on the edx.org site, the server emits this event.
- When a student unenrolls from a course, the server emits an `edx.course.enrollment.deactivated` event. For example, when a student clicks **Unenroll** for a course on the edx.org site, the server emits this event.

In addition, actions by course team members also generate enrollment events. For the actions that members of the course team complete that result in these events, see *Instructor Enrollment Events*.

Event Source: Server

History: These enrollment events were added on 03 Dec 2013. On 07 May 2014, the `name` field was added. These enrollment events include both a `name` field and an `event_type` field.

`event` **Member Fields**:

Field	Type	Details
`course_id`	string	The course in which the student was enrolled or unenrolled. If an external tool is used to enroll or unenroll students, this field contains a value and the `context.course_id` field is null.
`mode`	string	'audit', 'honor', 'professional', 'verified'. Identifies the student's enrollment mode.
`user_id`	number	Identifies the student who was enrolled or unenrolled.

Example

```
{
    "username": "AAAAAAAAAA",
    "event_source": "server",
    "name": "edx.course.enrollment.deactivated",
    "referer": "http:\/\/localhost:8001\/container\/i4x:\/\/edX\/DemoX\/vertical\/69dedd38233a46fc89e
    "accept_language": "en-US,en;q=0.8",
    "time": "2014-01-26T00:28:28.388782+00:00",
```

```
    "agent": "Mozilla\/5.0 (Windows NT 6.1; WOW64; Trident\/7.0; rv:11.0) like Gecko",
    "page": null
    "host": "courses.edx.org",
    "session": "a14j3ifhskngw0gfgn230g",
    "context": {
      "user_id": 9999999,
      "org_id": "edX",
      "course_id": "edX\/DemoX\/Demo_Course",
      "path": "\/change_enrollment",
    },
    "ip": "NN.NN.NNN.NNN",
    "event": {
      "course_id": "edX\/DemoX\/Demo_Course",
      "user_id": 9999999,
      "mode": "honor"
    },
    "event_type": "edx.course.enrollment.deactivated"
  }
```


edx.course.enrollment.mode_changed

The server emits an `edx.course.enrollment.mode_changed` event when the process of changing a student's `student_courseenrollment.mode` to a different mode is complete.

Event Source: Server

History: Added 21 Aug 2014.

`event` **Member Fields**:

Field	Type	Details
`course_id`	string	The course in which the student's enrollment mode has changed.
`mode`	string	'audit', 'honor', 'professional', verified'. Identifies the student's new enrollment mode.
`user_id`	number	Identifies the student whose enrollment mode changed.

edx.course.enrollment.upgrade.clicked

Students who enroll with a `student_courseenrollment.mode` of 'audit' or 'honor' in a course that has a verified certificate option see a **Challenge Yourself** link for the course on their dashboards. The browser emits this event when a student clicks this option, and the process of upgrading the `student_courseenrollment.mode` for the student to 'verified' begins. See *Columns in the student_courseenrollment Table*.

Event Source: Browser

History: Added 18 Dec 2013.

`context` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `context` member field.

Field	Type	Details and Member Fields
`mode`	string	Enrollment mode when the user clicked **Challenge Yourself**: 'audit' or 'honor'.

`event` **Member Fields**: None.

edx.course.enrollment.upgrade.succeeded

The server emits this event when the process of upgrading a student's `student_courseenrollment.mode` from 'audit' or 'honor' to 'verified' is complete.

Event Source: Server

History: Added 18 Dec 2013.

`context` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `context` member field.

Field	Type	Details and Member Fields
`mode`	string	Set to 'verified'.

`event` **Member Fields**: None.

11.3.2 Navigational Events

This section includes descriptions of the following events.

page_close

The `page_close` event originates from within the JavaScript Logger itself.

Component: JavaScript Logger

Event Source: Browser

`event` **Member Fields**: None

seq_goto, seq_next, and seq_prev

The browser emits these events when a user selects a navigational control.

- `seq_goto` is emitted when a user jumps between units in a sequence.
- `seq_next` is emitted when a user navigates to the next unit in a sequence.
- `seq_prev` is emitted when a user navigates to the previous unit in a sequence.

Component: Sequence

Event Source: Browser

`event` **Member Fields**:

All of these navigational events have the same `event` member fields.

Field	Type	Details
`id`	number	The edX ID of the sequence.
`new`	number	For `seq_goto`, the index of the unit being jumped to. For `seq_next` and `seq_prev`, the index of the unit being navigated to.
`old`	number	For `seq_goto`, the index of the unit being jumped from. For `seq_next` and `seq_prev`, the index of the unit being navigated away from.

11.3.3 Video Interaction Events

This section includes descriptions of the following events. Due to a naming convention change, many of these events have two identifying names. In this list, the original name, which is present in the `event_type` field for all events, is followed by a newer, revised name. The revised name is present in the `name` field only for events that have an `event_source` of 'mobile'.

A browser or the edX mobile app emits video interaction events when a user interacts with a video.

- When users use a browser to stream video files on a desktop computer or mobile device, the browser emits the events.
- When users use the edX mobile app to stream or download course videos for offline viewing, the mobile app emits the events.

  When a user interacts with a downloaded video file offline using the edX mobile app, note that the app can only forward its events during the next connection opportunity. As a result, the date and time in the event's `time` field can be different from the date and time in its `context.received_at` field. Data packages can include events emitted on past dates.

This section presents the video interaction events alphabetically. Typically, an interaction with the video player begins with a *play_video/edx.video.played* event.

For courses that include a pre-roll video, user interactions with the pre-roll video result in different events. For more information, see *Pre-Roll Video Interaction Events*.

Component: Video

History: The edX mobile app for iOS began to emit a subset of the video events on 25 Feb 2015. The edX mobile app for Android began to emit a subset of the video events on 23 Dec 2014.

hide_transcript/edx.video.transcript.hidden

When a user selects **CC** to suppress display of the video transcript, the browser or mobile app emits a `hide_transcript` event.

In addition to the identifying `event_type` of `hide_transcript`, events that the edX mobile app emits also include a `name` field with a value of `edx.video.transcript.hidden`.

Event Source: Browser or Mobile

History: Updated 25 Feb 2015 to include events emitted by the edX mobile app for iOS. Updated 23 Dec 2014 to include events emitted by the edX mobile app for Android.

`context` **Member Fields**:

Only video interaction events with an `event_source` of 'mobile' include additional `context` member fields in addition to the *common* member fields. The same set of additional context fields are added for `hide_transcript`/ `edx.video.transcript.hidden` events as for the *play_video/edx.video.played* events. For an example of an event with these fields, see *Example: Mobile App-Emitted edx.video.played Event*.

`event` **Member Fields**:

The `hide_transcript`/`edx.video.transcript.hidden` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *play_video/edx.video.played* events.

- `code`
- `currentTime`: The point in the video file at which the transcript was hidden.
- `id`

load_video/edx.video.loaded

When the video is fully rendered and ready to play, the browser or mobile app emits a `load_video` event.

In addition to the identifying `event_type` of `load_video`, the events that the edX mobile app emits also include a `name` field with a value of `edx.video.loaded`.

Event Source: Browser or Mobile

History: Updated 25 Feb 2015 to include events emitted by the edX mobile app for iOS. Updated 23 Dec 2014 to include events emitted by the edX mobile app for Android.

`context` **Member Fields**:

Only video interaction events with an `event_source` of 'mobile' include additional `context` member fields in addition to the *common* member fields. The same set of additional context fields are added for `load_video` events as for *play_video/edx.video.played*. For an example of an event with these fields, see *Example: Mobile App-Emitted edx.video.played Event*.

`event` **Member Fields**:

The `load_video`/ `edx.video.loaded` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *play_video/edx.video.played* events.

- `code`
- `id`

pause_video/edx.video.paused

When a user selects the video player's **pause** control, the player emits a `pause_video` event. For videos that are streamed in a browser, when the player reaches the end of the video file and play automatically stops it emits both this event and a `stop_video` event (as of June 2014).

Note that course teams can specify a **Video Stop Time** for video files.

- If the user streams a video file in a browser and a **Video Stop Time** is present for the video, the player stops at the specified time and emits the `pause_video` and `stop_video` events.
- If the user plays a streaming or downloaded video in the edX mobile app, the app ignores the **Video Stop Time** and plays the file to its end. The app then emits only the `stop_video` event.

For more information, see Working with Video Components in the *Building and Running an edX Course* guide.

In addition to the identifying `event_type` of `pause_video`, the events that the edX mobile app emits include a `name` field with a value of `edx.video.paused`.

Event Source: Browser or Mobile

History:

- Updated 5 May 2015 to include the effect of a **Video Stop Time**.
- Updated 25 Feb 2015 to include events emitted by the edX mobile app for iOS.
- Updated 23 Dec 2014 to include events emitted by the edX mobile app for Android.

`context` **Member Fields**:

Only video interaction events with an `event_source` of 'mobile' include additional `context` member fields in addition to the *common* member fields. The same set of additional context fields are added for `pause_video`/`edx.video.paused` events as for *play_video/edx.video.played*. For an example of an event with these fields, see *Example: Mobile App-Emitted edx.video.played Event*.

`event` **Member Fields**:

The `pause_video`/`edx.video.paused` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *play_video/edx.video.played* events.

- `code`
- `currentTime`: The time in the video at which the video paused.
- `id`

play_video/edx.video.played

When a user selects the video player's **play** control, the player emits a `play_video` event.

Note that course teams can specify a **Video Start Time** for video files.

- If the user streams a video file in a browser and a **Video Start Time** is present for the video, the player starts at the specified time and emits the `play_video` event.
- If the user plays a streaming or downloaded video in the edX mobile app, the app ignores the **Video Start Time** and emits the `play_video` event when it plays the file from the beginning.

For more information, see Working with Video Components in the *Building and Running an edX Course* guide.

In addition to the identifying `event_type` of `play_video`, events that the edX mobile app emits also include a `name` field with a value of `edx.video.played`.

Event Source: Browser or Mobile

History:

- Updated 5 May 2015 to include the effect of a **Video Start Time**.
- Updated 25 Feb 2015 to include events emitted by the edX mobile app for iOS.
- Updated 23 Dec 2014 to include events emitted by the edX mobile app for Android.

`context` **Member Fields**:

Only video interaction events with an `event_source` of 'mobile' include additional `context` member fields in addition to the *common* member fields. Other video interaction events with an `event_source` of mobile also include these fields. For an example of an event with these fields, see *Example: Mobile App-Emitted edx.video.played Event*.

Field	Type	Details and Member Fields
`application`	object	Includes `name` and `version` member fields to identify the edX mobile app.
`client`	object	Includes member objects and fields with device-specific data. The `client` data is gathered by the event collection library, which is provided by a third party. The content of this field is subject to change without notice.
`component`	string	'videoplayer'
`received_at`	number	Indicates the time at which the event collection library received the event. Events can only be forwarded when the mobile device is connected to the Internet. Therefore, this value can be different than the event's `time` value. The data in this field reflects a third-party integration and is subject to change at any time without notice.

`event` **Member Fields**:

Field	Type	Details
`code`	string	For YouTube videos played in a browser, the ID of the video being loaded (for example, OEyXaRPEzfM). For non-YouTube videos played in a browser, 'html5'. For videos played by the edX mobile app, 'mobile'.
`currentTime`	number	The time in the video at which the event was emitted.
`id`	string	The optional name value supplied by the course creators, or the system- generated hash code for the video being watched. For example, `0b9e39477cf34507a7a48f74be381fdd`. This value is part of the `courseware_studentmodule.module_id`. See *Columns in the courseware_studentmodule Table*. **History**: In October 2014, identifiers for some new courses began to use the format shown above. Other new courses, and all courses created prior to October 2014, use an HTML-escaped version of the `courseware_studentmodule.module_id`. For example, `i4x-HarvardX-PH207x-video-Simple_Random_Sample`.

Example: Browser-Emitted `play_video` Event


```
{
  "event_source": "browser",
  "event": "{\"id\":\"i4x-BerkeleyX-Stat_2_1x-video-58424ad2f75048798b4480aa699cc215\",\"currentTime\
  "time": "2014-12-23T14:26:53.723188+00:00",
  "referer": "http:\/\/localhost:8001\/container\/i4x:\/\/edX\/DemoX\/vertical\/69dedd38233a46fc89e4c
  "accept_language": "en-US,en;q=0.8",
  "event_type": "play_video",
  "session": "11a1111111a1a1a1aa1a11a1a1111111",
```

```
  "agent": "Mozilla\/5.0 (Windows NT 6.1; WOW64) AppleWebKit\/537.36 (KHTML, like Gecko) Chrome\/39.0
  "page": "https:\/\/courses.edx.org\/courses\/BerkeleyX\/Stat_2.1x\/1T2014\/courseware\/d4ff35dabfe0
  "username": "AAAAAAAAAA",
  "ip": "123.123.123.123",
  "context": {
    "org_id": "BerkeleyX",
    "path": "\/event",
    "course_id": "BerkeleyX\/Stat_2.1x\/1T2014",
    "user_id": 99999999
  },
  "host": "courses.edx.org"
}
```


Example: Mobile App-Emitted `edx.video.played` Event


```
{
  "username": "AAAAAAAAAA",
  "event_source": "mobile",
  "name": "edx.video.played",
  "time": "2014-12-09T03:57:24+00:00",
  "agent": "Dalvik/1.6.0 (Linux; U; Android 4.0.2; sdk Build/ICS_MR0)",
  "page": "http://courses.edx.org/courses/edX/DemoX/Demo_Course/courseware/d8a6192ade314473a78242dfe
  "host": "courses.edx.org",
  "session": "",
  "context": {
      "component": "videoplayer",
      "received_at": "2014-12-09T03:57:56.373000+00:00",
      "course_id": "edX/DemoX/Demo_Course",
      "path": "/segment/event",
      "user_id": 99999999,
      "org_id": "edX",
      "application": {
        "name": "edx.mobileapp.android",
        "version": "0.1.8",
      },
      "client": {
          "network": {
              "wifi": false,
              "carrier": "Android",
              "cellular": true,
              "bluetooth": false
          },
          "locale": "en-US",
          "app": {
              "name": "edX",
              "packageName": "org.edx.mobile",
              "version": "0.1.8",
              "build": "org.edx.mobile@29",
              "versionName": "0.1.8",
              "versionCode": 29
          },
          "library": {
              "version": 203,
              "name": "analytics-android",
              "versionName": "2.0.3"
          },
          "device": {
```

```
                "model": "sdk",
                "type": "android",
                "id": "aaa11111aaaa11a1",
                "name": "generic",
                "manufacturer": "unknown"
            },
            "os": {
                "version": "4.0.2",
                "name": "REL",
                "sdk": 14
            },
            "screen": {
                "densityBucket": "xhdpi",
                "density": 2,
                "height": 1184,
                "width": 768,
                "densityDpi": 320,
                "scaledDensity": 2
            }
        }
    },
    "ip": "",
    "event": "{\"code\": \"mobile\", \"id\": \"i4x-edX-DemoX-video-0b9e39477cf34507a7a48f74be381fdd\",
    "event_type": "play_video"
}
```


seek_video/edx.video.position.changed

A browser emits `seek_video` events when a user selects a user interface control to go to a different point in the video file.

- On a desktop computer, users can click and drag in the playback bar or click in a transcript to change position.
- In the edX mobile app, users can click and drag in the playback bar or tap the "back 30 seconds" button to change position.
- When using a browser on a mobile device, users can click and drag in the playback bar to change position.

In addition to the value `seek_video` in the `event_type` field, the events that the edX mobile app emits include the value `edx.video.position.changed` in the `name` field.

Event Source: Browser or Mobile

History:

- Updated 10 Mar 2015 to include the final implementation for events emitted by the edX mobile app for Android and iOS. Prototype events were emitted by the mobile app in February and March 2015.
- Prior to 25 Jun 2014, the `old_time` and `new_time` fields were set to the same value.

`context` **Member Fields**:

Only video interaction events with an `event_source` of 'mobile' include additional `context` member fields in addition to the *common* member fields. The same set of additional context fields are added for `seek_video/edx.video.position.changed` events as for *play_video/edx.video.played*. For an example of an event with these fields, see *Example: Mobile App-Emitted edx.video.played Event*.

`event` **Member Fields**:

The `seek_video/edx.video.position.changed` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *play_video/edx.video.played* events.

- code
- id

The following additional event member fields apply specifically to seek_video/edx.video.position.changed events.

Field	Type	Details
new_time	number	The time in the video, in seconds, that the user selected as the destination point.
old_time	number	The time in the video, in seconds, at which the user chose to go to a different point in the file.
requested_skip_interval	number	Applies only to events with an event_source of 'mobile'. The number of seconds that the user moved backward (expressed as a negative) or forward in the file. **History**: Added 10 Mar 2015.
type	string	The navigational method used to change position within the video. In events for a user of a desktop computer, this value can be 'onCaptionSeek' or 'onSlideSeek'. In events for a user of the mobile app, this value can be 'onSlideSeek' or 'onSkipSeek'.

show_transcript/edx.video.transcript.shown

When a user selects **CC** to display the video transcript, the browser or mobile app emits a show_transcript event.

In addition to the identifying event_type of show_transcript, events that the edX mobile app emits also include a name field with a value of edx.video.transcript.shown.

Event Source: Browser or Mobile

History: Updated 25 Feb 2015 to include events emitted by the edX mobile app for iOS. Updated 23 Dec 2014 to include events emitted by the edX mobile app for Android.

context **Member Fields**:

Only video interaction events with an event_source of 'mobile' include additional context member fields in addition to the *common* member fields. The same set of additional context fields are added for show_transcript/edx.video.transcript.shown events as for *play_video/edx.video.played*. For an example of an event with these fields, see *Example: Mobile App-Emitted edx.video.played Event*.

event **Member Fields**:

The show_transcript/edx.video.transcript.shown events include the following event member fields. These fields serve the same purpose for events of this type as for the *play_video/edx.video.played* events.

- code
- currentTime: The point in the video file at which the transcript was opened.
- id

speed_change_video

A browser emits speed_change_video events when a user selects a different playing speed for the video.

Event Source: Browser

History: Prior to 12 Feb 2014, this event was emitted when a user selected either the same speed or a different speed.

event **Member Fields**:

Field	Type	Details
current_time	number	The time in the video that the user chose to change the playing speed.
new_speed	number	The speed that the user selected for the video to play: '0.75', '1.0', '1.25', '1.50'.
old_speed	number	The speed at which the video was playing.

stop_video/edx.video.stopped

When the video player reaches the end of the video file and play automatically stops, the player emits a stop_video event.

Note that course teams can specify a **Video Stop Time** for video files.

- If the user streams a video file in a browser and a **Video Stop Time** is present for the video, the player stops at the specified time and emits the pause_video and stop_video events.
- If the user plays a streaming or downloaded video in edX mobile app, the app ignores the **Video Stop Time** and plays the file to its end. The app then emits the stop_video event.

For more information, see Working with Video Components in the *Building and Running an edX Course* guide.

In addition to the identifying event_type of stop_video, the events that the edX mobile app emits include a name field with a value of edx.video.stopped.

Event Source: Browser or Mobile

History:

- Updated 5 May 2015 to include the effect of a **Video Stop Time**.
- Updated 25 Feb 2015 to include events emitted by the edX mobile app for iOS.
- Updated 23 Dec 2014 to include events emitted by the edX mobile app for Android.
- Added 25 June 2014.

context **Member Fields**:

Only video interaction events with an event_source of 'mobile' include additional context member fields in addition to the *common* member fields. The same set of additional context fields are added for stop_video/edx.video.stopped events as for *play_video/edx.video.played*. For an example of an event with these fields, see *Example: Mobile App-Emitted edx.video.played Event*.

event **Member Fields**:

The stop_video/edx.video.stopped events include the following event member fields. These fields serve the same purpose for events of this type as for the *play_video/edx.video.played* events.

- code
- currentTime: The time in the video at which play stopped.
- id

video_hide_cc_menu

When a user selects a language from the **CC** menu for a video that has transcripts in multiple languages, the browser emits a video_hide_cc_menu event.

Event Source: Browser

History: Added 17 Feb 2015.

event **Member Fields**:

The `video_hide_cc_menu` events include the following `event` member fields. These fields serve the same purpose for events of this type as for *play_video/edx.video.played*.

- `code`
- `id`

`video_show_cc_menu`

When a user selects **CC** for a video that has transcripts in multiple languages, the browser emits a `video_show_cc_menu` event. This event is emitted in addition to the `show_transcript` event.

Event Source: Browser

History: Added 17 Feb 2015.

`event` **Member Fields**:

The `video_show_cc_menu` events include the following `event` member fields. These fields serve the same purpose for events of this type as for *play_video/edx.video.played*.

- `code`
- `id`

11.3.4 Pre-Roll Video Interaction Events

Course teams can create a short video message and configure it to play automatically before the videos in a course.

- The pre-roll video plays on an infrequent schedule of once per user per week.
- Only courses that run on the edx.org website can include a pre-roll video.
- The edX mobile applications do not play pre-roll videos.

When a user interacts with the pre-roll video, different events are emitted than for the other videos in the course. This section presents the pre-roll video events alphabetically.

For more information about pre-roll videos, see Adding a Pre-Roll Video to Your edX Course in the *Building and Running an edX Course* guide.

Component: Video

History: Added 10 Jun 2015.

`edx.video.bumper.dismissed`

A browser emits this event when a user selects **Do not show again** for a pre-roll video. This option allows the user to opt out of viewing the course pre-roll video in the future.

Event Source: Browser

`event` **Member Fields**:

The `edx.video.bumper.dismissed` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- `bumper_id`
- `code`
- `currentTime`

- duration
- host_component_id

edx.video.bumper.loaded

When the pre-roll video is fully rendered and ready to play, the browser emits an edx.video.bumper.loaded event.

Event Source: Browser

event **Member Fields**:

The edx.video.bumper.loaded events include the following event member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- bumper_id
- code
- duration
- host_component_id

edx.video.bumper.played

When a user selects the **play** control in the video player for a pre-roll video, the browser emits an edx.video.bumper.played event.

Event Source: Browser

event **Member Fields**:

Field	Type	Details
bumper_id	string	The escaped URL identifying the location of the pre-roll video that played.
code	string	Contains the value 'html5'. All pre-roll videos are non-YouTube videos.
currentTime	number	The time in the file at which the video played.
duration	number	The length of the video file, in seconds.
host_component_id	string	Identifier for the video component that the user selected, and that is queued to play after the pre-roll video.

Example: edx.video.bumper.played Event


```
{
  "username": "honor",
  "event_source": "browser",
  "name": "edx.video.bumper.played",
  "accept_language": "en-US,en;q=0.5",
  "time": "2015-05-26T18:22:07.684172+00:00",
  "agent": "Mozilla\/5.0 (X11; Ubuntu; Linux x86_64; rv:37.0) Gecko\/20100101 Firefox\/37.0",
  "page": "http:\/\/edx.org\/courses\/edX\/DemoX.1\/2015\/courseware\/0af8db2309474971bfa70cda98668a3
  "host": "precise64",
  "session": "feae6efa342b309e776d388b16da89a6",
  "referer": "http:\/\/localhost:8001\/container\/i4x:\/\/edX\/DemoX\/vertical\/69dedd38233a46fc89e4c
  "context": {
    "user_id": 7911,
```

```
    "org_id": "edX",
    "course_id": "edX\/DemoX.1\/2015",
    "path": "\/event"
  },
  "ip": "123.0.0.1",
  "event": "{\"duration\": 10, \"bumper_id\": \"http:\/\/www.w3schools.com\/html\/mov_bbb.webm\", \"
  "event_type": "edx.video.bumper.played"
}
```


edx.video.bumper.skipped

A browser emits this event when a user selects **Skip** for a bumper video. This option allows the user to advance past the pre-roll video and begin to play the selected video immediately.

Event Source: Browser

`event` **Member Fields**:

The `edx.video.bumper.skipped` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- `bumper_id`
- `code`
- `currentTime`: The point in the file at which the pre-roll video was skipped.
- `duration`
- `host_component_id`

edx.video.bumper.stopped

A browser emits this event when the video player reaches the end of the pre- roll video file and play automatically stops.

This is the only event that is emitted when a user pauses a pre-roll video.

Event Source: Browser

`event` **Member Fields**:

The `edx.video.bumper.stopped` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- `bumper_id`
- `code`
- `currentTime`: The point in the file at which the pre-roll video was stopped.
- `duration`
- `host_component_id`

edx.video.bumper.transcript.hidden

When a user selects **CC** to suppress display of the transcript for a pre-roll video, the browser emits a `edx.video.bumper.transcript.hidden` event.

Event Source: Browser

`event` **Member Fields**:

The `edx.video.bumper.transcript.hidden` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- `bumper_id`
- `code`
- `currentTime`: The point in the file at which the transcript was hidden.
- `duration`
- `host_component_id`

edx.video.bumper.transcript.menu.hidden

When a user selects a language from the **CC** menu for a pre-roll video that has transcripts in multiple languages, the browser emits an `edx.video.bumper.transcript.menu.hidden` event.

Event Source: Browser

`event` **Member Fields**:

The `edx.video.bumper.transcript.menu.hidden` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- `bumper_id`
- `code`
- `currentTime`: The point in the file at which the language was selected and the transcript menu was hidden.
- `duration`
- `host_component_id`

edx.video.bumper.transcript.menu.shown

When a user selects **CC** for a pre-roll video that has transcripts in multiple languages, the browser emits an `edx.video.bumper.transcript.menu.shown` event. This event is emitted in addition to the `edx.video.bumper.transcript.shown` event.

Event Source: Browser

`event` **Member Fields**:

The `edx.video.bumper.transcript.menu.shown` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- `bumper_id`
- `code`
- `currentTime`: The point in the file at which the transcript menu was shown.
- `duration`
- `host_component_id`

edx.video.bumper.transcript.shown

When a user selects **CC** to display the transcript for a pre-roll video, the browser emits a `edx.video.bumper.transcript.shown` event. If the video has more than one transcript file, the `edx.video.bumper.transcript.menu.shown` event is also emitted.

The `edx.video.bumper.transcript.shown` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the *edx.video.bumper.played* events.

- `bumper_id`
- `code`
- `currentTime`: The point in the file at which the transcript was shown.
- `duration`
- `host_component_id`

11.3.5 Textbook Interaction Events

This section includes descriptions of the following events.

book

The browser emits `book` events when a user navigates within the PDF Viewer or the PNG Viewer.

- For textbooks in PDF format, the URL in the common `page` field contains '/pdfbook/'.
- For textbooks in PNG format, the URL in the common `page` field contains '/book/'.

Component: PDF Viewer, PNG Viewer

Event Source: Browser

History: This event changed on 16 Apr 2014 to include `event` member fields `name` and `chapter`.

`event` **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file. **History**: Added for events produced by the PDF Viewer on 16 Apr 2014.
name	string	• For 'gotopage', set to textbook.pdf.page.loaded. • For 'prevpage', set to textbook.pdf.page.navigatedprevi • For 'nextpage', set to textbook.pdf.page.navigatednext. **History**: Added for events produced by the PDF Viewer on 16 Apr 2014.
new	number	Destination page number.
old	number	The original page number. Applies to 'gotopage' event types only.
type	string	• 'gotopage' is emitted when a page loads after the student manually enters its number. • 'prevpage' is emitted when the next page button is clicked. • 'nextpage' is emitted when the previous page button is clicked.

textbook.pdf.thumbnails.toggled

The browser emits textbook.pdf.thumbnails.toggled events when a user clicks on the icon to show or hide page thumbnails.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file.
name	string	textbook.pdf.thumbnails.toggled
page	number	The number of the page that is open when the user clicks this icon.

textbook.pdf.thumbnail.navigated

The browser emits textbook.pdf.thumbnail.navigated events when a user clicks on a thumbnail image to navigate to a page.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file.
name	string	textbook.pdf.thumbnail.navigated
page	number	The page number of the thumbnail clicked.
thumbnail_title	string	The identifying name for the destination of the thumbnail. For example, Page 2.

textbook.pdf.outline.toggled

The browser emits textbook.pdf.outline.toggled events when a user clicks the outline icon to show or hide a list of the book's chapters.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file.
name	string	textbook.pdf.outline.toggled
page	number	The number of the page that is open when the user clicks this link.

textbook.pdf.chapter.navigated

The browser emits textbook.pdf.chapter.navigated events when a user clicks on a link in the outline to navigate to a chapter.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file.
chapter_title	string	The identifying name for the destination of the outline link.
name	string	textbook.pdf.chapter.navigated

textbook.pdf.page.navigated

The browser emits textbook.pdf.page.navigated events when a user manually enters a page number.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file.
name	string	textbook.pdf.page.navigated
page	number	The destination page number entered by the user.

textbook.pdf.zoom.buttons.changed

The browser emits textbook.pdf.zoom.buttons.changed events when a user clicks either the Zoom In or Zoom Out icon.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file.
direction	string	'in', 'out'
name	string	textbook.pdf.zoom.buttons.changed
page	number	The number of the page that is open when the user clicks the icon.

textbook.pdf.zoom.menu.changed

The browser emits textbook.pdf.zoom.menu.changed events when a user selects a magnification setting.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
amount	string	'1', '0.75', '1.5', 'custom', 'page_actual', 'auto', 'page_width', 'page_fit'.
chapter	string	The name of the PDF file.
name	string	textbook.pdf.zoom.menu.changed
page	number	The number of the page that is open when the user selects this value.

textbook.pdf.display.scaled

The browser emits textbook.pdf.display.scaled events when the display magnification changes. These changes occur after a student selects a magnification setting from the zoom menu or resizes the browser window.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
amount	string	The magnification setting; for example, 0.95 or 1.25.
chapter	string	The name of the PDF file.
name	string	textbook.pdf.display.scaled
page	number	The number of the page that is open when the scaling takes place.

textbook.pdf.page.scrolled

The browser emits textbook.pdf.page.scrolled events each time the displayed page changes while a user scrolls up or down.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
chapter	string	The name of the PDF file.
direction	string	'up', 'down'
name	string	textbook.pdf.page.scrolled
page	number	The number of the page that is open when the scrolling takes place.

textbook.pdf.search.executed

The browser emits textbook.pdf.search.executed events when a user searches for a text value in the file. To reduce the number of events produced, instead of producing one event per entered character this event defines a search string as the set of characters that is consecutively entered in the search field within 500ms of each other.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

event **Member Fields**:

Field	Type	Details
caseSensitive	Boolean	'true' if the case sensitive option is selected. 'false' if this option is not selected.
chapter	string	The name of the PDF file.
highlightAll	Boolean	'true' if the option to highlight all matches is selected. 'false' if this option is not selected.
name	string	textbook.pdf.search.executed
page	number	The number of the page that is open when the search takes place.
query	string	The value in the search field.
status	string	A "not found" status phrase for a search string that is unsuccessful. Blank for successful search strings.

textbook.pdf.search.navigatednext

The browser emits textbook.pdf.search.navigatednext events when a user clicks on the Find Next or Find Previous icons for an entered search string.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

`event` **Member Fields**:

Field	Type	Details
`caseSensitive`	Boolean	'true' if the case sensitive option is selected. 'false' if this option is not selected.
`chapter`	string	The name of the PDF file.
`findprevious`	Boolean	'true' if the user clicks the Find Previous icon. 'false' if the user clicks the Find Next icon.
`highlightAll`	Boolean	'true' if the option to highlight all matches is selected. 'false' if this option is not selected.
`name`	string	`textbook.pdf.search.navigatednext`
`page`	number	The number of the page that is open when the search takes place.
`query`	string	The value in the search field.
`status`	string	A "not found" status phrase for a search string that is unsuccessful. Blank for successful search strings.

`textbook.pdf.search.highlight.toggled`

The browser emits `textbook.pdf.search.highlight.toggled` events when a user selects or clears the **Highlight All** option for a search.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

`event` **Member Fields**:

Field	Type	Details
`caseSensitive`	Boolean	'true' if the case sensitive option is selected. 'false' if this option is not selected.
`chapter`	string	The name of the PDF file.
`highlightAll`	Boolean	'true' if the option to highlight all matches is selected. 'false' if this option is not selected.
`name`	string	`textbook.pdf.search.highlight.toggled`
`page`	number	The number of the page that is open when the search takes place.
`query`	string	The value in the search field.
`status`	string	A "not found" status phrase for a search string that is unsuccessful. Blank for successful search strings.

`textbook.pdf.searchcasesensitivity.toggled`

The browser emits `textbook.pdf.searchcasesensitivity.toggled` events when a user selects or clears the **Match Case** option for a search.

Component: PDF Viewer

Event Source: Browser

History: This event was added on 16 Apr 2014.

`event` **Member Fields**:

Field	Type	Details
`caseSensitive`	Boolean	'true' if the case sensitive option is selected. 'false' if this option is not selected.
`chapter`	string	The name of the PDF file.
`highlightAll`	Boolean	'true' if the option to highlight all matches is selected. 'false' if this option is not selected.
`name`	string	`textbook.pdf.searchcasesensitivity.toggled`
`page`	number	The number of the page that is open when the search takes place.
`query`	string	The value in the search field.
`status`	string	A "not found" status phrase for a search string that is unsuccessful. Blank for successful search strings.

11.3.6 Problem Interaction Events

This section includes descriptions of the following events.

Problem interaction events are emitted by the server or the browser to capture information about interactions with problems.

These events were designed for the problem types implemented in the edX platform by the `capa_module.py` XBlock. Problem types that are implemented by other XBlocks, such as *open response assessments*, *polls and surveys*, are instrumented with different events.

For more information about designing problems to include hints, feedback, or both, see Adding Feedback and Hints to a Problem in the *Building and Running an edX Course* guide.

edx.problem.hint.demandhint_displayed

Course teams can design problems to include one or more hints. For problems that include hints, the server emits an `edx.problem.hint.demandhint_displayed` event each time a user requests a hint.

Event Source: Server

History: This event was added on 1 Jul 2015.

`event` **Member Fields**:

Field	Type	Details
`hint_index`	number	Identifier for the hint that was displayed to the user. The first hint defined for a problem is identified with `hint_index: 0`.
`hint_len`	number	The total number of hints defined for this problem.
`hint_text`	string	The text of the hint that was displayed to the user.
`module_id`	string	Identifier for the problem component for which the user requested the hint.

`edx.problem.hint.feedback_displayed`

Course teams can design problems to include feedback messages that appear after a user submits an answer. For problems that include feedback messages, the server emits an `edx.problem.hint.feedback_displayed` event each time a user selects **Check**.

Event Source: Server

History: This event was added on 1 Jul 2015.

`event` **Member Fields**:

Field	Type	Details
`choice_all`	array	For problems that have a set of possible answers defined, such as checkbox problems, lists all of the answer choices.
`correctness`	Boolean	'True' if the `student_answer` response is correct. 'False' if the `student_answer` is incorrect.
`hint_label`	string	The optional label, such as 'Correct: ' or 'Incorrect: ', provided for the feedback message.
`hints`	array	Contains a `text` member field with the feedback string that was displayed to the user. For some problem types, such as checkbox problems, feedback can be provided for more than one answer at a time, including both correct and incorrect answers. A separate `text` member field is included for each feedback message that was displayed.
`module_id`	string	Identifier for the problem component for which the user received the feedback.
`problem_part_id`	string	For problem components that contain more than one problem, identifies the specific problem for which the user received feedback.
`question_type`	string	The XML tag that identifies the problem type. For example, 'stringresponse' for a text input problem.
`student_answer`	array	The answer value selected or supplied by the user. For problem types that accept multiple answers, such as checkbox problems, every response, including both selected and unselected options, is included.
`trigger_type`	string	Identifies the type of feedback elicited by the `student_answer` response. For checkbox problems only, course teams can design 'compound' feedback that is provided when a user's response matches an exact set of correct and incorrect selections across all of the available choices. All other types of feedback are identified as 'single'. For more information, see Adding Feedback and Hints to a Problem in the *Building and Running an edX Course* guide.

`problem_check` (Browser)

Both browser interactions and server requests produce `problem_check` events. The browser emits `problem_check` events when a user checks a problem.

Event Source: Browser

`event` **Member Fields**: For browser-emitted `problem_check` events, the `event` field contains the values of all input fields from the problem being checked, styled as GET parameters.

`problem_check` (Server)

Both browser interactions and server requests produce `problem_check` events.

The server emits `problem_check` events when a problem is successfully checked.

Event Source: Server

History:

- On 5 Mar 2014, the `submission` object was added to the `event` field and `module` was added to the `context` field.
- Prior to 15 Oct 2013, this server-emitted event was named `save_problem_check`.
- Prior to 15 Jul 2013, this event was emitted twice for the same action.

`context` **Member Fields**:

This event type includes the *common* `context.module` member field.

`event` **Member Fields**:

Field	Type	Details
answers	object	The problem ID and the internal answer identifier in a name/value pair. For a component with multiple problems, lists every problem and answer.
attempts	number	The number of times the user attempted to answer the problem.
correct_map	object	For each problem ID value listed by answers, provides: • correctness: string; 'correct', 'incorrect' • hint: string; Gives optional hint. Nulls allowed. • hintmode: string; None, 'on_request', 'always'. Nulls allowed. • msg: string; Gives extra message response. • npoints: number; Points awarded for this answer_id. Nulls allowed. • queuestate: object; None when not queued, else {key:'', time:''} where key is a secret string dump of a DateTime object in the form '%Y%m%d%H%M%S'. Nulls allowed.
grade	number	Current grade value.
max_grade	number	Maximum possible grade value.
problem_id	string	ID of the problem that was checked.
state	object	Current problem state.
submission	object	Provides data about the response made. For components that include multiple problems, a separate submission object is provided for each one. • answer: string; The value that the student entered, or the display name of the value selected. • correct: Boolean; 'true', 'false' • input_type: string; The type of value that the student supplies for the response_type. Based on the XML element names used in the Advanced Editor. Examples include 'checkboxgroup', 'radiogroup', 'choicegroup', and 'textline'. • question: string; Provides the text of the question. • response_type: string; The type of problem. Based on the XML element names used in the Advanced Editor.

problem_check_fail

The server emits problem_check_fail events when a problem cannot be checked successfully.

Event Source: Server

History: Prior to 15 Oct 2013, this event was named save_problem_check_fail.

event **Member Fields**:

Field	Type	Details
answers	object	
failure	string	'closed', 'unreset'
problem_id	string	ID of the problem being checked.
state	object	Current problem state.

problem_graded

The server emits a problem_graded event each time a user clicks **Check** for a problem and it is graded successfully.

event **Member Fields**:

Field	Type	Details
[answers, contents]	array	answers provides the value checked by the user. contents delivers HTML using data entered for the problem in Studio, including the display name, problem text, and choices or response field labels. The array includes each problem in a problem component that has multiple problems.

problem_rescore

The server emits problem_rescore events when a problem is successfully rescored.

Event Source: Server

event **Member Fields**:

Field	Type	Details
attempts	number	
correct_map	object	See the fields for the problem_check server event above.
new_score	number	
new_total	number	
orig_score	number	
orig_total	number	
problem_id	string	ID of the problem being rescored.
state	object	Current problem state.
success	string	'correct', 'incorrect'

problem_rescore_fail

The server emits problem_rescore_fail events when a problem cannot be successfully rescored.

Event Source: Server

event **Member Fields**:

Field	Type	Details
failure	string	'unsupported', 'unanswered', 'input_error', 'unexpected'
problem_id	string	ID of the problem being checked.
state	object	Current problem state.

problem_reset

The browser emits problem_reset events when a user clicks **Reset** to reset the answer to a problem.

Event Source: Browser

event **Member Fields**:

Field	Type	Details
answers	string	The value reset by the user.

problem_save

The browser emits problem_save events when a user saves a problem.

Event Source: Browser

event **Member Fields**: None

problem_show

The browser emits problem_show events when the answer to a problem is shown; that is, the user selected **Show Answer**.

Note: This event does not indicate when a problem was shown to a user.

Event Source: Browser

event **Member Fields**:

Field	Type	Details
problem	string	The optional name value that the course creators supply or the system-generated hash code for the problem being shown. For example, input_303034da25524878a2e66fb57c91cf85_2_1`` or ``303034da25524878a2e66fb57c91cf85_2_1. This value is based on part of the courseware_studentmodule.module_id. See *Columns in the courseware_studentmodule Table*. **History**: In October 2014, identifiers for some new courses began to use the format shown above. Other new courses, and all courses created prior to October 2014, use an HTML-escaped version of the courseware_studentmodule.module_id. For example, i4x://MITx/6.00x/problem/L15:L15_Problem_2.

reset_problem

The server emits reset_problem events when a problem has been reset successfully.

Event Source: Server

event **Member Fields**:

Field	Type	Details
new_state	object	New problem state.
old_state	object	The state of the problem before the reset was performed.
problem_id	string	ID of the problem being reset.

reset_problem_fail

The server emits reset_problem_fail events when a problem cannot be reset successfully.

Event Source: Server

event **Member Fields**:

Field	Type	Details
failure	string	'closed', 'not_done'
old_state	object	The state of the problem before the reset was requested.
problem_id	string	ID of the problem being reset.

save_problem_fail

The server emits save_problem_fail events when a problem cannot be saved successfully.

Event Source: Server

event **Member Fields**:

Field	Type	Details
answers	object	
failure	string	'closed', 'done'
problem_id	string	ID of the problem being saved.
state	object	Current problem state.

save_problem_success

The server emits save_problem_success events when a problem is saved successfully.

Event Source: Server

event **Member Fields**:

Field	Type	Details
answers	object	
problem_id	string	ID of the problem being saved.
state	object	Current problem state.

showanswer

The server emits showanswer events when the answer to a problem is shown.

Event Source: Server

event **Member Fields**:

Field	Type	Details
problem_id	string	EdX ID of the problem being shown.

11.3.7 Library Interaction Events

This section includes descriptions of the following events.

Course teams in an organization can collaboratively contribute to libraries of content, such as a collection of problem components for a particular subject. Libraries are created and maintained separately from courses so that their content can be used in different courses.

In a course outline, course teams can include randomized content block components that reference a library and deliver its content to students. In a randomized content block component, the course team defines how many of the library components to deliver to each student.

For more information, see Working with Content Libraries.

edx.librarycontentblock.content.assigned

The server emits an `edx.librarycontentblock.content.assigned` event the first time that content from a randomized content block is delivered to a user. The `edx.librarycontentblock.content.assigned` event identifies the components delivered from the library to a user.

Additional `edx.librarycontentblock.content.assigned` events can also be emitted if the course team makes a change that results in an increase in the number of components that the randomized content block delivers. After such a change, the randomized content block delivers more components to any user who revisits it after that change. For those users, the `edx.librarycontentblock.content.assigned` event identifies the complete set of components delivered from the library and also the components that were delivered for the first time.

Event Source: Server

History Added 18 Mar 2015.

`event` **Member Fields**:

Field	Type	Details
`added`	list	Lists the library components that were delivered to the user for the first time. The content of this field is different from the content of the `result` field only if the user revisited the randomized content block and it delivered additional components from the library.
`location`	string	Identifies the randomized content block component.
`max_count`	number	The **Count** specified by a course team member in Studio. Defines the number of library components to deliver. This number is greater than the number of library components listed by the `result` field only when the library has too few matching blocks available.
`previous_count`	number	The number of components assigned to this student before this event. The first time the user views the randomized content block, this value is 0.
`result`	list	Lists all of the library components delivered to the user. • `descendants`, when present, is a list that identifies each part of a library component that contains multiple parts (the children of an XBlock with children). • `original_usage_key` and `original_usage_version` identify the component in the library. When students attempt a problem component delivered by a randomized content block, the resulting problem events also reference the `original_usage_key` and `original_usage_version` in `context.module` member fields. See *context Field*. • `usage_key` identifies the location of this component in the course. This value identifies a child of the randomized content block component. To identify a component consistently within a course, you can use either `usage_key` or `original_usage_key` as a consistent identifier. To identify components across courses, use `orignal_usage_key`.

edx.librarycontentblock.content.removed

The server emits an edx.librarycontentblock.content.removed event when a user revisits a randomized content block and one or more of the components that were previously delivered to that user can no longer be delivered.

- If components are removed from the library and the course team resynchronizes the randomized content block to the library, the server emits an edx.librarycontentblock.content.removed event if a user who was previously assigned one of those components revisits the randomized content block or accesses the progress page.
- If the course team changes settings for the randomized content block so that fewer or different components are allowed.

  For example, the course team reduces the number of library components to deliver or specifies a different type of problem to deliver.

Event Source: Server

History Added 18 Mar 2015.

event **Member Fields**:

The edx.librarycontentblock.content.removed events include the following event member fields. These fields serve the same purpose for events of this type as for the edx.librarycontentblock.content.assigned events.

- location
- max_count
- previous_count
- result

The following additional event member fields apply specifically to edx.librarycontentblock.content.removed events.

Field	Type	Details
reason	string	'overlimit' if a course team member reduces the **Count** of library components to deliver. 'invalid' if the component is no longer included in the library, or no longer matches the settings specified for the randomized content block.
removed	list	Identifies the components that are no longer delivered to this user. This field contains the same member fields as the event.result field for edx.librarycontentblock.content.assigned events.

11.3.8 Discussion Forum Events

This section includes descriptions of the following events.

The server emits discussion forum events when a user interacts with a course discussion. This section presents the discussion forum events alphabetically. However, several of these events have hierarchical or sequential relationships.

- When a user creates a new thread, such as a student asking a question, the server emits an *edx.forum.thread.created* event.
- When a user responds to a thread, such as another student answering the question, the server emits an *edx.forum.response.created* event.
- When a user adds a comment to a response, such as a course team member adding a clarification to the student answer, the server emits an *edx.forum.comment.created* event.

These events are emitted and included in daily event logs in addition to the MongoDB discussion forums database data that is included in the weekly database data files. For information about the discussion forums database, see *Discussion Forums Data*.

If a thread, response, or comment was part of a team discussion within a course, a `team_id` field is also included in events for creation or voting interactions. The `team_id` identifies the team that triggered the discussion event. For more information about events for teams, see *Teams-Related Events*.

edx.forum.comment.created

Users create a comment about a response by entering text and then submitting the contributions. When these actions are complete, the server emits an `edx.forum.comment.created` event.

Component: Discussion

Event Source: Server

History: Added 5 Mar 2015.

`event` **Member Fields**:

The `edx.forum.comment.created` events include many of the same `event` member fields that are described for *edx.forum.thread.created* and *edx.forum.response.created* events. The following member fields serve the same purpose for comments as they do for threads or responses.

- `body`
- `commentable_id`
- `discussion`
- `id`
- `options`
- `team_id`
- `truncated`
- `url`
- `user_course_roles`
- `user_forums_roles`

Field	Type	Details
`response`	object	Contains a member `id` field with the unique identifier of the response that the user added this comment to.

edx.forum.response.created

Users create a reply to a post by clicking **Add a Response** and then submitting their contributions. When these actions are complete, the server emits an `edx.forum.response.created` event.

Component: Discussion

Event Source: Server

History: Added 5 Mar 2015.

`event` **Member Fields**:

The `edx.forum.response.created` events include many of the same `event` member fields that are described for *edx.forum.thread.created* events. The following member fields serve the same purpose for responses as they do for threads.

- `body`
- `commentable_id`
- `id`
- `options`
- `team_id`
- `truncated`
- `url`
- `user_course_roles`
- `user_forums_roles`

The following additional `event` member field applies specifically to `edx.forum.response.created` events.

Field	Type	Details
`discussion`	object	Contains a member `id` field with the unique identifier of the thread that the user responded to. Also present for `edx.forum.comment.created` events.

`edx.forum.response.voted`

Users can indicate interest in a response by selecting a "Vote" icon. The "Vote" icon is a toggle, so users can also clear a vote made previously. When either of these actions is complete, the server emits an `edx.forum.response.voted` event.

In these events, the user who voted for the response is identified in the `username` and `context.user_id` fields, and the user who originally posted the thread is identified in the `event.target_username` field.

Component: Discussion

Event Source: Server

History: Added 1 Dec 2015.

`event` **Member Fields**:

The `edx.forum.response.voted` events include the same `event` member fields as *edx.forum.thread.voted* events. The following member fields serve the same purpose for votes on a response as they do for votes on a thread.

- `commentable_id`
- `id`
- `target_username`
- `team_id`
- `undo_vote`
- `url`

- user_course_roles
- user_forums_roles
- vote_value

edx.forum.searched

After a user executes a text search in the navigation sidebar of the course **Discussion** page, the server emits an edx.forum.searched event.

Component: Discussion

Event Source: Server

History: Added 16 May 2014. The corrected_text field was added 5 Jun 2014. The group_id field was added 7 October 2014.

event **Member Fields**:

Field	Type	Details
corrected_text	string	A re-spelling of the query, suggested by the search engine, which was automatically substituted for the original one. This happens only when there are no results for the original query, but the index contains matches for a similar term or phrase. Otherwise, this field is null.
group_id	number	The numeric ID of the cohort to which the user's search is restricted, or null if the search is not restricted in this way. In a course with cohorts enabled, a student's searches will always be restricted to the student's cohort. Discussion admins, moderators, and Community TAs in such a course can search all discussions without specifying a cohort, which leaves this field null, or they can specify a single cohort to search.
page	number	Results are returned in sets of 20 per page. Identifies the page of results requested by the user.
query	string	The text entered into the search box by the user.
total_results	number	The total number of results matching the query.

edx.forum.thread.created

Users create a new top-level thread, also known as a post, by clicking **New Post** and then submitting their contributions. When these actions are complete, the server emits an edx.forum.thread.created event.

Component: Discussion

Event Source: Server

History: Added 5 Mar 2015.

event **Member Fields**:

Field	Type	Details
anonymous	Boolean	Applies only to courses that allow discussion posts that are anonymous to all other users. 'true' only if the user selected the **post anonymously** check box.
anonymous_to_peers	Boolean	Applies only to courses that allow discussion posts that are anonymous to other students. The username of the thread creator is visible only to users who have discussion management privileges. 'true' only if the user selected the **post anonymously to classmates** check box.
body	string	The text supplied for the new post. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
category_id	string	Identifier for the specific discussion component or top-level, course-wide discussion. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
category_name	string	The display name for the specific discussion component or top-level, course-wide discussion. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
commentable_id	string	Identifier for the specific discussion component or top-level, course-wide discussion. Duplicates the `category_id`. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
group_id	string	The numeric ID of the cohort to which the contribution is restricted, or `null` if the contribution is not restricted to a specific cohort.
id	string	A unique identifier for this forum contribution. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
options	object	Contains the `followed` Boolean, which identifies whether the user elected to track the responses that others make to this post. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
team_id	string	If the thread is part of a team discussion within a course, this field identifies the team that the thread was created in. For more information about events for teams, see *Teams-Related Events*.
thread_type	string	The person who creates the thread specifies either 'discussion' or 'question' to characterize the purpose of the post.
title	string	The brief descriptive text supplied to identify the post.
truncated	Boolean	'true' only if the post was longer than 2000 characters, which is the maximum included in the event. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
url	string	The escaped URL of the page the user was visiting when this event was emitted. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
user_course_roles	array	Identifies the course-level 'Instructor' (that is, Admin) or 'Staff' privilege assigned to the user. No value is reported for students. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.
user_forums_roles	array	Identifies a user who does not have discussion management privileges as a 'Student'. Identifies users who have discussion management privileges as a course 'Community TA', 'Moderator', or 'Administrator'. Also present for `edx.forum.response.created` and `edx.forum.comment.created` events.

edx.forum.thread.voted

Users can indicate interest in a thread by selecting a "Vote" icon. The "Vote" icon is a toggle, so users can also clear a vote made previously. When either of these actions is complete, the server emits an `edx.forum.thread.voted` event.

In these events, the user who voted for the thread is identified in the `username` and `context.user_id` fields, and the user who originally posted the thread is identified in the `event.target_username` field.

Component: Discussion

Event Source: Server

History: Added 1 Dec 2015.

`event` **Member Fields**:

The `edx.forum.thread.voted` events include many of the same `event` member fields that are described for *edx.forum.thread.created* events. The following member fields serve the same purpose for votes on a thread as they do for thread creation.

- `commentable_id`
- `id`
- `team_id`
- `url`
- `user_course_roles`
- `user_forums_roles`

The following additional `event` member fields apply to `edx.forum.thread.voted` events.

Field	Type	Details
`target_username`	string	Identifies the user who originally posted the thread. Also present for `edx.forum.response.voted` events, where it indicates the user who originally made the response.
`undo_vote`	Boolean	'true' if the user clears selection of the "Vote" icon made previously. 'false' if the user selects the "Vote" icon. Also present for `edx.forum.response.voted` events.
`vote_value`	string	Set to 'up' for all `edx.forum.thread.voted` events. In the user interface, users can only vote for ("up vote") a thread or clear a previous vote. They cannot vote against ("down vote") a thread. Also present for `edx.forum.response.voted` events

11.3.9 Open Response Assessment Events

This section includes descriptions of the following events.

- *openassessmentblock.get_peer_submission*
- *openassessmentblock.peer_assess and openassessmentblock.self_assess*
- *openassessmentblock.submit_feedback_on_assessments*
- *openassessmentblock.create_submission*
- *openassessmentblock.save_submission*
- *openassessment.student_training_assess_example*
- *openassessment.upload_file*

In an open response assessment, students review a question and then submit a text response and, optionally, an image, .pdf, or other file. To evaluate their own and one or more other students' responses to the questions, students use a scoring rubric designed by the course team. For more information about open response assessments, see Create an Open Response Assessment Assignment.

Component: Open Response Assessments

History: The open response assessment feature was released in August 2014; limited release of this feature began in April 2014.

openassessmentblock.get_peer_submission

After students submit their own responses for evaluation, they use the scoring rubric to evaluate the responses of other course participants. The server emits this event when a response is delivered to a student for evaluation.

Event Source: Server

History: Added 3 April 2014.

`event` **Member Fields**:

Field	Type	Details
`course_id`	string	The identifier of the course that includes this assessment. For open response assessment problems, the course ID is stated in {org}/{course}/{run} format. (For courses created after mid-2014, the course ID is converted to this format for open response assessment problems only.)
`item_id`	string	The i4x:// style locator that identifies the problem in the course.
`requesting_student_id`	string	The course-specific anonymized user ID of the student who requested the response.
`submission_returned_uuid`	string	The unique identifier of the response that the student retrieved for assessment. If no assessment is available, this is set to "None".

openassessmentblock.peer_assess and openassessmentblock.self_assess

The server emits this event when a student either submits an assessment of a peer's response or submits a self-assessment of her own response.

Event Source: Server

History: Added 3 April 2014.

`event` **Member Fields**:

Field	Type	Details
feedback	string	The student's comments about the submitted response.
parts: [criterion, option, feedback]	array	The parts field contains member fields for each criterion in the rubric, the option that the student selected for it, and any feedback comments that the student supplied. These member fields are repeated in an array to include all of the rubric's criteria. • criterion (object) contains points possible and name member fields. • option (string). • feedback (string). When the only criterion in the rubric is student feedback, points possible is 0 and the option field is not included.
rubric	object	This field contains the member field contenthash, which identifies the rubric that the student used to assess the response.
scored_at	datetime	Timestamp for when the assessment was submitted.
scorer_id	string	The course-specific anonymized user ID of the student who submitted this assessment.
score_type	string	"PE" for a peer evaluation, "SE" for a self evaluation.
submission_uuid	string	The unique identifier for the submitted response.

openassessmentblock.submit_feedback_on_assessments

The server emits this event when a student submits a suggestion, opinion, or other feedback about the assessment process.

Event Source: Server

History: Added 3 April 2014.

event **Member Fields**:

Field	Type	Details
feedback_text	string	The student's comments about the assessment process.
options	array	The label of each check box option that the student selected to evaluate the assessment process.
submission_uuid	string	The unique identifier of the feedback.

openassessmentblock.create_submission

The server emits this event when a student submits a response. The same event is emitted when a student submits a response for peer assessment or for self assessment.

Event Source: Server

History: Added 3 April 2014.

`event` **Member Fields**:

Field	Type	Details
`answer`	object	This field contains a `text` (string) member field for the response. For responses that also include an image, .pdf, or other file, this field contains a `file_upload_key` (string) member field with the AWS S3 key that identifies the location of the uploaded file on the Amazon S3 storage service. This key is provided for reference only.
`attempt_number`	number	This value is currently always set to 1.
`created_at`	datetime	Timestamp for when the student submitted the response.
`submitted_at`	datetime	Timestamp for when the student submitted the response. This value is currently always the same as `created_at`.
`submission_uuid`	string	The unique identifier of the response.

openassessmentblock.save_submission

The server emits this event when a student saves a response. Students save responses before they submit them for assessment.

Event Source: Server

History: Added 3 April 2014.

`event` **Member Fields**:

Field	Type	Details
`saved_response`	object	This field contains a `text` (string) member field for the response. For responses that also include an image, .pdf, or other file, this field contains a `file_upload_key` (string) member field with the AWS S3 key that identifies the location of the uploaded file on the Amazon S3 storage service. This key is provided for reference only.

openassessment.student_training_assess_example

The server emits this event when a student submits an assessment for an example response. To assess the example, the student uses a scoring rubric provided by the course team. These events record the options the student selected to assess the example and identifies any criteria that the student scored differently than the course team.

Event Source: Server

History: Added 6 August 2014.

`event` **Member Fields**:

Field	Type	Details
corrections	object	A set of name/value pairs that identify criteria for which the student selected a different option than the course team, in the format `criterion_name: course-team-defined_option_name`.
options_selected	object	A set of name/value pairs that identify the option that the student selected for each criterion in the rubric, in the format `'criterion_name': 'option_name'`.
submission_uuid	string	The unique identifier of the response. Identifies the student who is undergoing training.

openassessment.upload_file

The browser emits this event when a student successfully uploads an image, .pdf, or other file as part of a response. Students complete the upload process before they submit the response.

Event Source: Browser

History: Added 6 August 2014.

`event` **Member Fields**:

Field	Type	Details
fileName	string	The name of the uploaded file, as stored on the student's client machine.
fileSize	number	The size of the uploaded file in bytes. Reported by the student's browser.
fileType	string	The MIME type of the uploaded file. Reported by the student's browser.

11.3.10 Poll and Survey Events

This section describes events emitted by the poll and survey XBlocks.

History: Added 8 Jul 2015.

xblock.poll.submitted

The server emits an `xblock.poll.submitted` event each time a user submits a response to a poll.

Event Source: Server

`event` **Member Fields**:

Field	Type	Details
url_name	string	The unique location identifier for the poll XBlock.
choice	string	The unique internal identifier for the response that the user submitted.

xblock.poll.view_results

The server emits an `xblock.poll.view_results` event when a tally of the responses to a poll is displayed to a user. For a poll that has the **Private Results** option set to False, the tally appears after a user submits a response.

Event Source: Server

`event` **Member Fields**: None

xblock.survey.submitted

The server emits an `xblock.survey.submitted` event each time a user submits responses to a survey.

Event Source: Server

`event` **Member Fields**:

Field	Type	Details
`url_name`	string	The unique location identifier for the survey XBlock.
`choices`	object	Name/value pairs that identify each question in the survey and the responses that the user selected, in the format `"question_name": "'response_name"`.

xblock.survey.view_results

The server emits an `xblock.survey.view_results` event when a matrix of survey response percentages is displayed to a user. For surveys that have the **Private Results** option set to False only, the matrix appears after a user submits survey responses.

Event Source: Server

`event` **Member Fields**: None

11.3.11 Third-Party Content Events

This section includes descriptions of the following events.

EdX courses can include components that present content that is hosted by a third party. The server emits events when users interact with the third-party content.

Google Component Events

Course teams use the Google Calendar and Google Drive Files tools in Studio to embed Google calendars and Google drive files, such as documents, spreadsheets, and images, in a course. When users interact with the files in the LMS, the server emits the following events.

For more information about these tools, see Google Calendar Tool or Google Drive Files Tool.

edx.googlecomponent.calendar.displayed

The server emits an `edx.googlecomponent.calendar.displayed` event when a Google Calendar component is shown in the LMS.

Event Source: Server

History: Added 5 Mar 2015.

`event` **Member Fields**:

Field	Type	Details
`displayed_in`	string	'iframe' for Google Calendars and for Google Drive files of other types. 'img' for Google Drive image files.
`url`	string	The URL of the image file or of the file loaded by the iFrame.

edx.googlecomponent.document.displayed

The server emits an `edx.googlecomponent.document.displayed` event when a Google Drive file, such as a document, spreadsheet, or image, is shown in the LMS.

Event Source: Server

History: Added 5 Mar 2015.

`event` **Member Fields**:

The `edx.googlecomponent.document.displayed` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the `edx.googlecomponent.calendar.displayed` events.

- `displayed_in`
- `url`

Oppia Exploration Events

Course teams can embed short, interactive tutorials created using Oppia in their courses with the Oppia exploration tool. When users interact with the Oppia tutorials, called explorations, in the LMS, the server emits the following events.

- *oppia.exploration.completed*
- *oppia.exploration.loaded*
- *oppia.exploration.state.changed*

For more information about adding Oppia explorations to a course, see Oppia Exploration Tool.

oppia.exploration.completed

The server emits an `oppia.exploration.completed` event when a user completes an interaction with an Oppia exploration component. Oppia explorations do not emit grading events.

Event Source: Server

History: Added 27 Oct 2015.

`event` **Member Fields**:

The `oppia.exploration.completed` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the `oppia.exploration.state.changed` events.

- `exploration_id`
- `exploration_version`

`oppia.exploration.loaded`

The server emits an `oppia.exploration.loaded` event when an Oppia exploration component is shown in the LMS.

Event Source: Server

History: Added 27 Oct 2015.

`event` **Member Fields**:

The `oppia.exploration.loaded` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the `oppia.exploration.state.changed` events.

- `exploration_id`
- `exploration_version`

`oppia.exploration.state.changed`

The server emits an `oppia.exploration.state.changed` event when a user interacts with an Oppia exploration component by submitting an answer. Answers are not incorrect or correct. All answer submissions change the state of the exploration.

Event Source: Server

History: Added 27 Oct 2015.

`event` **Member Fields**:

Field	Type	Details
`exploration_id`	string	The unique identifier of the Oppia exploration.
`exploration_version`	string	The version number for the Oppia exploration.
`new_state_name`	string	The name of the state that the exploration was changed to by the submitted answer.
`old_state_name`	string	The name of the state the exploration was in when the user submitted an answer.

Microsoft Office Mix Events

Course teams can use Office Mix to turn Microsoft PowerPoint presentations in to interactive online lessons, called mixes. They can then use the Office Mix tool in Studio to include mixes in a course. When users interact with the Office Mix player in the LMS, the server emits the following events.

- *microsoft.office.mix.loaded*
- *microsoft.office.mix.paused*
- *microsoft.office.mix.played*
- *microsoft.office.mix.slide.loaded*
- *microsoft.office.mix.stopped*

For more information about adding mixes to a course, see Office Mix Tool.

microsoft.office.mix.loaded

The server emits a `microsoft.office.mix.loaded` event when a mix is fully loaded and ready to play in the Office Mix player in the LMS.

Event Source: Server

History: Added 1 Dec 2015.

`event` **Member Fields**:

Field	Type	Details
`duration`	number	The total length of the mix, in seconds.
`total_slides`	number	The total number of slides in the mix.
`url`	string	The URL of the embedded mix, in the format "https://mix.office.com/embed/10g8h9rvi1yg8".

microsoft.office.mix.paused

The server emits a `microsoft.office.mix.paused` event when a user selects **pause** for an Office Mix.

Event Source: Server

History: Added 1 Dec 2015.

`event` **Member Fields**:

The `microsoft.office.mix.paused` events include the following `event` member field. This field serves the same purpose for events of this type as for the `microsoft.office.mix.loaded` events.

- `url`

The following additional `event` member fields apply specifically to `microsoft.office.mix.paused` events.

Field	Type	Details
`current_slide`	number	The slide presented to the user when the user chose to pause the mix.
`current_time`	number	The relative time in the video, in seconds, when the user chose to pause the mix.

microsoft.office.mix.played

The server emits a `microsoft.office.mix.played` event when a user selects **play** for an Office Mix.

Event Source: Server

History: Added 1 Dec 2015.

`event` **Member Fields**:

The `microsoft.office.mix.played` events include the following `event` member fields. These fields serve the same purpose for events of this type as for the `microsoft.office.mix.paused` events.

- `current_slide`
- `current_time`
- `url`

`microsoft.office.mix.slide.loaded`

The server emits a `microsoft.office.mix.slide.loaded` event each time the Office Mix player changes the slide that is presented to the user.

`event` **Member Fields**:

The `microsoft.office.mix.slide.loaded` events include the following `event` member field. This field serves the same purpose for events of this type as for the `microsoft.office.mix.loaded` events.

- `url`

The following additional `event` member field applies specifically to `microsoft.office.mix.slide.loaded` events.

Field	Type	Details
`slide`	number	The slide presented to the user.

`microsoft.office.mix.stopped`

The server emits an `microsoft.office.mix.stopped` event when the Office Mix player reaches the end of a mix and automatically stops.

Event Source: Server

History: Added 1 Dec 2015.

`event` **Member Fields**:

The `microsoft.office.mix.stopped` events include the following `event` member field. This field serves the same purpose for events of this type as for the `microsoft.office.mix.loaded` events.

- `url`

11.3.12 Testing Events for Content Experiments

This section includes descriptions of the following events.

Course authors can configure course content to present modules that contain other modules. Content experiments, also known as A/B or split tests, use this structure. For example, a parent module can include two child modules that contain content that differs in some way for comparison testing.

- Internally, a *partition* defines the type of experiment: comparing the effectiveness of video alone to text alone, for example. A course can include any number of modules that have the same partition or experiment type.

- For each partition, students are randomly assigned to a *group*. The group determines which content, either video or text in this example, is shown by every module with that partitioning.
- Students are assigned to groups randomly. Assignment to a group takes place when student navigation through the course requires data from that module. For example, one student is assigned to a group when he visits the course progress page, while another student is assigned to a group when she visits a course component that is the parent module of a content experiment. Based on this random group assignment, the content of just one of the two child modules is shown to the student.
- For investigations into which students in each group actually interacted with tested content, review the events for the behavior you want to learn about. For example, review the students' `play_video` , `textbook.pdf.page.navigated`, or `problem_check` events.

The events that follow apply to modules that are set up to randomly assign students to groups so that different content can be shown to the different groups.

For more information about how course teams add content experiments to their courses, see Add Content Experiments to Your Course.

History: These events were added on 12 Mar 2014.

xmodule.partitions.assigned_user_to_partition

When a student views a module that is set up to test different child modules, the server checks the `user_api_usercoursetag` table for the student's assignment to the relevant partition, and to a group for that partition.

- The partition ID is the `user_api_usercoursetag.key`.
- The group ID is the `user_api_usercoursetag.value`.

If the student does not yet have an assignment, the server emits an `xmodule.partitions.assigned_user_to_partition` event and adds a row to the `user_api_usercoursetag` table for the student. See *Columns in the user_api_usercoursetag Table*.

Note: After this event is emitted, the common `context` field in all subsequent events includes a `course_user_tags` member field with the student's assigned partition and group.

Component: Split Test

Event Source: Browser

`event` **Member Fields**:

Field	Type	Details
`group_id`	number	Identifier of the group.
`group_name`	string	Name of the group.
`partition_id`	number	Identifier for the partition, in the format `xblock.partition_service.partition_ID` where ID is a number.
`partition_name`	string	Name of the partition.

xblock.split_test.child_render

When a student views a module that is set up to test different content using child modules, the server emits a `xblock.split_test.child_render` event to identify the child module that was shown to the student.

Component: Split Test

Event Source: Server

`event` **Member Fields**:

Field	Type	Details
`child_id`	string	ID of the module that was displayed to the student. **History**: Renamed on 16 Oct 2014 from `child-id` to `child_id`.

11.3.13 Student Cohort Events

This section includes descriptions of the following events.

- *edx.cohort.created*
- *edx.cohort.user_added*
- *edx.cohort.user_removed*

For information about including student cohorts in a course, see Using Cohorts in Your Courses in the *Building and Running an edX Course* guide.

edx.cohort.created

When a course team or the system creates a cohort, the server emits an `edx.cohort.created` event. Cohorts can be created manually by members of the course team. The system automatically creates the default cohort and any cohorts that are defined by the `auto_cohort_groups` advanced setting when they are needed (for example, when a student is assigned to one of those cohorts).

Additional events are emitted when members of the course team interact with the Instructor Dashboard to create a cohort. See *Course Team Cohort Events*.

Event Source: Server

History Added 7 Oct 2014.

`event` **Member Fields**:

Field	Type	Details
`cohort_id`	number	The numeric ID of the cohort.
`cohort_name`	string	The display name of the cohort.

edx.cohort.user_added

When a user is added to a cohort, the server emits an `edx.cohort.user_added` event. Members of the course team can add users to cohorts individually or by uploading a .csv file of student cohort assignments. The system automatically adds a user to the default cohort or a cohort included in the course's `auto_cohort_groups` setting if a user who has not yet been assigned to a cohort accesses course content.

Additional events are emitted when members of the course team interact with the Instructor Dashboard to add a user to a cohort. See *Course Team Cohort Events*.

Event Source: Server

History Added 7 Oct 2014.

`event` **Member Fields**:

Field	Type	Details
cohort_id	number	The numeric ID of the cohort.
cohort_name	string	The display name of the cohort.
user_id	number	The numeric ID (from auth_user.id) of the added user.

edx.cohort.user_removed

When a course team member changes the cohort assignment of a user on the Instructor Dashboard, the server emits an edx.cohort.user_removed event.

Event Source: Server

History Added 7 Oct 2014.

event **Member Fields**:

Field	Type	Details
cohort_id	number	The numeric ID of the cohort.
cohort_name	string	The display name of the cohort.
user_id	number	The numeric ID (from auth_user.id) of the removed user.

11.3.14 Teams-Related Events

This section includes descriptions of the following events, which are generated if a course includes teams, and learners or course team members perform particular teams-related actions. This list includes both student events and course team events, because some of these events are triggered by actions that can be performed both by students and course staff (with the Staff, Admin, Discussion Admin or Discussion Moderator roles), or by students with special roles such as Community TAs.

This section presents teams-related events alphabetically. Typically, the first event produced when teams are included in a course is the edx.team.created event.

Note: The Teams feature is in limited release. For more information, contact your edX partner manager. For Open edX sites, contact your system administrator.

edx.team.activity_updated

When team discussion activity has occurred on a team, including a team member posting, editing posts, commenting, responding, endorsing, and so on), the server emits an edx.team.activity_updated event.

The definition of activity that would trigger this event does not include changes in team membership.

Event Source: Server

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member field.

Field	Type	Details
`team_id`	string	The identifier for the team.

edx.team.changed

When a team's information is edited, the server emits one `edx.team.team_changed` event for each modified field.

Event Source: Server

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member fields.

Field	Type	Details
`field`	string	The name of the field within the team's details that was modified such as team name, description, primary country, or primary language.
`new`	string	The value of the field after the modification. If this value is longer than 1250 characters, the string is truncated, `...` is added at the end of the string, and this field is included in the `truncated` array.
`old`	string	The value of the field before the modification. If this value is longer than 1250 characters, the string is truncated, `...` is added at the end of the string, and this field is included in the `truncated` array.
`truncated`	array	The `truncated` event field is an array of the `old` and `new` fields that have been truncated.

The `edx.team.changed` event also includes the following `event` member field.

- `team_id`

This field serves the same purpose for this event as it does for the *edx.team.activity_updated* event.

edx.team.created

When a team is created, either by a course team member or by a learner, the server emits an `edx.team.created` event.

Event Source: Server

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member field.

This field serves the same purpose for this event as it does for the *edx.team.activity_updated* event.

- `team_id`

edx.team.deleted

When a team is deleted, the server emits an `edx.team.deleted` event. Course team members who have any of the **Staff**, **Admin**, **Discussion Admin**, **Discussion Moderator**, or **Community TA** roles can delete teams.

Event Source: Server

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member field.

This field serves the same purpose for this event as it does for the *edx.team.activity_updated* event.

- `team_id`

edx.team.learner_added

When a user joins a team or is added by someone else, the server emits an `edx.team.learner_added` event.

Event Source: Server

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member fields.

Field	Type	Details
`add_method`	string	The method by which the user joined the team. Possible values are `added_on_create`, `joined_from_team_view`, or `added_by_another_user`.
`user_id`	string	The identifier for the user who joined or was added to the team.

The `edx.team.learner_added` event also includes the following `event` member field.

- `team_id`

This field serves the same purpose for this event as it does for the *edx.team.activity_updated* event.

edx.team.learner_removed

When a user leaves a team or is removed by someone else, the server emits an `edx.team.learner_deleted` event. This event is also triggered when a team is deleted, because all members are removed when a team is deleted.

Course team members who have any of the **Staff**, **Admin**, **Discussion Admin**, **Discussion Moderator**, or **Community TA** roles can remove learners from teams.

Event Source: Server

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member fields.

Field	Type	Details
`remove_method`	string	The method by which the user was removed from the team. Possible values are `self_removal`, `team_deleted`, or `removed_by_admin`.
`user_id`	string	The identifier for the user who left or was removed from the team.

The `edx.team.learner_removed` event also includes the following `event` member field.

- `team_id`

This field serves the same purpose for this event as it does for the *edx.team.activity_updated* event.

edx.team.page_viewed

When a user views any page with a unique URL under the **Teams** page in the courseware, the browser emits an `edx.team.page_viewed` event.

Event Source: Browser

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member fields.

Field	Type	Details
`page_name`	string	The name of the page that was viewed. Possible values are: `browse`, `edit-team`, `my-teams`, `new-team`, `search teams`, `single- team`, and `single-topic`.
`topic_id`	string	The identifier of the topic related to the page that was viewed. This value is set to `null` if a topic is not applicable to the page that was viewed, or if the topic does not exist.

The `edx.team.page_viewed` event also includes the following `event` member field.

- `team_id`

This field serves the same purpose for this event as it does for the *edx.team.activity_updated* event. For the `edx.team.page_viewed` event, the value of this field is set to `null` if the page that was viewed has no applicable team, or if a team does not exist.

edx.team.searched

When a user performs a search for teams from the topic view under the **Teams** page of the courseware, the server emits an `edx.team.searched` event.

Event Source: Server

History Added 16 Sept 2015.

`event` **Member Fields**:

In addition to the *common* `context` member fields, this event type also includes the following `event` member fields.

Field	Type	Details
`number_of_results`	number	The count of results that matched the search text.
`search_text`	string	The text or keywords used in the search.
`topic_id`	string	The identifier for the topic under which this search for teams was performed.

11.3.15 Certificate Events

This section includes descriptions of the events related to certificates, which are awarded to qualified learners when they complete a course.

edx.certificate.created

When a certificate is generated, a record is created in the certificates_generatedcertificate table, triggering an edx.certificate.created event. For details, see *Columns in the certificates_generatedcertificate Table*.

Event Source: Server

History Added 2 September 2015.

event **Member Fields**:

Field	Type	Details
certificate_id	string	The verify.uuid value from the certificates_generatedcertificate table. This string appears at the bottom of each certificate.
certificate_url	string	The URL for the certificate web page.
course_id	string	The course for which this certificate is issued.
enrollment_mode	string	The course enrollment mode associated with this certificate.
generation_mode	string	Indicates whether this certificate was generated for all learners in a course by a batch command, or whether a learner generated her own certificate. Possible values are "batch" and "self".
user_id	number	The numeric ID of the learner who earned this certificate.

edx.certificate.shared

When a learner shares the URL for her certificate on a social media web site, the server emits an edx.certificate.shared event.

Event Source: Browser

History Added 2 September 2015.

event **Member Fields**:

The edx.certificate.shared event includes many of the same event member fields that are described for the edx.certificate.created event. The following member fields serve the same purpose for edx.certificate.shared as they do for *edx.certificate.created*.

- certificate_id
- certificate_url
- course_id
- enrollment_mode
- user_id

The following additional event member field applies specifically to edx.certificate.shared events.

Field	Type	Details
social_network	string	The social network to which the certificate is shared, such as "LinkedIn", "Facebook", or "Twitter".

edx.certificate.evidence_visited

When a learner shares her certificates on social network sites such as LinkedIn, and the link back to the certificate is followed by some visitor to that social network site, the server emits an edx.certificate.evidence_visited event.

Event Source: Browser

History Added 2 September 2015.

`event` **Member Fields**:

The `edx.certificate.evidence_visited` event includes all of the same `event` member fields that are described for the `edx.certificate.created` event. The following member fields serve the same purpose for `edx.certificate.evidence_visited` as they do for `edx.certificate.created`.

- `certificate_id`
- `certificate_url`
- `course_id`
- `enrollment_mode`
- `user_id`

The following additional `event` member fields apply specifically to `edx.certificate.evidence_visited` events.

Field	Type	Details
`social_network`	string	The social network to which the certificate is shared, such as "LinkedIn", "Facebook", or "Twitter".
`source_url`	string	The URL of the web site where the certificate evidence link was selected. This URL is the same as the URI in the `context.referer` field. For details, see *referer field*.

11.3.16 Open Response Assessment Events (Deprecated)

The events described in this section recorded interactions with the prototype implementation of open response assessment (ORA 1) problem types. EdX deprecated this feature in May 2014, and removed the ability to add a new ORA 1 assignment to courses in December 2014.

For more information about events for the current implementation of open response assessments, see *Open Response Assessment Events*.

`oe_hide_question` and `oe_show_question`

The browser emits `oe_hide_question` and `oe_show_question` events when the user hides or redisplays a combined open-ended problem.

History: These events were previously named `oe_hide_problem` and `oe_show_problem`.

Component: Combined Open-Ended

Event Source: Browser

`event` **Member Fields**:

Field	Type	Details
`location`	string	The location of the question whose prompt is being shown or hidden.

`rubric_select`

Component: Combined Open-Ended

Event Source: Browser

`event` **Member Fields**:

Field	Type	Details
category	number	Rubric category selected.
location	string	The location of the question whose rubric is being selected.
selection	number	Value selected on rubric.

oe_show_full_feedback and oe_show_respond_to_feedback

Component: Combined Open-Ended

Event Source: Browser

event **Member Fields**: None.

oe_feedback_response_selected

Component: Combined Open-Ended

Event Source: Browser

event **Member Fields**:

Field	Type	Details
value	number	Value selected in the feedback response form.

peer_grading_hide_question and peer_grading_show_question

The browser emits peer_grading_hide_question and peer_grading_show_question events when the user hides or redisplays a problem that is peer graded.

History: These events were previously named peer_grading_hide_problem and peer_grading_show_problem.

Component: Peer Grading

Event Source: Browser

event **Member Fields**:

Field	Type	Details
location	string	The location of the question whose prompt is being shown or hidden.

staff_grading_hide_question and staff_grading_show_question

The browser emits staff_grading_hide_question and staff_grading_show_question events when the user hides or redisplays a problem that is staff graded.

History: These events were previously named staff_grading_hide_problem and staff_grading_show_problem.

Component: Staff Grading

Event Source: Browser

event **Member Fields**:

Field	Type	Details
location	string	The location of the question whose prompt is being shown or hidden.

11.4 Course Team Events

This section lists events that are generated by interactions with the Instructor Dashboard in the LMS.

The schema definitions for events include only the JSON fields that are common to all events follow.

- `dump-answer-dist-csv`
- `dump-graded-assignments-config`
- `dump-grades`
- `dump-grades-csv`
- `dump-grades-csv-raw`
- `dump-grades-raw`
- `list-beta-testers`
- `list-instructors`
- `list-staff`
- `list-students`

Event Source: Server

For more information about the common fields, see *Common Fields*.

Course team events that have additional `context` or `event` member fields follow.

11.4.1 `add-instructor` and `remove-instructor`

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`instructor`	string

11.4.2 delete-student-module-state and rescore-student-submission

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`course`	string
`problem`	string
`student`	string

11.4.3 edx.instructor.report.downloaded

The browser emits an `edx.instructor.report.downloaded` event when the user clicks a report link on the Instructor Dashboard to download a report.

History: Added 8 May 2015.

Component: Instructor Dashboard

Event Source: Browser

`event` **Member Fields**:

Field	Type	Details
`report_url`	string	The URL to the report file.

11.4.4 edx.instructor.report.requested

The server emits an `edx.instructor.report.requested` event when the user clicks to request the generation of a report on the Instructor Dashboard.

History: Added 8 May 2015.

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type	Details
`report_type`	string	The type of report that was requested.

11.4.5 get-student-progress-page

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`course`	string
`instructor`	string
`student`	string

11.4.6 `rescore-all-submissions` and `reset-all-attempts`

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`course`	string
`problem`	string

11.4.7 `reset-student-attempts`

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`course`	string
`old_attempts`	string
`problem`	string
`student`	string

11.4.8 List Discussion Team Events

- `list-forum-admins`
- `list-forum-community-TAs`
- `list-forum-mods`

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`course`	string

11.4.9 Manage Discussion Team Events

- `add-forum-admin`
- `add-forum-community-TA`
- `add-forum-mod`
- `remove-forum-admin`
- `remove-forum-community-TA`
- `remove-forum-mod`

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`course`	string
`username`	string

11.4.10 `psychometrics-histogram-generation` (Deprecated)

Component: Instructor Dashboard

Event Source: Server

History: The chart feature intended to emit these events was never enabled on the edX Edge or edx.org Instructor Dashboard.

`event` **Member Fields**:

Field	Type
`problem`	string

11.4.11 `add-or-remove-user-group`

Component: Instructor Dashboard

Event Source: Server

`event` **Member Fields**:

Field	Type
`event`	string
`event_name`	string
`user`	string

11.4.12 Instructor Enrollment Events

In addition to the enrollment events that are generated when students enroll in or unenroll from a course, actions by course team members also generate enrollment events.

- When a course author creates a course, his or her user account is enrolled in the course and the server emits an `edx.course.enrollment.activated` event.
- When a user with the Admin or Staff role enrolls in a course, the server emits `edx.course.enrollment.activated`. The server emits `edx.course.enrollment.deactivated` events when these users unenroll from a course.
- When a user with the Admin or Staff role uses the **Batch Enrollment** feature to enroll students or other course team members in a course, the server emits an `edx.course.enrollment.activated` event for each enrollment. When this feature is used to unenroll students from a course, the server emits a `edx.course.enrollment.deactivated` for each unenrollment.

  For events emitted as a result of a batch enrollment, the `username` and `context.user_id` identify the course team member who made the change, and the `event.user_id` identifies the student who was enrolled or unenrolled.

For details about the enrollment events, see *Enrollment Events*.

11.4.13 Course Team Cohort Events

In addition to the cohort events that are generated when cohorts are created and users are assigned to them (see *Student Cohort Events*), actions by course team members also generate cohort-related events.

For more information about student cohorts, see Using Cohorts in Your Courses in the *Building and Running an edX Course* guide.

edx.cohort.creation_requested

When a course team member manually creates a cohort on the Instructor Dashboard, the server emits an `edx.cohort.creation_requested` event.

Event Source: Server

History Added 7 Oct 2014.

`event` **Member Fields**:

Field	Type	Details
`cohort_id`	number	The numeric ID of the cohort.
`cohort_name`	string	The display name of the cohort.

edx.cohort.user_add_requested

When a course team member adds a student to a cohort on the Instructor Dashboard, the server emits an `edx.cohort.user_add_requested` event. Course team members can add students to a cohort individually, or by uploading a .csv file of student cohort assignments.

Event Source: Server

History Added 7 Oct 2014.

`event` **Member Fields**:

Field	Type	Details
`cohort_id`	number	The numeric ID of the cohort.
`cohort_name`	string	The display name of the cohort.
`previous_cohort_id`	number	The numeric ID of the cohort that the user was previously assigned to. Null if the user was not previously assigned to a cohort.
`previous_cohort_name`	string	The display name of the cohort that the user was previously assigned to. Null if the user was not previously assigned to a cohort.
`user_id`	number	The numeric ID (from `auth_user.id`) of the added user.

CHAPTER 12

Glossary

A - C - D - E - F - G - H - I - K - L - M - N - O - P - R - S - T - V - W - XYZ

12.1 A

A/B Test

See *Content Experiment*.

About Page

The course page that provides potential students with a course summary, prerequisites, a course video and image, and important dates.

For more information, see The Course Summary Page.

Accessible Label

The descriptive, identifying name that you supply when you add a problem component to your course. All problems require accessible labels.

For more information, see Creating Exercises and Tools.

Advanced Editor

An XML-only editor in a problem component that allows you to can create and edit any type of problem. For more information, see The Advanced Editor.

Assignment Type

The category of graded student work, such as homework, exams, and exercises.

For more information, see Establishing a Grading Policy.

12.2 C

Capa Problem

Any of the problem types implemented in the edX platform by the `capa_module` XBlock. Examples range from text input, drag and drop, and math expression input problem types to circuit schematic builder, custom JavaScript, and chemical equation problem types.

Other assessment methods are also available, and implemented using other XBlocks. An open response assessment is an example of a non-capa problem type.

Certificate

A document issued to an enrolled student who successfully completes a course. Not all edX courses offer certificates, and not all students enroll as certificate candidates.

Chapter

See *Section*.

Checkbox Problem

A problem that prompts the student to select one or more options from a list of possible answers. For more information, see Checkbox Problem.

Chemical Equation Response Problem

A problem that allows the student to enter chemical equations as answers. For more information, see Chemical Equation Problem.

Circuit Schematic Builder Problem

A problem that allows the student to construct a schematic answer (such as an electronics circuit) on an interactive grid.

For more information, see Circuit Schematic Builder Problem.

Closed Captions

See *Transcript*.

Cohort

A group of students who participate in a class together. Students who are in the same cohort group can communicate and share experiences in private discussions.

Cohorts are an optional feature of courses on the edX platform. For information about how you enable the cohort feature, set up cohorts, and assign students to them, see Using Cohorts in Your Courses.

Component

The part of a unit that contains your actual course content. A unit can contain one or more components. For more information, see Developing Course Components.

Content Experiment

You can define alternative course content to be delivered to different, randomly assigned groups of students. Also known as A/B or split testing, you use content experiments to compare the performance of students who have been exposed to different versions of the content. For more information, see Creating Content Experiments.

Content Library

See *Library*.

Content-Specific Discussion Topic

A category within the course discussion that appears at a defined point in the course to encourage questions and conversations. To add a content-specific discussion topic to your course, you add a discussion component to a unit. Students cannot contribute to a content-specific discussion topic until the release date of the section that contains it.

For more information, see Working with Discussion Components and Creating Discussion Topics for Your Course.

Course Catalog

The page that lists all courses offered in the edX learning management system.

Course Handouts

Course handouts are files you make available to students in the Course Info page.

For more information, see Add Course Handouts.

Course Info Page

The page that opens first every time students access your course. You can post announcements on the Course Info page. The Course Handouts sidebar appears in the right pane of this page.

Course Run

The term or time frame in which a specific offering of your course takes place. You set the course run when you create your course. For more information, see Create a New Course.

Courseware

The page where students access the primary instructional materials for your course. Sections, subsections, units, and components are all accessed from the Courseware page.

Course-Wide Discussion Topic

Optional categories that you create to guide how students find and share information in the course discussion. Examples of course-wide discussion topics include Announcements and Frequently Asked Questions. Students can contribute to these topics as soon as your course starts.

For more information, see Creating Discussion Topics for Your Course.

Custom Response Problem

A custom response problem evaluates text responses from students using an embedded Python script. These problems are also called "write-your-own- grader" problems. For more information, see Write-Your-Own-Grader Problem.

12.3 D

Data Czar

A data czar is the single representative at a partner institution who is responsible for receiving course data from edX, and transferring it securely to researchers and other interested parties after it is received.

For more information, see the edX Research Guide.

Discussion

The set of topics defined to promote course-wide or unit-specific dialog. Students use the discussion topics to communicate with each other and the course team in threaded exchanges.

For more information, see Managing Course Discussions.

Discussion Component

Discussion topics that course teams add directly to units. For example, a video component can be followed by a discussion component so that students can discuss the video content without having to leave the page. When you add a discussion component to a unit, you create a content-specific discussion topic.

For more information, see Working with Discussion Components.

Dropdown Problem

A problem that asks students to choose from a collection of answer options, presented as a drop-down list. For more information, see Dropdown Problem.

12.4 E

edX101

An online course about how to create online courses. The intended audience for edX101 is faculty and university administrators.

edX Edge

Edge is a less restricted site than edX.org. While only edX employees and consortium members can create and post content on edX.org, any users with course creator permissions for Edge can create courses with Studio on studio.edge.edx.org, then view the courses on the learning management system at edge.edx.org.

edX Studio

The edX tool that you use to build your courses.

For more information, see What is Studio?.

Exercises

Practice or practical problems interspersed in edX course content to keep the learner engaged. Exercises are also an important measure of teaching effectiveness and learner comprehension.

Export

A tool in edX Studio that you use to export your course or library for backup purposes, or so that you can edit the course or library directly in XML format. See also *Import*.

For more information, see Export a Course or Export a Library.

12.5 F

Forum

See *Discussion*.

12.6 G

Grade Range

Thresholds that specify how numerical scores are associated with grades, and the score a student must obtain to pass a course.

For more information, see Set the Grade Range.

Grading Rubric

See *Rubric*.

12.7 H

HTML Component

A type of component that you can use to add and format text for your course. An HTML component can contain text, lists, links, and images.

For more information, see Working with HTML Components.

12.8 I

Image Mapped Input Problem

A problem that presents an image and accepts clicks on the image as an answer.

For more information, see Image Mapped Input Problem.

Import

A tool in edX Studio that you use to load a course or library in XML format into your existing course or library. When you use the Import tool, Studio replaces all of your existing course or library content with the content from the imported course or library. See also *Export*.

For more information, see Import a Course or Import a Library.

12.9 K

Keyword

A variable in a bulk email message. When you send the message, a value that is specific to the each recipient is substituted for the keyword.

12.10 L

Label

See *Accessible Label*.

LaTeX

A document markup language and document preparation system for the TeX typesetting program.

In edX Studio, you can import LaTeX code.

You can also create a problem written in LaTeX.

Learning Management System (LMS)

The platform that students use to view courses, and that course team members use to manage learner enrollment, assign team member privileges, moderate discussions, and access data while the course is running.

Learning Sequence

The horizontal navigation bar that appears at the top of the **Courseware** page in the LMS. The learning sequence contains an icon for each unit in the selected subsection. When a learner moves the cursor over one of these icons, the names of each component in that unit appear.

Left Pane

The navigation frame that appears at the left side of the **Courseware** page in the LMS. The left pane shows the sections in the course. When you click a section, the section expands to show subsections.

Library

A pool of components for use in randomized assignments that can be shared across multiple courses from your organization. Course teams configure randomized content blocks in course outlines to reference a specific library and randomly provide a specified number of problems from that library to each student.

For more information, see Libraries Overview.

Live Mode

A view that allows the course team to review all published units as students see them, regardless of the release dates of the section and subsection that contain the units.

For more information, see View Your Live Course.

LON-CAPA

The LearningOnline Network with Computer-Assisted Personalized Approach e-learning platform. The structure of capa problem types in the edX platform is based on the LON-CAPA assessment system, although they are not compatible.

See *Capa Problems*.

12.11 M

Math Expression Input Problem

A problem that requires students to enter a mathematical expression as text, such as e=m*c^2.

For more information, see Entering Mathematical and Scientific Expressions.

MathJax

A LaTeX-like language that you use to write equations. Studio uses MathJax to render text input such as x^2 and sqrt(x^2-4) as "beautiful math."

For more information, see A Brief Introduction to MathJax in Studio.

Module

An item of course content, created in an XBlock, that appears on the **Courseware** page in the edX learning management system. Examples of modules include videos, HTML-formatted text, and problems.

Module is also used to refer to the structural components that organize course content. Sections, subsections, and units are modules; in fact, the course itself is a top-level module that contains all of the other course content as children.

Multiple Choice Problem

A problem that asks students to select one answer from a list of options.

For more information, see Multiple Choice Problem.

12.12 N

Numerical Input Problem

A problem that asks students to enter numbers or specific and relatively simple mathematical expressions.

For more information, see Numerical Input Problem.

12.13 O

Open Response Assessment

A type of assignment that allows learners to answer with text, as in a short essay and, optionally, an image or other file. Learners then evaluate each others' work by comparing each response to a rubric created by the course team.

These assignments can also include a self assessment, in which learners compare their own responses to the rubric.

For more information, see Open Response Assessments.

12.14 P

Pages

Pages organize course materials into categories that students select in the learning management system. Pages provide access to the courseware and to tools and uploaded files that supplement the course. Each page appears in your course's navigation bar.

For more information, see Adding Pages to a Course.

Partner Manager

Each EdX partner institution has an edX partner manager. The partner manager is the primary contact for the institution's course teams.

Pre-Roll Video

A short video file that plays before the video component selected by the learner. Pre-roll videos play automatically, on an infrequent schedule.

For more information, see Adding a Pre-Roll Video.

Preview Mode

A view that allows you to see all the units of your course as students see them, regardless of the unit status and regardless of whether the release dates have passed.

For more information, see Preview Course Content.

Problem Component

A component that allows you to add interactive, automatically graded exercises to your course content. You can create many different types of problems.

For more information, see Working with Problem Components.

Progress Page

The page in the learning management system that shows students their scores on graded assignments in the course.

12.15 Q

Question

A question is a type of contribution that you can make to a course discussion topic to bring attention to an issue that the discussion moderation team or other students can resolve.

For more information, see Managing Course Discussions.

12.16 R

Rubric

A list of the items that a student's response should cover in an open response assessment.

For more information, see Rubric.

12.17 S

Section

The topmost category in your course outline. A section can represent a time period or another organizing principle for course content. A section contains one or more subsections.

For more information, see Developing Course Sections.

Sequential

See *Subsection*.

Short Course Description

The description of your course that appears on the edX Course List page.

For more information, see Describe Your Course.

Simple Editor

The graphical user interface in a problem component that contains formatting buttons and is available for some problem types. For more information, see The Studio View of a Problem.

Split Test

See *Content Experiment*.

Subsection

A division in the course outline that represents a topic in your course, such as a lesson or another organizing principle. Subsections are defined inside sections and contain units.

For more information, see Developing Course Subsections.

12.18 T

Text Input Problem

A problem that asks the student to enter a line of text, which is then checked against a specified expected answer.

For more information, see Text Input Problem.

Transcript

A text version of the content of a video. You can make video transcripts available to students.

For more information, see Working with Video Components.

12.19 U

Unit

A unit is a division in the course outline that represents a lesson. Learners view all of the content in a unit on a single page.

For more information, see Developing Course Units.

12.20 V

Vertical

See *Unit*.

Video Component

A component that you can use to add recorded videos to your course.

For more information, see Working with Video Components.

12.21 W

Wiki

The page in each edX course that allows both students and members of the course team to add, modify, or delete content.

Students can use the wiki to share links, notes, and other helpful information with each other.

For more information, see Hide or Show the Course Wiki Page.

12.22 XYZ

XBlock

EdX's component architecture for writing courseware components: XBlocks are the components that deliver course content to learners.

Third parties can create components as web applications that can run within the edX learning management system.

XSeries

A set of related courses in a specific subject. Learners qualify for an XSeries certificate when they pass all of the courses in the XSeries.

For more information, see https://www.edx.org/xseries.